

Сокеты как средство взаимодействия процессов.  
Не рассм. вопросы сетей.

Сети ~ распределенные системы  
↳ у каждого хоста своя память;

В сетях - только передача сообщений,  
которые должны соответствовать  
адресам.

Пакет - сообщение с адресом (+ служебная информация);

Парные сокеты обеспечивают дуплексную связь,  
т.е. сообщения можно передавать через один  
сокет в обе стороны; (альтернатива p2p)

Сокеты для взаимодействия на отдельной станции  
машины/в сети, создаются сист. вызовом  
`int socket(int family, int type, int protocol);`  
↳ иногда наз. domain;

---

сист. вызов = API - <sup>(набор ф-ций)</sup> интерфейс, предо-  
ставляемый ОС программистам,  
разрабатывающим приложения;

Сам по себе язык Си очень маленький;  
всё, чем он оперирует, походится  
в библиотеках;

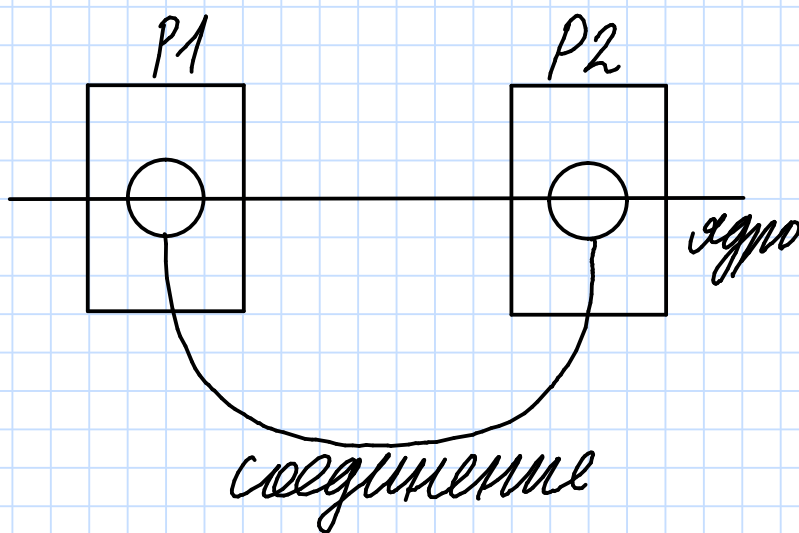
Сист. вызов переводит ОС из режима пользо-  
вателя в режим ядра, чтобы ОС могла обсу-  
жить соответствующий запрос;



„Ядро будет ветвиться по этим параметрам“;

Сокет — абстракция конечной точки взаимодействия;

→ взаимодействие на отд. машине и в сети существенно разные: в сети это будет транспортный уровень (сетевые протоколы, например TCP/IP);



Сетевые сокеты надо сопоставить с RPC;

Параметры сист. вызова `socket()`:

Family / domain — ир-во или иён:

- AF\_UNIX
  - AF\_INET IPv4
  - AF\_INET6 IPv6
  - AF\_IPX IPX
  - AF\_UNSPEC — неопределённый домен;
- сфокусированные на них

AF\_UNIX — сокеты в файловой ир-ве ииён;

Сокеты на отдельно стоящей машине (локальные сокеты) взаимодействуют через файловое ир-во ииён. Чтобы организовать взаимодействие процессов через сокеты AF\_UNIX, объявляется файл, который виден в файл. подсистеме как спец. файл (s);

type:

- SOCK\_STREAM - потоковые сокет;
- Определяет ориентированное на потоки, надёжное, упорядоченное логическое соединение между двумя сокетами;
- SOCK\_DGRAM - определяют ненадёжную службу дейтаграм без установления логического соединения, где пакеты могут передаваться без сохранения порядка;  
(широковещательная передача данных);

Почему логическое?

