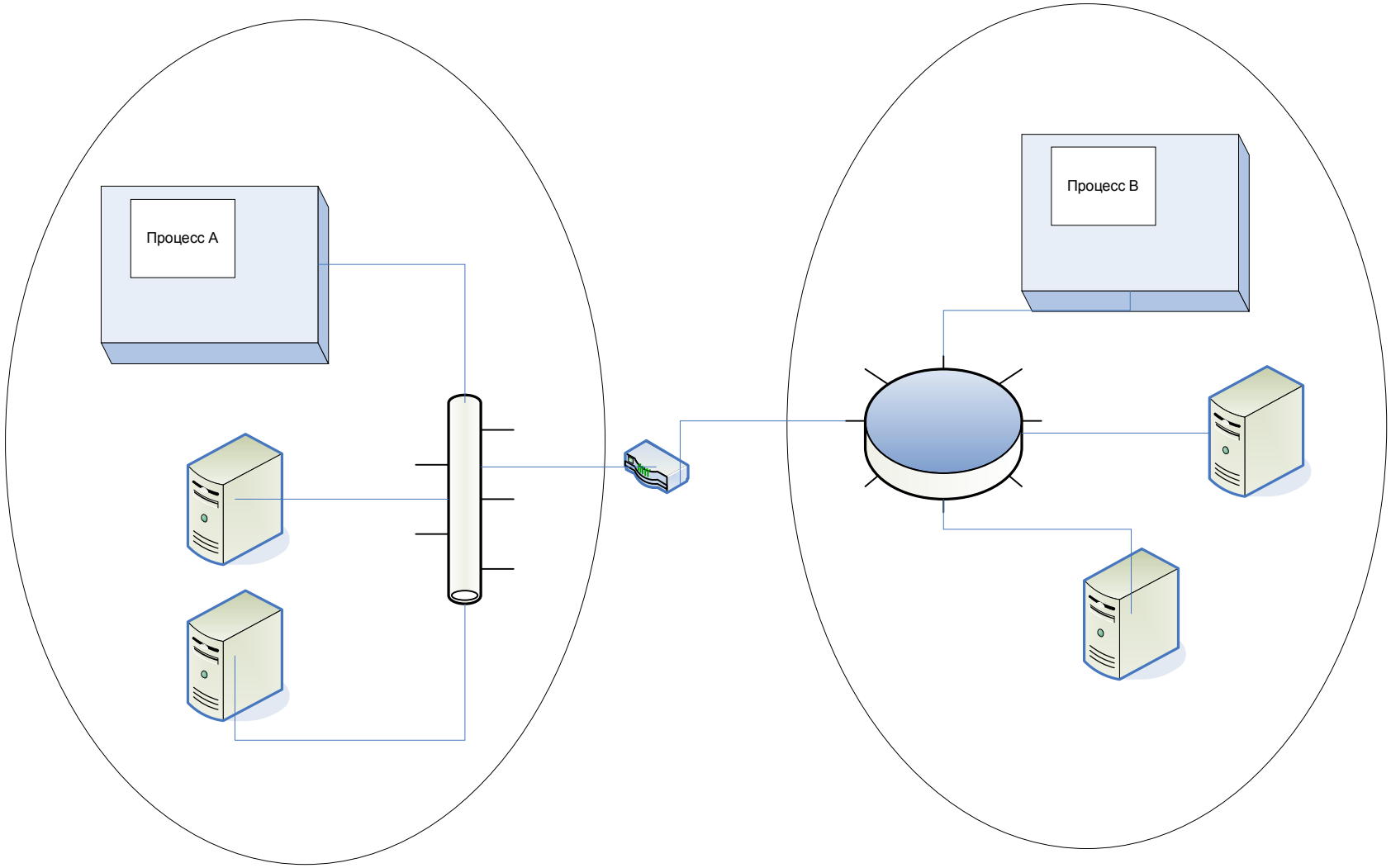
UNIX

Лекция 9

Постановка задачи



Семейство протоколов TCP/IP

1969 г. Сеть ARPANET.

Коммуникационная инфраструктура включает 3 объекта:

- сеть;
- хост (host);
- процесс

Коммуникационная инфраструктура отвечает за маршрутизацию и доставку данных между хостами.

Хосты обеспечивают доставку данных процессам.

Коммуникационный протокол – это набор правил и форматов данных, необходимых для установления связи и передачи данных.

4 уровня протоколов:

- Уровень приложений/процессов
- Транспортный уровень
- Уровень Internet
- Уровень сетевого интерфейса

Протоколы

Протоколы сетевого интерфейса

- Описывают конкретную **физическую** сеть. Описания сетей Ethernet, Token Ring, оптоволоконных сетей и т.д.
- С помощью этих протоколов осуществляется передача данных между коммуникационными узлами, подключенными к одному и тому же сетевому сегменту.

Протоколы Internet

- Маршрутизация сообщений между сетями, т.е. каким образом сообщение из одной сети дойдет до другой. Протокол **IP** отвечает за доставку сообщений между хостами в рамках единой гетерогенной сети.
- Передача данных между **хостами**, подключенными к различным сетям.
- Основная функция – выбор маршрута следования данных (маршрутизация).
- Сетевые элементы, осуществляющие передачу данных из одной сети в другую – шлюзы (маршрутизаторы, gateway, rooter).
- ***Шлюз*** – это машина, которая включает в себя реализацию нескольких протоколов сетевого интерфейса (маршрутизатор или рутер).
- Шлюзы имеют несколько сетевых интерфейсов, подключенных к различным физическим сетям.

