OS Lab 30

Status	approved
checkbox	☑
class	OS
due date	@Apr 21, 2021

Task

💡 30. Связь через Unix domain socket

Напишите две программы, взаимодействующих через Unix domain socket. Первый процесс (сервер) создает сокет и слушает на нем. При присоединении клиента, сервер получает через соединение текст, состоящий из символов верхнего и нижнего регистров, переводит его в верхний регистр и выводит в свой стандартный поток вывода, аналогично задаче 25. Второй процесс (клиент) устанавливает соединение с сервером и передает ему текст. После разрыва соединения клиентом, оба процесса завершаются.

Notes

https://illumos.org/man/3HEAD/socket

💡 Сокет — это абстракция конечной точки взаимодействия. Подобно тому как для работы с файлами приложения используют дескрипторы файлов, для работы с сокетами они используют дескрипторы сокетов. В UNIX дескрипторы сокетов реализованы так же, как дескрипторы файлов. В действительности большинство функций, работающих с дескрипторами файлов, таких как read или write, будут работать и с дескрипторами сокетов.

Создается дескриптор сокета с помощью функции `socket`.

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

Возвращает дескриптор файла (сокета) в случае успеха, −1 — в случае ошибки

Аргумент domain определяет природу взаимодействия, включая формат адреса. В табл. 16.1 приводится список доменов, которые определены стандартом `POSIX.1.` Имена констант начинаются с префикса AF_ (от address family — семейство адресов), потому что каждый домен обладает своим собственным форматом представления адресов.

Таблица 16.2. Типы сокетов

Тип	Описание
SOCK_DGRAM	Не ориентированы на создание логического соединения, сообщения фиксированной длины, доставка сообщений не гарантируется
SOCK_RAW	Интерфейс дейтаграмм к протоколу IP (необязателен в POSIX.1)
SOCK_SEQPACKET	Ориентированы на создание логического соединения, упорядоченность передачи данных, сообщения фиксированной длины, гарантируется доставка сообщений
SOCK_STREAM	Ориентированы на создание логического соединения, упорядоченность передачи данных, гарантируется доставка сообщений, двунаправленный поток байтов

Таблица 16.3. Протоколы для сокетов из домена Интернета

Протокол	Описание
IPPROTO_IP	Протокол Интернета IPv4
IPPROTO_IPV6	Протокол Интернета IPv6 (необязателен в POSIX.1)
IPPROTO_ICMP	Протокол управляющих сообщений Интернета
IPPROTO_RAW	Протокол простых пакетов IP (необязателен в POSIX.1)
IPPROTO_TCP	Протокол управления передачей данных
IPPROTO_UDP	Протокол пользовательских дейтаграмм

```
The protocol parameter is a protocol-family-specific value which
specifies a particular protocol to be used with the socket.  Normally
this value is zero, as commonly only a single protocol exists to support
a particular socket type within a given protocol family. However,
multiple protocols may exist, in which case a particular protocol may be
specified in this manner.
```

Таблица 16.1. Домены сокетов

Домен	Описание
AF_INET	Домен Интернета IPv4
AF_INET6	Домен Интернета IPv6 (необязательный в POSIX.1)
AF_UNIX	Домен UNIX
AF_UNSPEC	Неопределенный домен

Константа `AF_UNSPEC` обозначает неопределенный домен, который может пред-ставлять любой домен.

16.4. Установка соединения

Если мы имеем дело с сетевой службой, которая ориентирована на установление логического соединения (`SOCK_STREAM` или `SOCK_SEQPACKET`), прежде чем начать обмениваться данными, необходимо установить соединение между сокетом процесса, посылающего запрос (клиентом), и процессом, предоставляющим услугу (сервером). Для создания соединения используется функция `connect`.

```
#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr *addr, socklen_t len);
```

Возвращает 0 в случае успеха, –1 — в случае ошибки

Адрес, который передается функции `connect`, — это адрес сервера, с которым предполагается установить связь. Если сокету *sockfd* не присвоен какой-либо адрес, функция присвоит ему адрес по умолчанию.

