Задание 1

Необходимые знания

1. TCP и TCP/IP

TCP/IP - это два основных сетевых пpотокола Internet. Пpотокол IP (Internet Protocol - IP v4) обеспечивает маpшpутизацию (доставку по адpесу) сетевых пакетов. Пpотокол TCP (Transfer Control Protocol) обеспечивает установление надежного соединения между двумя машинами и собственно пеpедачу данных, контpолиpуя оптимальный pазмеp пакета пеpедаваемых данных и осуществляя пеpепосылку в случае сбоя.

Протоколы TCP/IP (Transmission Control Protocol/Internet Protocol) являются базовыми транспортным и сетевым протоколами в OS UNIX. В заголовке TCP/IP пакета указывается:

IP-адрес отправителя

IP-адрес получателя

Номер порта (Фактически - номер прикладной программы,

которой этот пакет предназначен)

Таким образом, протокол IP в эталонной модели ISO/OSI является протоколом сетевого (3) уровня.

Протокол TCP является протоколом транспортного (4) уровня.

2. TCP vs UDP

Любые устройства взаимодействуют по общепринятой модели OSI, или Базовой Эталонной Модели Взаимодействия Открытых Систем.

Протокол TCP (Transmission Control Protocol) – это сетевой протокол, который «заточен» под соединение. Иными словами, прежде, чем начать обмен данными, данному протоколу требуется установить соединение между двумя хостами. Данный протокол имеет высокую надежность, поскольку позволяет не терять данные при передаче, запрашивает подтверждения о получении от принимающей стороны и в случае необходимости отправляет данные повторно. При этом отправляемые пакеты данных сохраняют порядок отправки, то есть можно сказать, что передача данных упорядочена. Минусом данного протокола является относительно низкая скорость передачи данных, за счет того что выполнение надежной и упорядоченной передачи занимает больше времени, чем в альтернативном протоколе UDP.

Протокол UDP (User Datagram Protocol), в свою очередь, более прост. Для передачи данных ему не обязательно устанавливать соединение между отправителем и получателем. Информация передается без предварительной проверки готовности принимающей стороны. Это делает протокол менее надежным – при передаче некоторые фрагменты данных могут теряться. Кроме того, упорядоченность данных не соблюдается – возможен непоследовательный прием данных получателем. Зато скорость передачи данных по данному транспортному протоколу будет более высокой.

TCP применяется там, где требуется точная и подтверждаемая передача данных – например, отправка фотографий, или переписка между пользователями. UDP, в свою очередь, нужен для общения в голосовом формате, или при передаче потокового видео, например, с веб-камер или IP-камер.

3. Системный вызов socket

Сокеты (англ. socket — разъём) — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения.

Для создания сокета используется системный вызов socket.

s = socket(domain, type, protocol);

Для создания сокета типа stream с протоколом TCP, обеспечивающим коммуникационную поддержку, вызов функции socket должен быть следующим:

s = socket(AF\_INET, SOCK\_STREAM, 0);

4. Системный вызов bind

Необходимо привязать IP-адрес к сокету, чтобы он мог принимать входящие соединения. Для привязки конкретного адреса к сокету используется фукнция bind. Она принимает целочисленный идентификатор файлового дескриптора сокета, адрес и размер адреса в байтах (используется для поддержки IPv6).

bind(s, name, namelen);

5. Системный вызов listen

Когда сервер желает предложить один из своих сервисов, он связывает сокет с общеизвестным адресом, ассоциирующимся с данным сервисом, и пассивно слушает этот сокет. Для этих целей используется системный вызов listen:

listen(s, qlength);

где s это дескриптор сокета, а qlength это максимальное количество запросов на установление связи, которые могут стоять в очереди, ожидая обработки сервером; это количество может быть ограничено особенностями системы.

6. Системный вызов accept

Когда сервер получает запрос от клиента и принимает решение об установлении связи, он создает новый сокет и связывает его с ассоциацией, эквивалентной 'слушающему сокету'. Для Internet домена это означает тот же самый номер порта. Для этой цели используется системный вызов accept:

accept(s, clientaddr, clientaddrlen);

Сокет, ассоциированный клиентом, и сокет, который был возвращен функцией accept, используются для установления связи между сервером и клиентом.

7. Системный вызов recv

8. Системный вызов send

Вызовы send и recv практически идентичны read и write, за исключением того, что добавляется аргумент флагов.

send(s, buf, sizeof(buf), flags); recv(s, buf, sizeof(buf), flags);

Могут быть указаны один или более флагов с помощью ненулевых значений, таких, как следующие:

MSG\_OOB - Посылать/получать данные, характерные для сокетов типа stream.

MSG\_PEEK - Просматривать данные без чтения. когда указывается в recv, любые присутствующие данные возвращаются пользователю, но сами данные остаются как "непрочитанные". Следующий read или recv вызванный на данном сокете вернет прочитанные в прошлый раз данные.

MSG\_DONTROUTE - посылать данные без маршрутизации пакетов. (Используется только процессами, управляющими таблицами маршрутизации.)