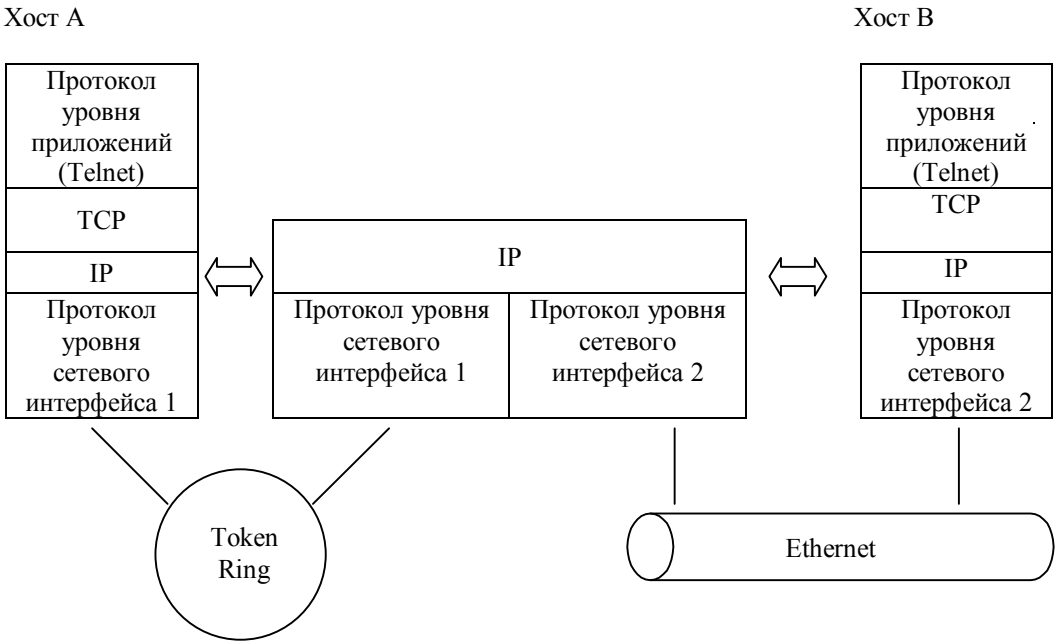
Протоколы транспортного уровня

- Доставка сообщений между **процессами**. Основные протоколы: **TCP, UDP**.
- Передача данных между процессами, выполняющимися на разных хостах. Могут обеспечивать гарантированную доставку, создание виртуального канала и т.п.

Протоколы уровня приложений

- Определяют, что следует делать с сообщением.
- Например: ftp, telnet, http и т.п.

Общая схема



Уровень приложений процессов	Передача файлов FTP	Электронная почта STMP	Эмуляция терминала Telnet	Сетевая файловая система NFS

Транспортный уровень	Transmission Control Protocol TCP	User Datagram Protocol UDP

Уровень Internet	Internet Protocol IP

Уровень сетевого интерфейса	Ethernet, Token Ring, FDDI, serial
	Витая пара, коаксиальный кабель, волоконно-оптический канал

Протокол IP (Internet Protocol)

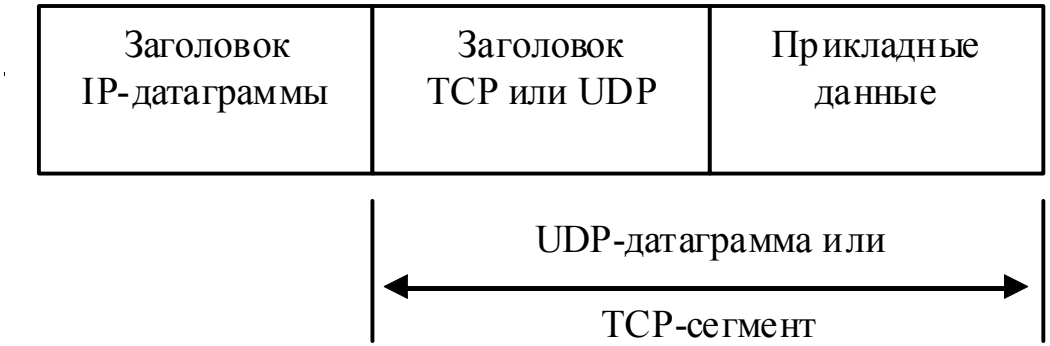
- Базовая доставка данных в гетерогенной коммуникационной среде.
- Протокол IP отсылает сообщение - IP-датаграмму.

IP-датаграмма:

а) заголовок IP;

б) все остальное (заголовок UDP, сегменты TCP);

в) прикладные данные.



UDP-датаграмма

Простейшая форма передачи прикладных данных.

