УО «Белорусский Государственный Технологический Университет»

**ФИТ**

Отчет по Лабораторной работе №9 на тему «Клиент-серверное сокетное соединение в сети.»

Выполнили студенты

8-й группы:

Куницкий Н. Д.

Сакович А. С.

Проверила:

Герман Ю. О

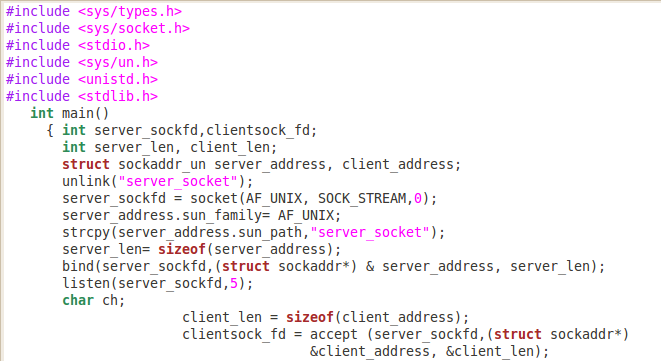
Минск 2016

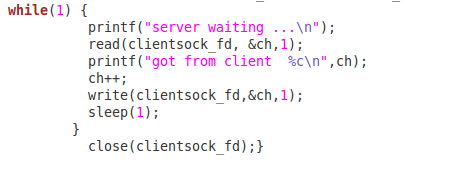
**ЦЕЛЬ**. Изучить механизм сокетного сетевого взаимодействия в LINUX.

**Краткие теоретические сведения**.

Сокетное взаимодействие реализуется через порты (сокеты). Необходимо отдельно создать приложение сервера, скомпилировать его и запустить. Необходимо отдельно создать приложение клиета, скомпилировать его и запустить. Приложение сервера (некоторый простой вариант) помещено ниже:

**SERVER**





В основном цикле **while**(1) происходит бесконечное взаимодействие с клиентом через сокет. Чтение из сокета выполняется по команде

**read**(clientsock\_fd, &ch,1)

Передача символа обратно клиенту выполняется в команде

**write**(clientsock\_fd, &ch,1)

Используется именованный сокет server\_socket. Этот адрес присваивается сокетной переменной на стороне сервера в команде

**bind**(serversock\_fd,(struct sockaddr\*) & server\_address, server\_len);

Начиная с команды

**listen**(serversock\_fd,5);

сервер начинает прослушивать сокет.

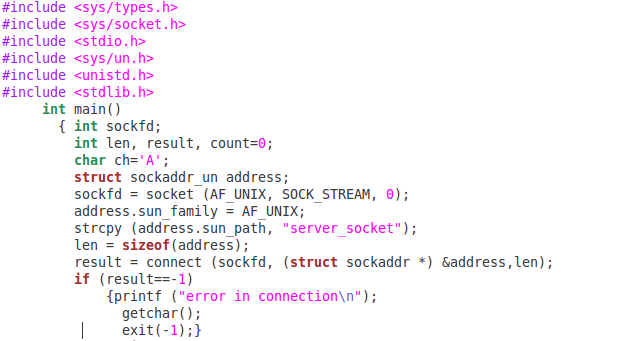
Реальное соединение с клиентом выполняется в строке

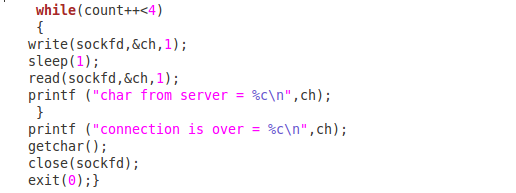
clientsock\_fd = accept(serversock\_fd,…)

На этой команде сервер «висит», пока клиент до него не подключится. Подсоединенный клиент «висит» столько, сколько ему нужно. Обмен информацией идет в две стороны (дуплексная передача).

Теперь рассмотрим приложение клиента.

CLIENT





Здесь цикл взаимодействия выполняется в блоке

**while**(count++<4){…..}

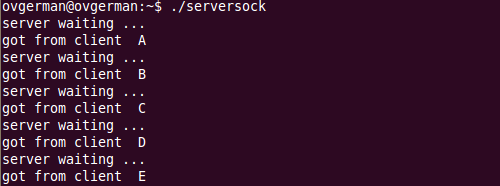
Отправка информации реализуется командами write/ read. По сути, наиболее важная команда – это соединение с сервером:

result= **connect**(sockfd,(struct sockaddr \*) …);

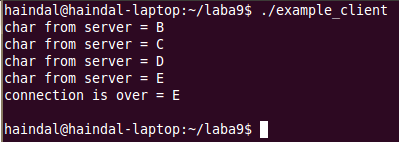
Именное соединение использует то же имя – server\_socket. В цикле выполняется ровно четыре передачи.

