

- Сокет** - абстракция конечной точки взаимодействия. (введена в BSD Unix)
- универсальное средство взаимодействия параллельных процессов.

<sys/types.h> `sock_fd = socket(int socket_family, int socket_type, int protocol)` - создает конечную точку соединения (взаимодействия) и возвращает файл. дескриптор.

- **socket-family** (домен) - domain связи, выбираем семейство протоколов, которое будем использовать для связи.

AF_UNIX (AF LOCAL) - сокеты в файловом пространстве имен.

AF_INET [6] - сетевые сокеты по протоколу TCP/IP для семейств IPv4 [6].

AF_UNSPEC - не специфицированное семейство.

" gp.

- **socket-type** - тип сокета; изменение поведения `socket()` => содержит побитовую сумму любых значений (*)

SOCK_STREAM - создание 2-сторонних потоков байтов на основе установленного соединения.

SOCK_DGRAM - поддерживает дейтаграммы (сообщения без установления соединения с определенной длиной).

" gp.

(*) **SOCK_NONBLOCK** - устанавливает флаг `O_NONBLOCK` для нового окн. файлового дескриптора (запись в `fcntl`) - если возможно, файл открывается в неблокирующем режиме

SOCK_CLOEXEC - устанавливает флаг `close-on-exec (FD_CLOEXEC)` для нового окн. файл. дескриптора

- **protocol** - задает опр. протокол, используемый с сокетами. - о по умолчанию; номер протокола.

Struct socket { // описываем сокет }

```
socket_state
short
unsigned long
struct socket_wq - указатель на структуру wait-queue (очередь ожидания) для сокета.
struct file
struct sock
const struct proto_ops
```

state; // состояние сокета (enum) } {
 type; // тип сокета (*) socket_type
 flags; // флаги сокета, опр. поведение сокета
 *wq; // указатель на структуру wait-queue (очередь ожидания) для сокета.
 *file; // указатель на файловый дескриптор
 *sk; // указатель на более низкоуровневое представление сокета для исп. сетевого протокола (TCP, UDP,...)
 *ops; // указатель на таблицу операций протокола - функции, специфичные для исп. протокола.

SS_FREE - сокет свободен и не используется
 SS_UNCONNECTED - сокет создан, но еще не подключен (`socket()`, но `connect()` не вызван)
 SS_CONNECTING - сокет находиться в процессе установления соединения (после вызова `connect()`)
 SS_CONNECTED - сокет подключен и готов к общению данным
 SS_DISCONNECTING - сокет находиться в процессе разъединения или в процессе закрытия опр.-зап-з

SOCK_NOSPACE - в сокете недостаточно места для записи данных.
 SOCK_ASYNC_NOSPACE - асинхронные

" gp.

Специальный тип сокетов **socket pair** - альтернатива pipe; создает пару неименованных сокетов.

`int socketpair (int domain, int type, int protocol, int SV[2]);`

- **SV[2]** - файловые дескрипторы парных сокетов. Единственный поддерживаемый домен - AF_UNIX (AF_LOCAL).

Парные сокеты обмениваются дуплексную связь, а pipe - симплексную. - open/close управляют сторонами взаимодействия.

! Нет встроенных средств взаимосвязи.

При **SOCK_STREAM** - передача потоковых данных - байты за байтами.

И/п. Все взаимодействия = модель Клиент - Сервер => две стороны - Обслуживатель и клиент.

I. Клиент посыпает серверу сообщение и ничего от сервера не получает.

II. _____ // _____ и сервер посыпает ответ -

Сервер работает как калькулятор: клиент := 2 числа + операция (+, -, *, /)

Клиенты работают параллельно, а сервер обслуживает в режиме монопол. доступа.

Взаимодействие в сети выполняется через сетевой адрес.

Хард (аппаратная часть)

- Среда передачи
 - Витая пара - кабель
 - Беспроводной слой (Wi-Fi, Bluetooth, ...)
 - Сетевые карты

Сокет - средство взаимодействия, а адрес - адрес то-
чечка, куда и откуда передаются данные (пакеты).

`struct sockaddr {` - общая структура для описания адреса сокета.

```
sa_family_t sa_family; // тип адреса (AF_INET, AF_INET6, ...)  
char sa_data[14]; // данные адреса - массив байтов, который хранит сам адрес.  
};
```

`struct sockaddr_in {` - структура для IPv4-адресов

```
sa_family_t sin_family; // тип адреса  
in_port_t sin_port; // номер порта  
struct in_addr sin_addr; // IP-адрес  
unsigned char sin_zero = sizeof(struct sockaddr_in) - sizeof(sa_family_t) - sizeof(...); // выравнивание для структур  
};
```

`struct in_addr {` - структура для IP-адресов
in_addr_t s_addr; // 32-битный IP-адрес
};

`struct sockaddr_in6 {` - структура для IPv6-адресов

```
sa_family_t sin6_family; // тип адреса  
in_port_t sin6_port; // номер порта  
uint32_t sin6_flowinfo; // дополнительная информация  
struct in6_addr sin6_addr; // 128-битный IP-адрес  
uint32_t sin6_scope_id; // Идентификатор области для адресов в ср. областях  
};
```

`struct in6_addr {` - структура для IP-адресов (128-бит.)
uint8_t s6_addr[16]
};

`struct sockaddr_un {` - структура для (локальных) Unix-сокетов

```
sa_family_t sun_family; // тип адреса  
char sun_path[128]; // путь к файлу  
};
```

и gp. (`struct sockaddr_ll` - структура для адр. канала связи - AF_PACKET (Link layer)).