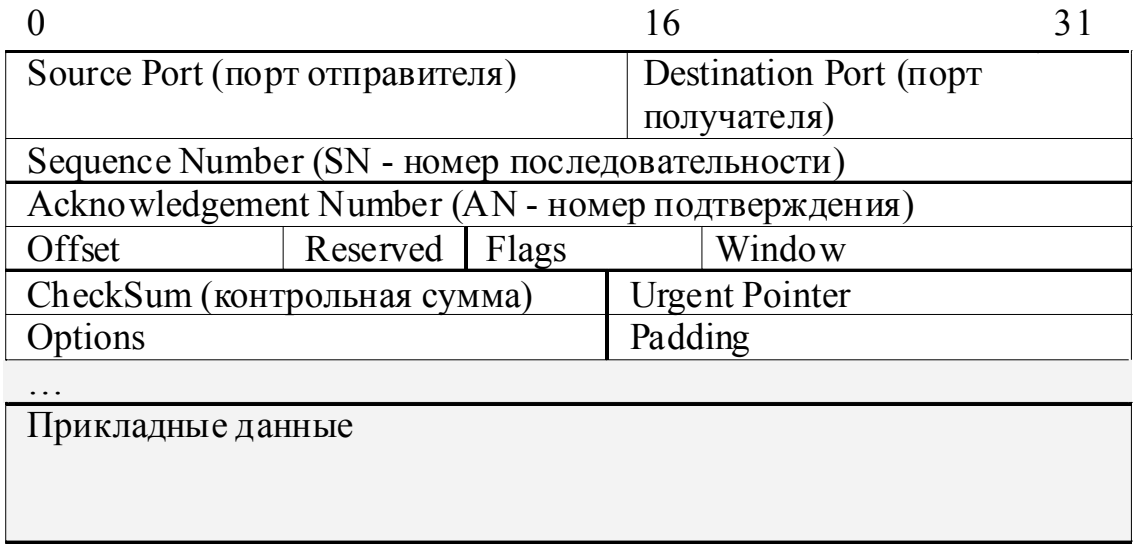
Заголовок UDP:

0	16	32	48	64
Source Port (порт источника)	Destination Port (порт приемника)	Length (длина)	Checksum (контрольная сумма)	Прикладные данные

Протокол TCP

- ***TCP*** (Transmission Control Protocol) - это протокол транспортного уровня, отвечающий за **надежную** доставку данных между пользователями в ненадежной сетевой инфраструктуре.
- Данные передаются в виде пакетов – *сегментов*. Для различных сетей величина передаваемого пакета различна (примерно 1.5k для Ethernet, 4k для FFDI).

Формат TCP-сегмента:



Sequence Number – порядковый номер сегмента в потоке;
Acknowledgement Number – номер подтверждения – сообщает отправителю количество полученных от него последовательных октетов данных (номер первого неподтвержденного октета или номер следующего ожидаемого октета);
Offset – указывает на начало данных сегмента;
Flags::
URG - экстренные данные
ACK - подтверждение
SYN - часть "тройного рукопожатия"
FIN - сторона прекращает передачу данных
Window –количество данных, которое адресат готов принять,

Порты

Номер порта	Название	Назначение
7	echo	Эхо
20	ftp-data	Передача данных по протоколу FTP
21	ftp	Управляющие команды протокола FTP
23	telnet	Удаленный доступ
80	www-http	Word Wide Web (Hyper Text Transfer Protocol)

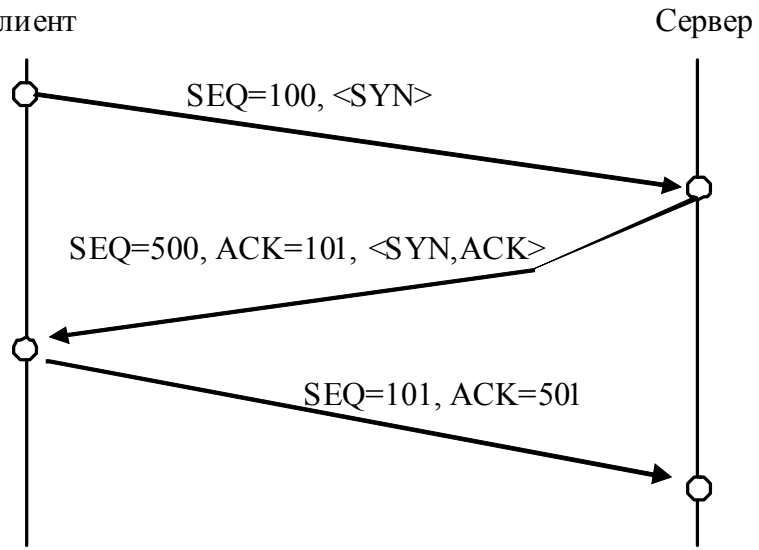
Установка связи между двумя процессами протокола TCP

Состояния TCP-сеанса

- Выбор начальных номеров потоков (сегментов) для уменьшения вероятности обработки "потерявшихся" сегментов. Номер выбирается из адресного пространства, составляющего 2^{32}.

Начальная фаза сеанса. "Тройное рукопожатие" (Three-way handshare)