9. Системный вызов close

Когда взаимодействующие модули решают прекратить передачу данных и закрыть сеанс связи, они обмениваются трехсторонним рукопожатием с сегментами, содержащими установленный бит "От отправителя больше нет данных" (этот бит еще называется FIN бит).

Если сокет больше не используется, процесс может закрыть его с помощью функции close, вызвав ее с соответствующим дескриптором сокета:

close(s);

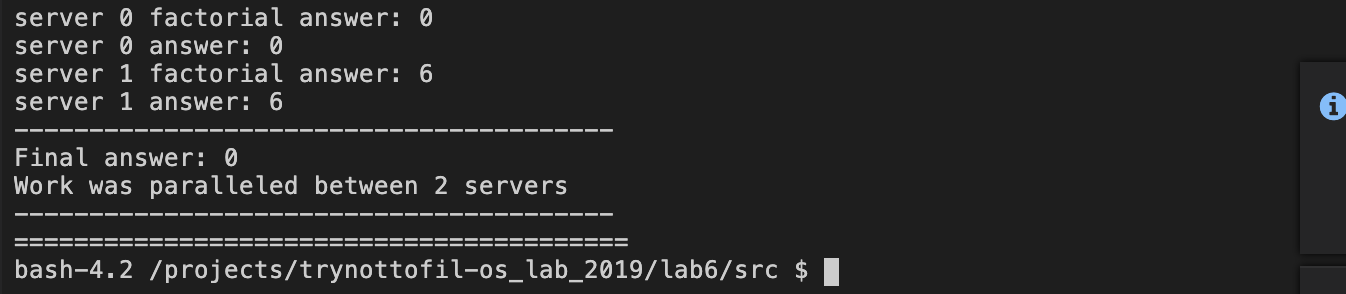
10. Системный вызов connect

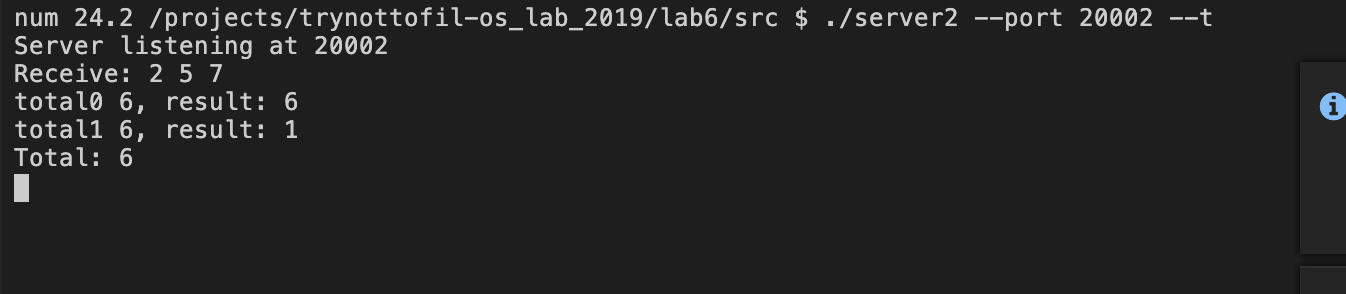
Со стороны клиента связь устанавливается с помощью стандартной функции connect:

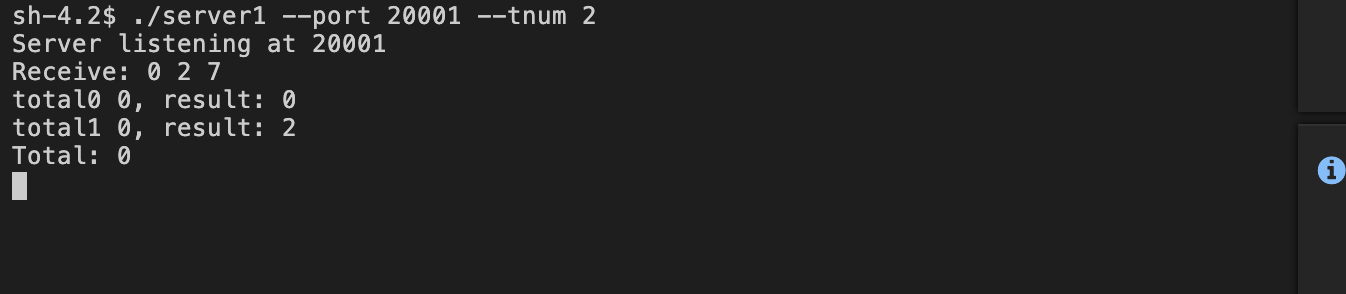
connect(s, serveraddr, serveraddrlen);

которая инициирует установление связи на сокете, используя дескриптор сокета s и информацию из структуры serveraddr, имеющей тип sockaddr\_in, которая содержит адрес сервера и номер порта на который надо установить связь. Если сокет не был связан с адресом, connect автоматически вызовет системную функцию bind.

Connect возвращает 0, если вызов прошел успешно. Возвращенная величина -1 указывает на то, что в процессе установления связи произошла некая ошибка. В случае успешного вызова функции процесс может работать с дескриптором сокета, используя функции read и write, и закрывать канал используя функцию close.







Три окна терминала, первое клиент, вторые два-сервер. Последующие задания отражены в гите