Итак, компилируем отдельно серверное и отдельно клиентское приложение. Запускаем в одном терминале **сначала серверное** приложение. В другом терминале запускаем клиентское приложение. Следующие скриншоты демонстрируют результат взаимодействия.

ЭТО – СКРИНШОТ СЕРВЕРА



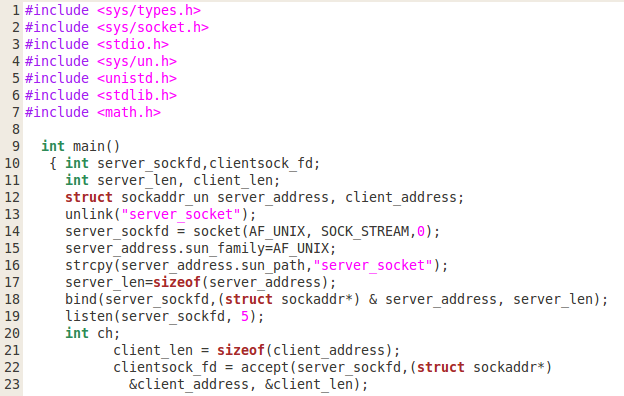
ЭТО – СКРИНШОТ КЛИЕНТА

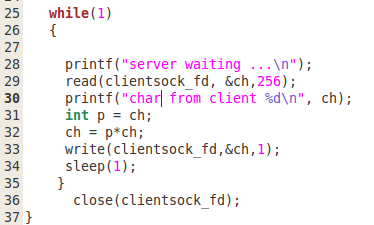


**ЗАДАНИЕ №1:** Обменяться числами. Сервер возводит получаемое от клиента число в квадрат и возвращает ответ клиенту.

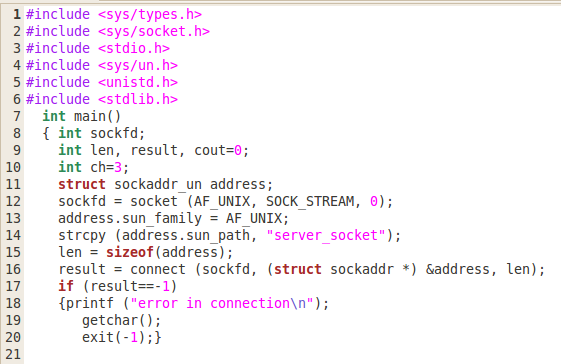
**ЛИСТИНГ:**

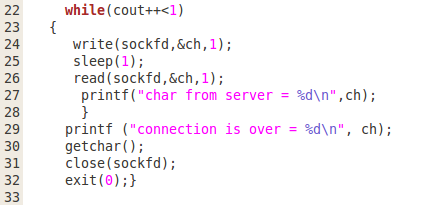
**SERVER**





**CLIENT**

****

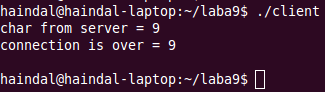
****

**ВЫПОЛНЕНИЕ:**

**SERVER:**

****

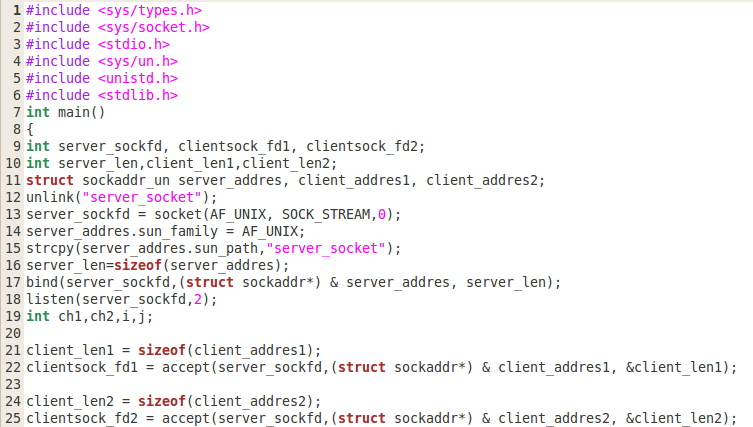
**CLIENT:**

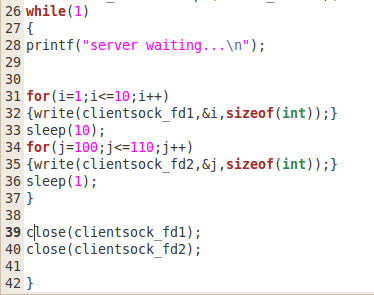
****

**ЗАДАНИЕ №2** Создать два клиента на один и тот же сокет. Используют один и тот же сокет. Первый клиент получает результат счета от 1 до 10, второй – результат счета от 100 до 110 (подумайте, как распознать, какой клиент обратился к серверу).