Попытка соединения с сервером может потерпеть неудачу по нескольким причинам. Машина, с которой устанавливается соединение, должна быть включена и связана с сетью. Серверу должен быть присвоен адрес, с которым мы пытаемся соединиться, и в очереди запросов на соединение на стороне сервера должно быть достаточно места, чтобы поставить в очередь наш запрос (вскоре мы поговорим об этом более подробно). То есть приложение должно уметь обрабатывать возможные ошибки соединения.

С помощью функции listen сервер заявляет о своем желании принимать запросы на установление соединения

```
#include <sys/socket.h>

int listen(int sockfd, int backlog);
```

Возвращает 0 в случае успеха, –1 — в случае ошибки

Аргумент *backlog* определяет желаемое количество ожидающих обработки запросов, которые должны быть поставлены в очередь от имени процесса. Фактическое значение определяется самой системой, но верхний предел определен под именем `SOMAXCONN` в заголовочном файле `<sys/socket.h>`.

После вызова функции listen указанный сокет будет использоваться для приема запросов на соединение. Функция accept принимает запрос и преобразует его в соединение.

Функция accept возвращает дескриптор сокета, соединенного с клиентом, вызвавшим функцию connect. Этот новый сокет имеет тот же тип и семейство адресов, что и сокет `sockfd`. Первоначальный сокет, который передается функции accept, не связан с установленным соединением, он остается свободным для приема по-следующих запросов на соединение.Если нас не беспокоит проблема идентификации клиента, мы можем передать в ар-гументах `addr` и `len` значение `NULL`. Иначе необходимо передать в `addr` адрес буфера достаточного размера для хранения адреса, а в аргументе `len` — адрес целого числа, определяющего размер буфера. По возвращении из функции accept в буфере будет находиться адрес клиента, а по адресу `len` — фактический размер адреса. Если запросы, ожидающие обработки, отсутствуют, функция accept будет забло-кирована,

Обмен данными через сокеты является двунаправленным. Выполнение отдельных операций над сокетами можно запретить с помощью функции `shutdown`.

```
#include <sys/socket.h>

int shutdown(int sockfd, int how);
```

Возвращает 0 в случае успеха, –1 — в случае ошибки

Если в аргументе *how* передать значение `SHUT_RD`, операция чтения из сокета будет запрещена. Если передать значение `SHUT_WR`, будет запрещена операция записи в сокет. Если передать значение `SHUT_RDWR`, будет запрещена возможность передачи данных в обоих направлениях.

```
cc [ flag ... ] file ... -lsocket  -lnsl  [ library ... ]
#include <sys/types.h>
#include <sys/socket.h>


int bind(int s, const struct sockaddr *name, int namelen);


DESCRIPTION

     The bind() function assigns a name to an unnamed socket. When a socket is
     created with socket(3SOCKET), it exists in a name space (address family)
     but has no name assigned. The bind() function requests that the name
     pointed to by name be assigned to the socket.
```