- SOCK\_RAW (raw - "сырой", прямой);  
↳ низкоуровневый  
(низкоуровневые сокет);

protocol:

обычно ставится 0 - протокол назначается по умолчанию;

Пример: для AF\_INET SOCK\_STREAM - протокол TCP;

Но можно задать протокол предопределённой константой;  
(макрос)

Предопределённые типы

макрос  
≠  
макросопределение

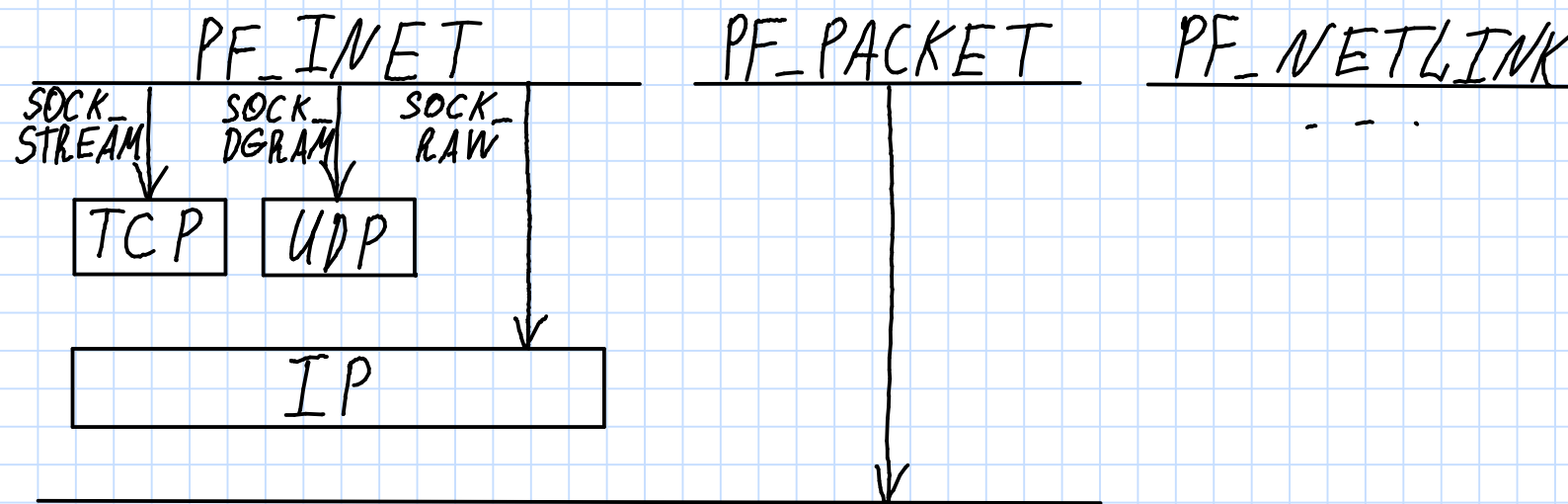
На самом деле макрос - #define, которая подставляется в код;



- IPPROTO\_\*

Пример: IPPROTO\_TCP;

## BSD сокеты



## Сетевое ядро

Сокеты Packet созданы для непосредственного доступа приложений к сетевым устройствам;

Сокетов NetLink очень много;  
Основные:

- NETLINK\_USERSOCK;
- NETLINK\_FIREWALL;

Созданы для обмена данными между частями ядра и ядро-пользователями

Эффективный ~ оптимизированный  
(например, по памяти)

В Linux есть в.ф.с. `proc`, созданная специально для того, чтобы в пр-ве пользователя можно было получить информацию о выполняемых процессах;

(Будет в л/р)

Но в ядре информации значительно больше (и о процессах, и о ресурсах);

Очень важно иметь возможность получить её;

Ядро предоставляет средства для получения этой информации. Одним из таких средств являются сокет `NETLINK`;

`socket - API`;

В ядре `socket` вызывает `sys_socket`;

```
<net/socket.c>
asm/long sys_socketcall
(int call, unsigned long *args)
```