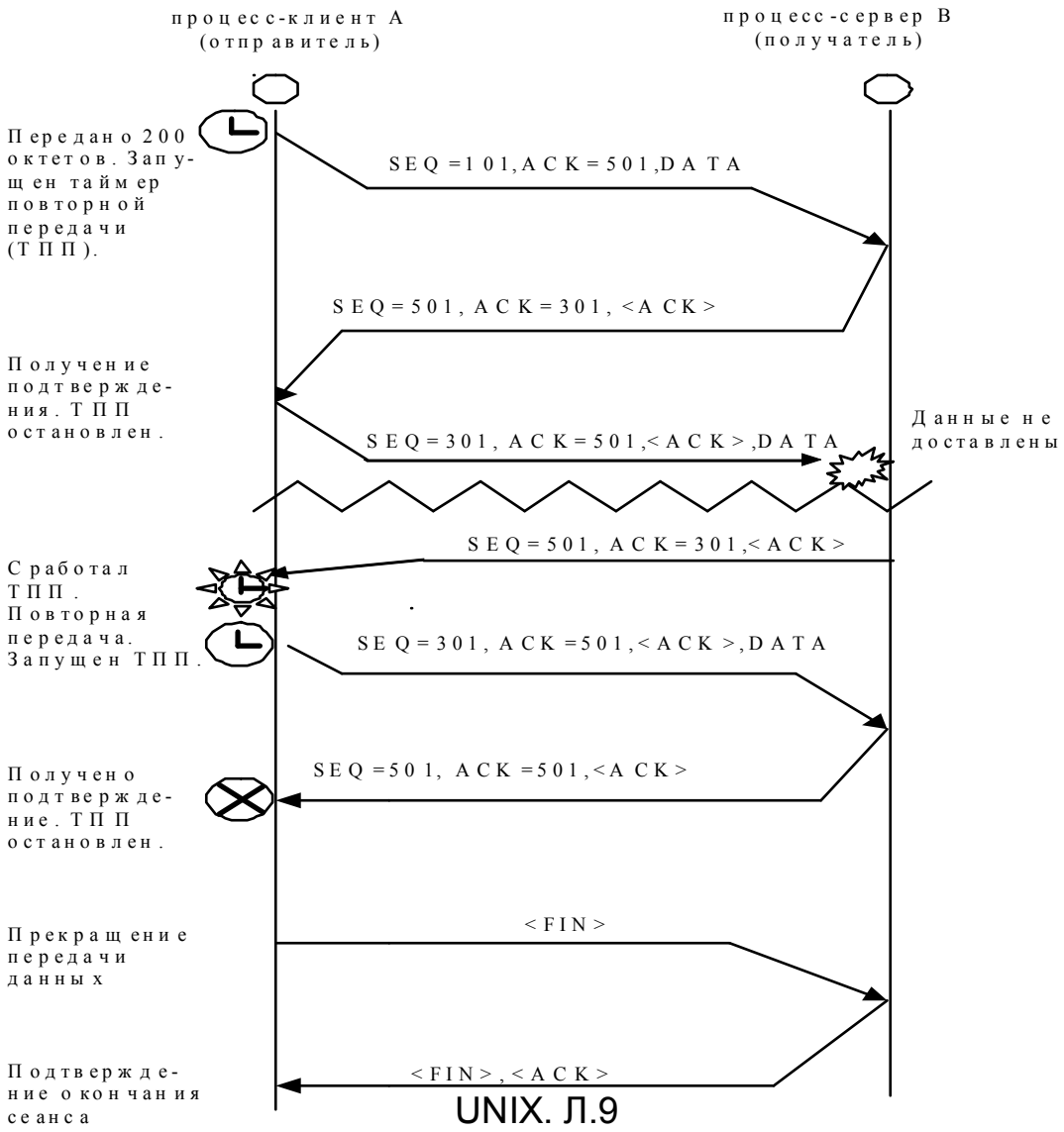
- Клиент посылает серверу сегмент SYN (определяет номер порта и порядковый номер потока)
- Если сервер готов принять данные от клиента, он создает логический канал и отправляет клиенту сегмент с установленным своим начальным порядковым номером потока и флагами SYN и ACK.



Передача TCP-сегментов, каждый из которых содержит подтверждение полученных данных и новое значение окна. Дублированные данные тоже подтверждаются (хотя и уничтожаются).

- Ожидание недостающих данных в случае неупорядоченного потока.

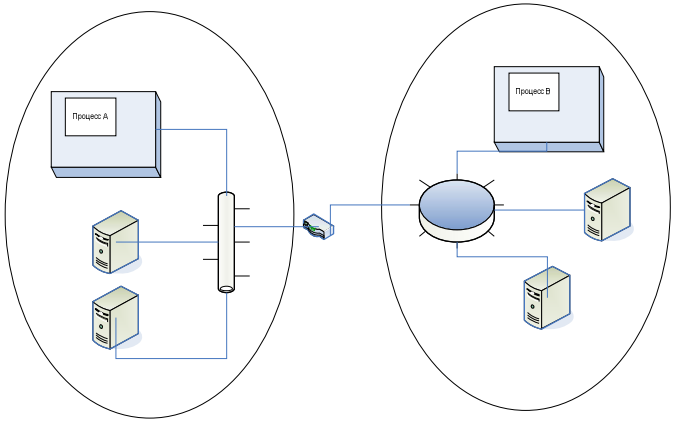
Процесс передачи

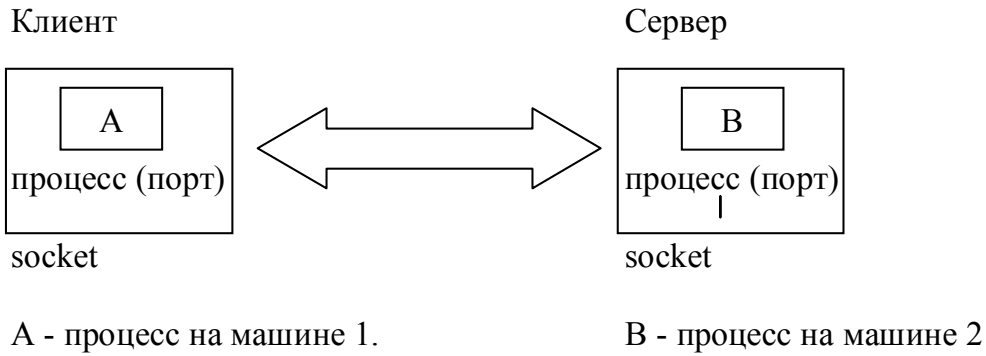


ГНЕЗДА

Проблема взаимодействия процессов, работающих на разных машинах. В BSD UNIX 4.2. введено понятие гнезда (socket).