2 байта (min) — Если больше - 2 рядом стоящих слова.

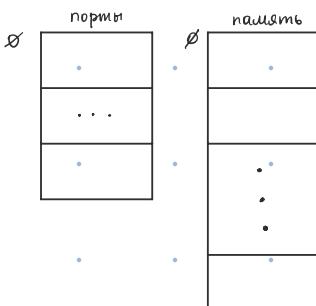
Адрес сопоставим: IP-адрес : порт - сетевой адрес

IP-адрес - уникальный адрес устройства в сети.
Порт - канал связи для программы внутри устройства.

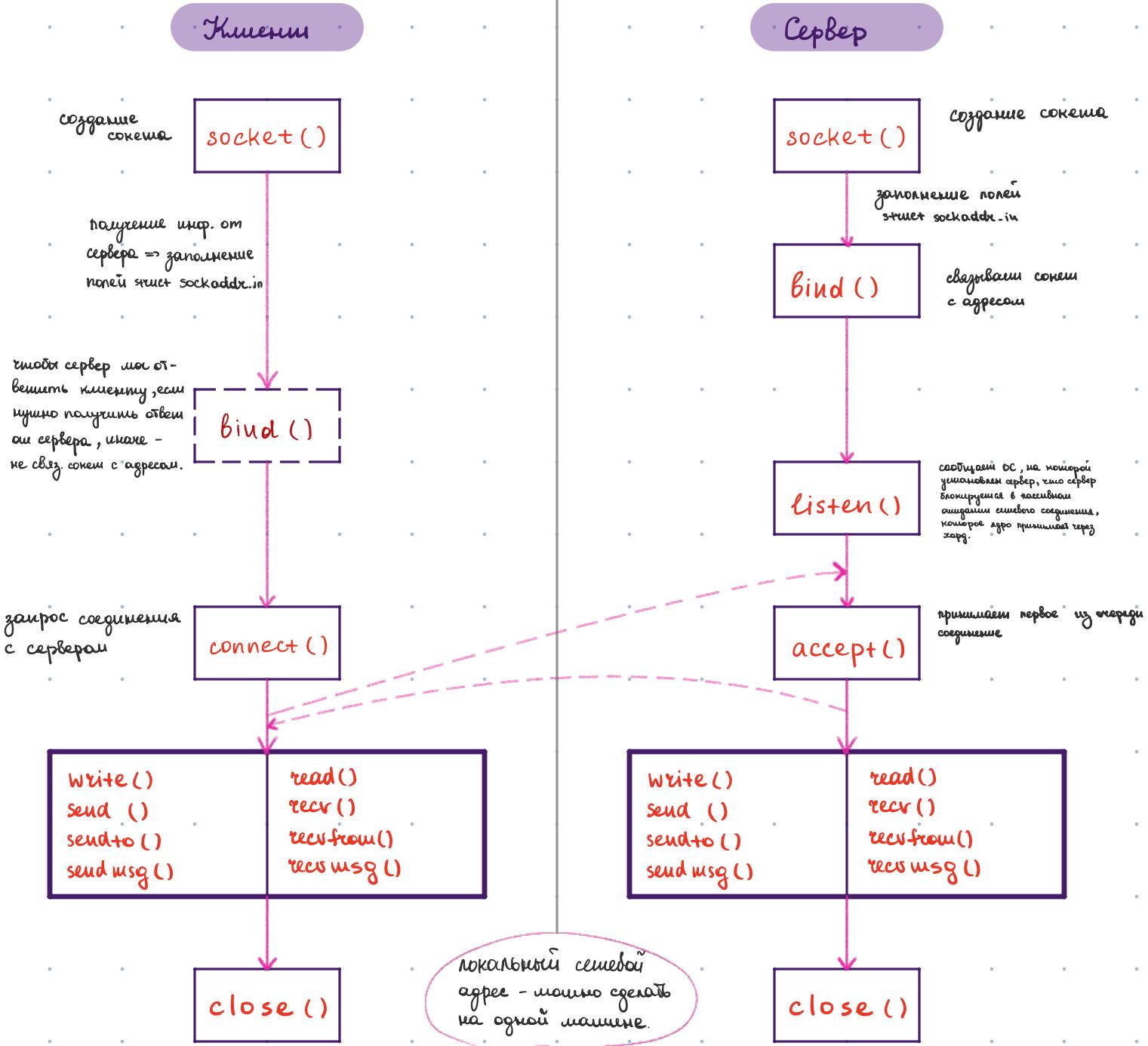
В мемодитекте int8h

Оба адр. ip-ва начинаются с \emptyset \Rightarrow наложение.

Адресация ви. устройств выполняется через порты ввода/вывода.



Протокол TCP - по умолчанию



Н/р Для сетевых сокетов написать 3 варианта "Читателей-Писателей".

- ① сервер - параллельная обработка запросов - fork
- ② сервер - параллельная обработка запросов - потоки
- ③ мультиplexирование

Чтобы сервер мог обслуживать много клиентов, accept() создает копию исходного сокета. При этом исходной сокет остается в состоянии listen(), а копия - connected => сервер продолжает прослушивать след. соединения.

Напомним из лекции! Первое поле в struct socket (state) + приведи объявления струк. вызовов с параметрами.