Её текст — switch, переключатель  
ядро на разные ф-ции, связанные  
с сокетом;

```
{
    (будет работать с ней позже)
    ф-ция ядра, которая позволяет полу-
    чить информацию из user в kernel;
    int err;
    if copy_from_user(a, args, nargs[call])
        return -EFAULT;
    a0 = a[0];
    a1 = a[1];
    switch (call)
    {
        case SYS_SOCKET: err = sys_socket(a0, a1, a[2]);
                        break;
        case SYS_BIND: err = sys_bind(a0, (struct
                        sock_addr *)a1, a[2]); break;
        case SYS_CONNECT: err = sys_connect(...);
                        break;
        default: err = -EINVAL;
                break;
    }
    return err;
}
```

Код ядра часто ведется по switch

В switch переключаются ф-ции т.н. сетевого стека. Для них определены предопределенные константы (макροопределения):

```
<include/linux/net.h>
#define SYS_SOCKET 1
#define SYS_BIND 2
#define SYS_CONNECT 3
#define SYS_LISTEN 4
```

(строковая  
и числовая  
идентификатор)

...

```
asm inline long sys_socket(int family, int type,
                           int protocol)
```

{

```
    int retval;
```

```
    struct socket *sock;
```

```
    retval = sock_create(family, type, protocol, &sock);
```

```
    return retval;
```

}



struct socket (нет в 6 версии ядра)

```
{  
    socket_state      state;  
    short             type;  
    unsigned long     flags;  
    const struct proto_ops *ops;  
    struct fasync_struct *fasync_list;  
    struct file        *file;  
    struct sock        *sk;  
    wait_queue_head_t  wait;  
};
```

state - у сокета есть состояние  
(очень важно для сетевых сокетов и надежного  
соединения TCP/IP);

Надежное соединение ~ запросы сопровождаются  
ответами;  
(т.е. двойное рукопожатие)

Надежность - слабое место протокола TCP/IP;

DDOS - атаки, полуоткрытое соединение

DDOS - атака - сервер атакуется большим кол-вом  
запросов, которое он не может обработать;

State:

- SS\_FREE;
- SS\_UNCONNECTED;
- SS\_CONNECTING;
- SS\_CONNECTED;
- SS\_DISCONNECTING;

Flags - исп. для синхронизации доступа;  
struct proto\_ops - действия на сокетах;  
(protocol operations)

Здесь можно зарегистрировать  
свои функции работы с сокетами;

struct file описывает открытые  
файлы;

В системе есть таблица открытых  
файлов;

Сокеты описываются как открытые файлы (они  
не хранятся во вторичной памяти, это файлы  
спец. типа);

↳ можно увидеть только в сокетах в файловом  
пр-ве или в (AF\_UNIX);

"В UNIX все файлы, если это не процесс"

wait\_queue\_head\_t wait - очередь  
отсидания;

struct sockaddr - обращение к сокету  
выполняется по адресу (сокеты  
адресуются).

Адреса (адресация) сокетов:

```
struct sockaddr  
{
```

взаимодействие на сокет-  
техосущ. модели  
клиент-сервер

```
    sa_family_t sa_family;  
    char sa_data[14]; //не детализировано  
}
```

Такая структура адреса не подходит для интернета, т.к. там необходимо указывать номер порта и сетевой адрес. Для интернета разработана другая структура:

```
struct sockaddr_in  
{
```

```
    sa_family_t sa_family;  
    unsigned short int sin_port;  
    struct in_addr sin_addr;
```

выравнивание  
полей

```
    unsigned char sin_zero[sizeof(struct sockaddr) -  
        sizeof(sa_family_t) - sizeof(uint16_t) -  
        sizeof(struct in_addr)];
```

```
};
```

Основное правило программирования:

Ни один тип, ни одна пер-кая, ни одна до-ция не могут использоваться до объявления, определения, описания;

# Понятия Сайтов

Аппаратный

Сетевой

Маршрут

Обратить внимание на 4 главу книги  
Стивена по сетям