Гнездо - это совокупность независимых от протокола услуг по организации сетевого интерфейса, обеспечивающих работоспособность методов IPC в масштабах сети.





Гнезда различаются способами адресации, используемыми транспортными протоколами, способами передачи данных и видам связи.

Способы адресации

- *Домен гнезда* определяет способ его адресации. Каждое гнездо может относиться к тому или иному домену.
- В зависимости от типа домена используются разные форматы адресов гнезд и базовые транспортные протоколы.
- Два основных вида адресации гнезд:

1. Домен UNIX (AF_UNIX). Формат адреса – путевое имя UNIX. С помощью домена UNIX обычно адресуют процессы на одной машине.
2. Сетевой домен INET (AF_INET). Формат адреса – хост-имя и номер порта.
   Обращение к гнезду осуществляется указанием **IP-адреса** хост-машины (четыре байта, которые могут адресовать любую машину в сети) и **номера порта**.

Используемый транспортный протокол

- Гнезда могут работать как с протоколом TCP (Transmission Control Protocol – протокол контроля передачи), так и с протоколом UDP (User Datagram Protocol).
- Протоколы определяют, как должно выглядеть сообщение с точки зрения способа и формата передачи по сети.

1. UDP служит для передачи датаграмм. Это простой по своей структуре, быстрый, но ненадежный протокол.
2. Протокол TCP поддерживает контроль передачи данных, организует надежную передачу, служит для организации каналов связи между процессами.

Способ передачи данных

Для каждого гнезда назначается тип, определяющий способ передачи данных между двумя гнездами:

1. SOCK_STREAM – сокет потоков, через который осуществляется надежная последовательная передача данных без дублирования с предварительным установлением связи. При этом сохраняются границы сообщений. Виртуальный канал (virtual circuit) характеризуется тем, что данные передаются в виде двунаправленного потока последовательно, с достаточной степенью надежности, без дублирования.
   Для передачи данных используется протокол TCP. Реализуется обычно в схемах типа «клиент-сервер». Этот тип сокетов надежный, но медленный и требующий больших затрат.
2. SOCK_DGRAM – это самый простой тип сокетов, быстро, но ненадежно определяющий пересылку пакета.
   Датаграмма характеризуется тем, что условие последовательности пересылки не выполняется, надежность передачи низкая, скорость высокая (используется там, где скорость важнее надежности). Используется протокол UDP.

Вид связи

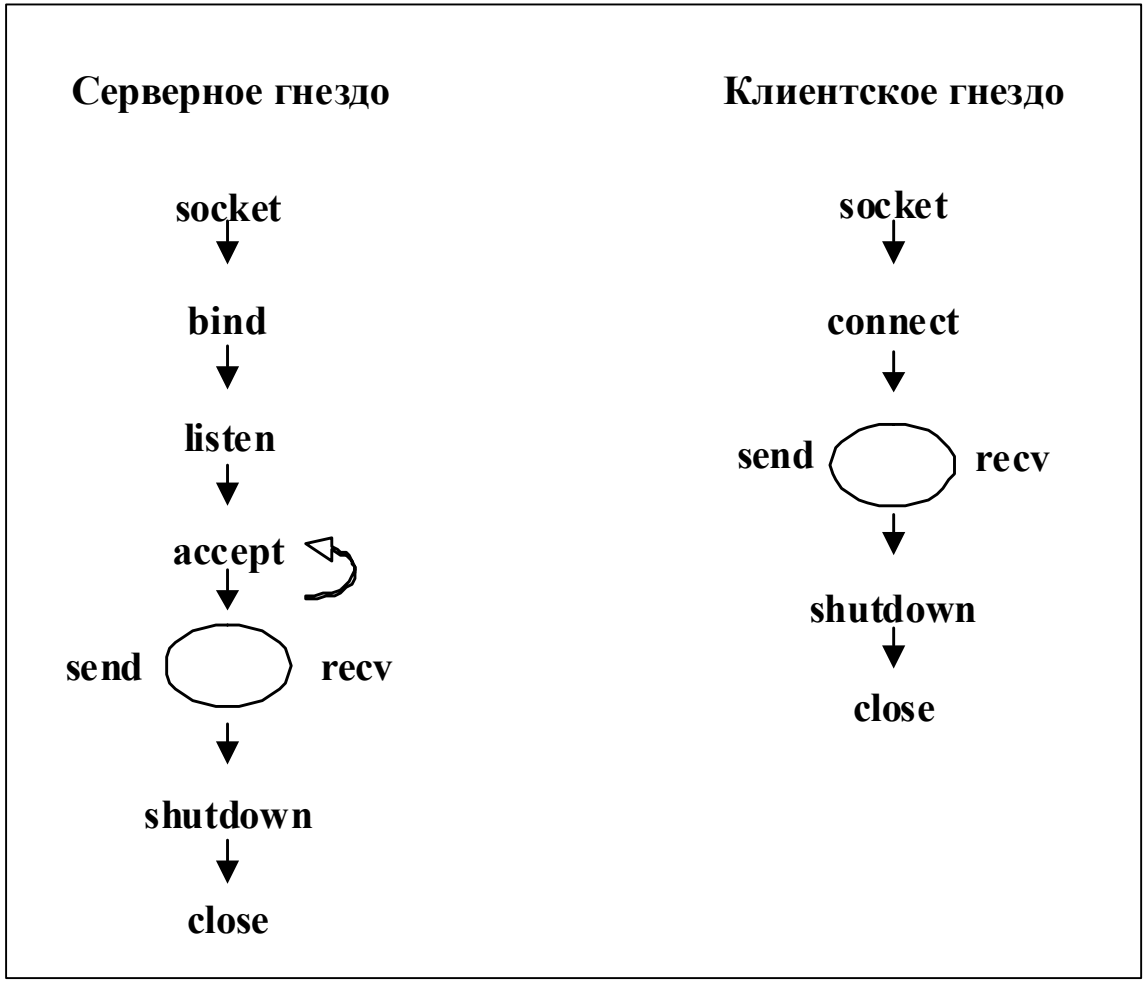
Два основных класса гнезд по видам связи:

1. **С установленным (предварительным) соединением** (адреса отправителя и получателя указываются заранее). Схема работает для архитектуры типа **клиент-сервер**. Адрес сервера известен всем клиентам заранее, и клиент настраивается на сервер. Такие сокеты используют **каналы**, образованные между процессами.
2. **Без установленного (предварительного) соединения.** В этом случае адреса гнезд отправителя и получателя передаются с каждым сообщением, посылаемым из одного процесса в другой. В этом случае мы имеем дело с **одноранговой системой**, в которой все гнезда равноправны, т.е. каждый процесс может стать клиентом или сервером.

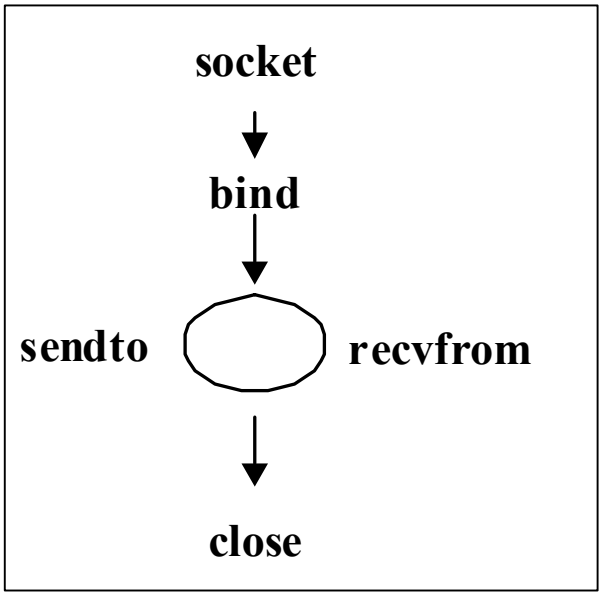
Работа с гнездами

Вид соединения	С установленным соединением	Без установленного соединения
Тип (способ связи)	Виртуальный канал	Датаграмма
Протокол	TCP	UDP
Схема	Клиент-сервер: Серверному гнезду назначается общеизвестный адрес, и оно непрерывно ожидает прибытия клиентских сообщений. Назначать адреса клиентским гнездам не нужно, т.к. они никому не нужны.	Одноранговая схема: Каждому гнезду назначается адрес, и процесс может посылать сообщения другим процессам, используя адреса гнезд.

Схема организации системы клиент-сервер (установленная связь, виртуальный канал)

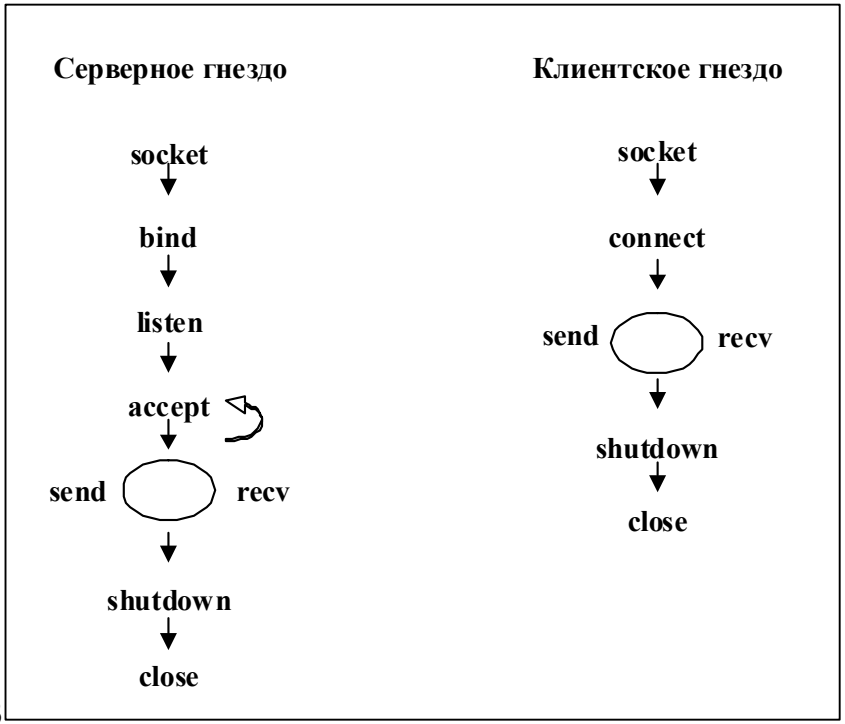


Одноранговая система (без установленной связи, датаграмма)