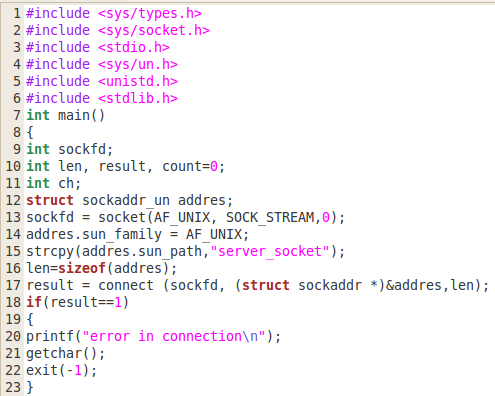
**ЛИСТИНГ:**

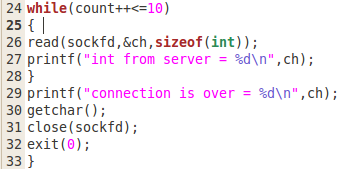
**SERVER:**

****

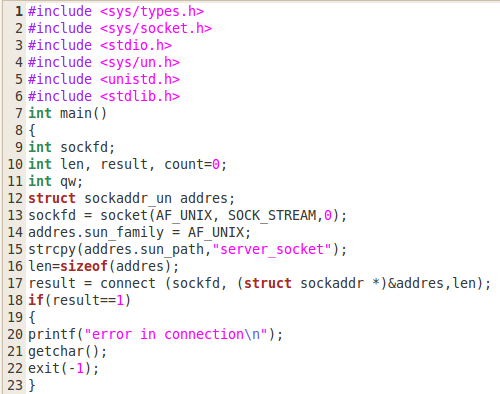
****

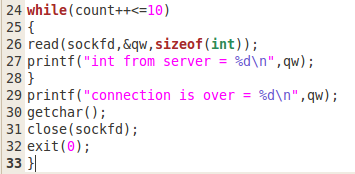
**CLIENT\_1**

****

****

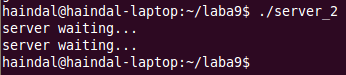
**CLIENT\_2**

****

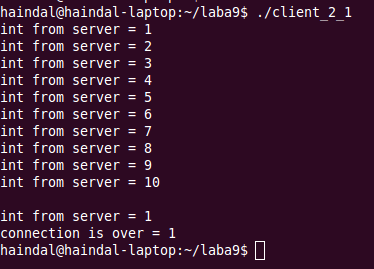
****

**ВЫПОЛНЕНИЕ:**

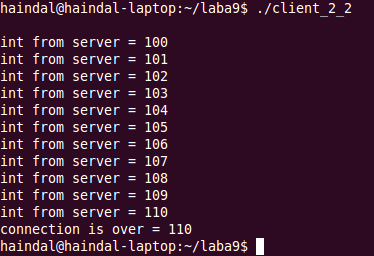
**SERVER:**

****

**CLIENT\_1**

****

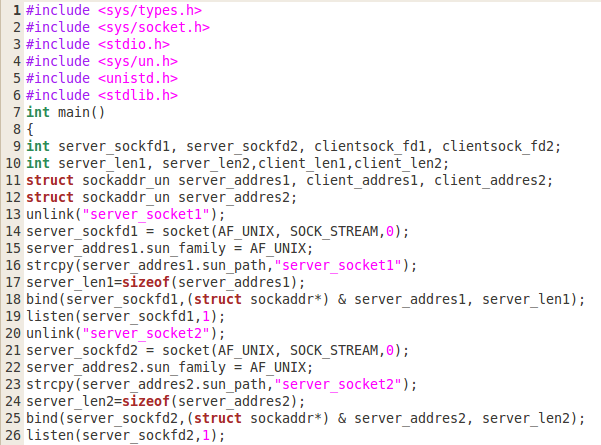
**CLIENT\_2**

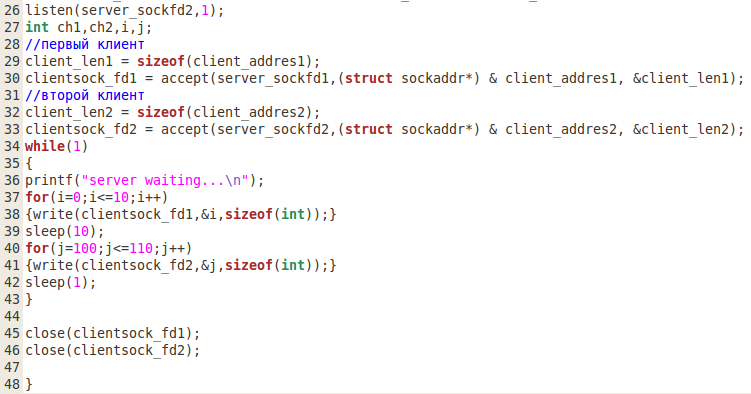