SOCKADDR_UN

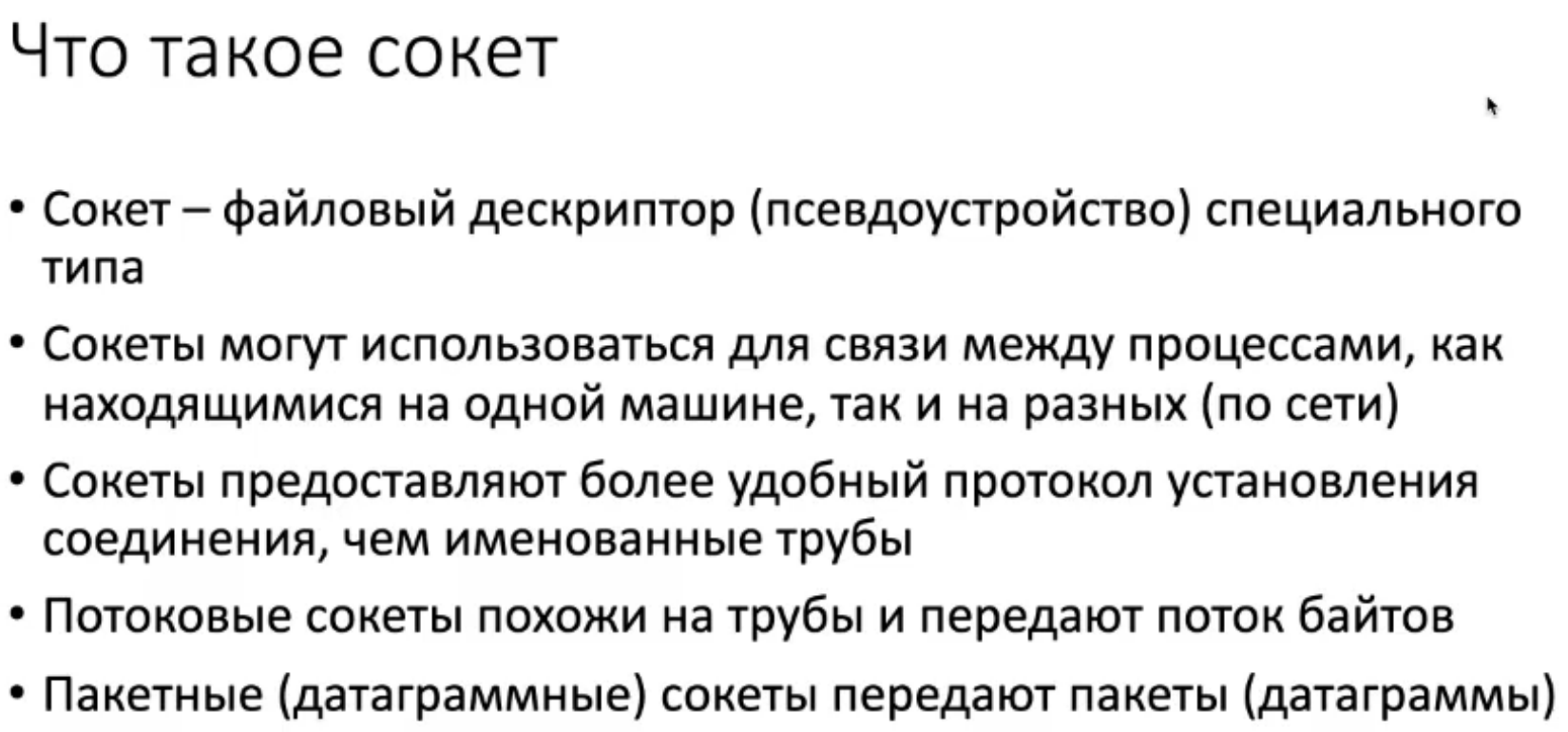
```
struct sockaddr_un
  The sockaddr_un structure specifies the address of a socket used to
  communicate between processes running on a single system, commonly known as
  a UNIX domain socket.  Sockets of this type are identified by a path in the
  file system.  The struct sockaddr_un has the following members:

        sa_family_t     sun_family      /* address family */
        char            sun_path[108]   /* path name */

  The member sun_family must always have the value AF_UNIX.  The member
  sun_path is populated with a NUL terminated array of characters that
  specify a file system path.  The maximum length of any such path, including
  the NUL terminator, is 108 bytes.
```

Поле `sun_path` содержит полное имя файла. Присваивая имя сокету домена UNIX, система создает файл типа `S_IFSOCK` с этим именем.

Этот файл существует, только чтобы сообщить имя сокета клиентам. Сам файл не может быть открыт или как-то иначе использован для взаимодействия приложений.

Если во время попытки присвоить имя сокету файл уже существует, вызов функции `bind` завершится с признаком ошибки. При закрытии сокета этот файл не удаляется автоматически, поэтому мы должны сами побеспокоиться о его удалении перед завершением приложения.

пока не поступит хотя бы один запрос. Если `sockfd` находится в небло-кирующем режиме, функция accept вернет значение `-1` и код ошибки `EAGAIN` или `EWOULDBLOCK` в переменной `errno`.На всех четырех платформах, обсуждаемых в этой книге, константа `EAGAIN` определена с тем же значением, что и `EWOULDBLOCK`.Если сервер вызовет функцию accept при отсутствии запросов на соединение, он окажется заблокированным, пока не придет хотя бы один запрос. Как вариант сервер может использовать функцию `poll` или `select` для ожидания прибытия запросов. В этом случае сокет, содержащий запросы на соединение, будет выглядеть как доступный для чтения.



Reading list

- [ ]