Функции

- **socket -** создание гнезда заданного типа с указанием протокола
- **bind** - присваивание имени гнезду. Фактически, это означает регистрацию гнезда в системе.
- **listen -** задание количества ожидающих клиентских сообщений, которые можно поставить в очередь к одному серверному гнезду
- **accept -** прием запроса на соединение от клиентского гнезда
- **connect -** посылка запроса на соединение в серверное гнездо
- **send** (sendto) - передача сообщения в удаленное гнездо
- **recv** (recvfrom) - прием сообщения из удаленного гнезда
- **shutdown -** закрытие части полнодуплексного соединения для чтения и/или записи
- **close -** закрытие гнезда



int **socket**(int domain, int type, int protocol)

#include <sys/types.h>

#include <sys/socket.h>

Создает сокет заданного типа и с указанным протоколом.

- domain – определение правила именования гнезда:
  - AF_UNIX - домен UNIX
  - AF_INET - домен Internet
- type – тип гнезда:
  - SOCK_STREAM – Сообщения передаются в виде упорядоченного двунаправленного потока байтов с высокой степенью надежности и предварительным установлением соединения
  - SOCK_DGRAM – Межпроцессное взаимодействие обеспечивается с помощью датаграмм. Сообщения передаются быстро, но с низкой степенью надежности.
  - SOCK_SEQPACKET – Двунаправленная последовательная высоконадежная передача сообщений фиксированной максимальной длины с предварительным установлением соединения
- protocol – указывает конкретный транспортный протокол (TCP или UDP). Значение его зависит от значения domain. Если protocol = 0, то ядро само выбирает для указанного домена соответствующий протокол.

int **bind**(int sockfd, struct sockaddr *sa, int addrlen)

Присваивает сокету имя.

- sockfd - дескриптор сокета - значение которое, было возвращено функцией socket.
- struct sockaddr *sa - адрес гнезда, указатель на структуру которая содержит адрес сокета. Структура адреса сокета зависит от домена, которому принадлежит гнездо.
- sockaddr - это некая "заглушка", вместо которой подставляется конкретный тип адреса: *struct sockaddr_un* или *struct sockaddr_in*.

```
struct sockaddr
{   ushort sa_family;      //определение коммуникационного домена
    char sa_data[14];      //адрес
}
```

- addrlen - размер адреса (размер структуры, на которую указывает аргумент sa).

int **connect**(int sockfd, struct sockaddr *sa, int addrlen)

- Вызывается в **клиентском** процессе для установления соединения с серверным сокетом.
- sockfd - дескриптор сокета (см. функция socket).
- sa – это указатель на адрес объекта.
- addrlen - размер объекта (в байтах), на который указывает аргумент sa.

*int **listen**(int sockfd, int backlog)*

- Вызывается **серверным** процессом для создания сокета, ориентированного на установление соединения.
- *sockfd* - дескриптор сокета
- *backlog* **-** максимальное число запросов на установление соединения, которые могут быть поставлены в очередь к данному сокету. В большинстве UNIX-систем значение аргумента *backlog* лежит в пределах от 5 до 10.

*int **accept**(int sockfd, struct sockaddr *clntaddr, int *addrlen)*

- Извлечение запроса из очереди и создание нового сокета.
- Вызывается в **серверном процессе** для установления соединения с клиентским сокетом (которое делает запрос на установление посредством вызова функции connect).
- *sockfd* - дескриптор сокета
- *clntaddr* – это указатель на адрес объекта типа *struct sockaddr*.
- *addrlen* изначально устанавливается равным максимальному размеру объекта, указанному аргументом *clntaddr*. При возврате он содержит размер адреса клиентского сокета, на который указывает аргумент *clntaddr*.
- Если аргумент *clntaddr* или аргумент *addrlen* имеет значение NULL, эта функция не передает имя клиентского сокета обратно в вызывающий процесс.
- В случае неудачи рассматриваемая функция возвращает –1. В противном случае она возвращает **дескриптор нового сокета**, с помощью которого серверный процесс может взаимодействовать исключительно с данным клиентом.

int ***shutdown****(int sockfd, int mode)*

Закрытие части полнодуплексного соединения между серверным и клиентским сокетами.