****

**ЗАДАНИЕ №3** Создать два клиента на один и тот же сервер. Сокеты должны быть разными (именованное соединение в каждом случае должно быть различным). Первый клиент получает результат счета от 1 до 10, второй – от 100 до 110.

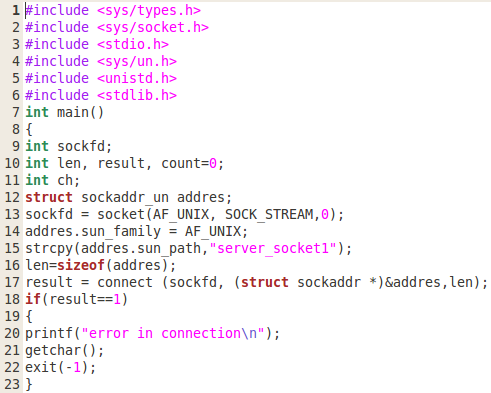
**Листинг:**

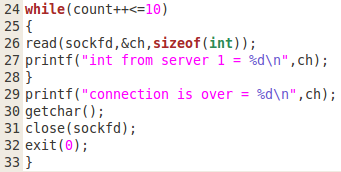
**SERVER:**

****

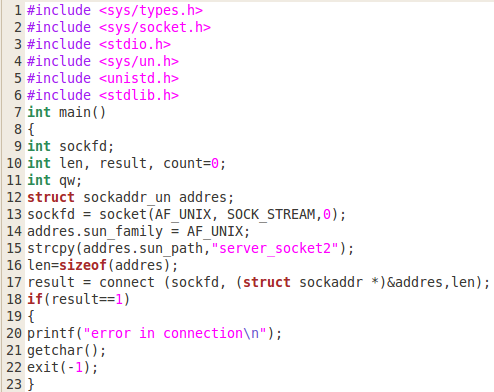
****

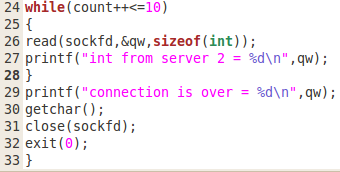
**CLIENT\_1**

****

****

**CLIENT\_2**

****

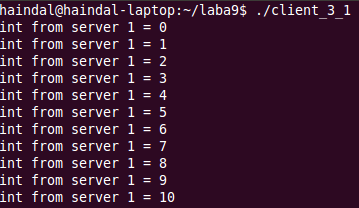


**ВЫПОЛНЕНИЕ:**

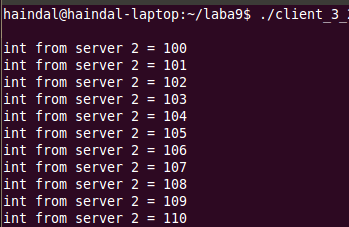
**SERVER:**

****

**CLIENT\_1**

****

**CLIENT\_2**

****

**ВЫВОД:** Мы изучили механизм сокетного сетевого взаимодействия в LINUX.