- *sockfd* – дескриптор сокета, возвращенный функцией *socket*.
- *mode* – режим закрытия

  0 – канал закрывается для чтения. При попытке продолжить чтение будут возвращаться нулевые байты;

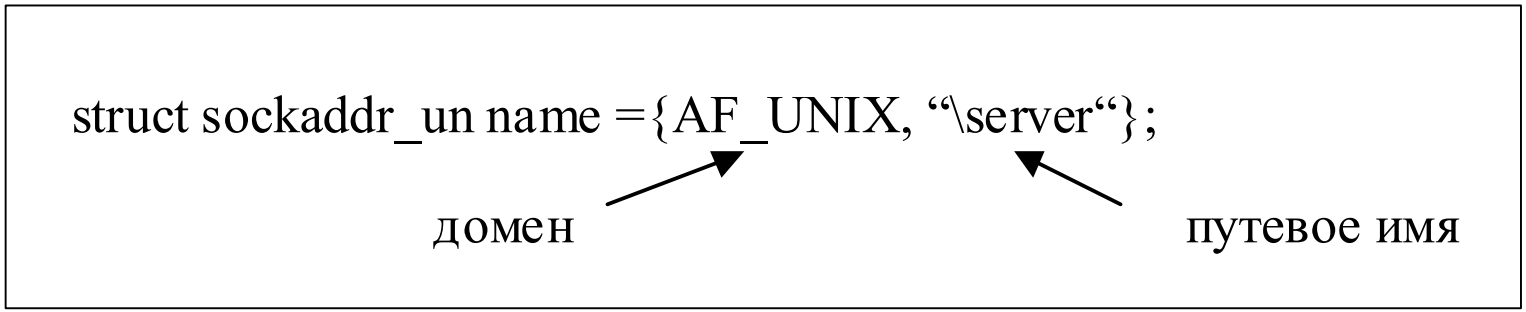
  1 – канал закрывается для отправки информации. Дальнейшие попытки передать данные в этот сокет приведут к выдаче кода неудачного завершения (-1);

  2 – канал закрывается и для чтения, и для записи. Дальнейшие попытки передать данные в этот сокет приведут к выдаче кода неудачного завершения –1, а при продолжении чтения будет возвращаться нулевое значение.
- Функция возвращает 0 в случае успеха и –1 при ошибке.
- Ошибка может возникнуть тогда, когда *sockfd* является неверным дескриптором, или *sockfd* указывает на файл, а не на сокет, или отсутствует соединение с указанным сокетом.

Адрес сокета. Домен UNIX.

Адресат использует указание путевого имени как канал. Адрес сокета содержит указание его типа (AF_UNIX) и путь path

```
struct sockaddr_un
{  short sun_family;              // AF_UNIX
   char sun_path[108];  // UNIX-name
}
```



Адрес сокета. Домен INET

Каждый процесс характеризуется IP-адресом и номером порта. Следовательно, адрес сокета содержит IP-адрес и номер порта.

```
struct sockaddr_in
{   short       sin_family;          // тип домена (AF_INET)
    ushort      sin_port;            // номер порта
    struct in_addr sin_addr;         // IP-адрес
    char        sin_zero[8];         // не используется
}
```

Примеры

- Создаем гнездо:

int sock = socket(AF_UNIX,SOCK_STREAM,0);

- Регистрируем его в системе:

bind(sock, (struct sockaddr)&name, sizeof(name));*

- Настраиваем очередь, подсоединяем socket

listen(sock, 10)

- Если в очереди запросов накопилось более 10-ти сообщений, то произойдет возврат с кодом ошибки.

Отправитель и получатель должны иметь один и тот же тип сокета и относиться к одному домену. Домен обычно используется для связи процессов работающих на нескольких машинах.

Функции чтения/записи

int ***send****(int sock, char *msg, int len, int flags)*

Функция передает содержащееся в аргументе msg сообщение длиной len байтов в сокет, заданный аргументом sock и соединенный с данным сокетом. Аргументу flags обычно присваивается нуль.

- Возвращает число переданных байтов данных.

int ***sendto****(int sock, char *msg, int len, int flags, struct sockaddr *toaddr, int tolen)*

- Аргумент *toaddr* – это указатель на объект, который содержит имя сокета-получателя. *tolen* содержит число байтов в объекте, на который указывает аргумент *toaddr*.
- Возвращает число переданных байтов данных.

int ***recv****(int sock, char *msg, int len, int flags)*

- Принимает сообщение в буфер *msg*.
- Возвращает число байтов данных, записанных в буфер *msg*.

int ***recvfrom****(int sock, char *msg, int len, int flags, struct sockaddr *fromaddr, int *fromlen)*

- fromaddr и len позволяют узнать имя сокета-отправителя.
- Возвращает число принятых байтов данных.

flags:

- MSG_OOB – принять/передать экстренные данные (out-of-band)
- MSG_PEEK – просмотреть данные, не удаляя их из системного буфера (следующая операция чтения получит те же данные).

Некоторые полезные функции

*struct hostent **__gethostbyname__*(char *dname)*

- Трансляция доменного имени хоста в IP-адрес (можно использовать как строку, содержащую IP-адрес ("192.168.0.6"), так и доменное имя. Например: "uiits.miem.edu.ru", и тогда функция обратится к серверу доменных имен для определения адреса).

```
struct      hostent {
    char*   h_name;    /* официальное имя хоста */
    char** h_aliases;   /* список областей - массив альтернативных имен хоста
    (заканчивается нулем)*/
    int     h_addrtype; /* тип адреса хоста */
    int     h_length;   /* длина адреса в байтах */
    char**  h_addr_list; /* список адресов (из сервера имен - name server). Это - массив сетевых
    адресов хоста (заканчивается нулем). Адреса хоста записаны в сетевом порядке
    следования байтов.*/
};
```

Для совместимости обычно на практике используют следующий макрос:

```
#define             h_addr      h_addr_list[0]
```

Здесь h_addr трактуется как первый адрес в h_addr_list.

Например:

```
struct hostent *hp;
hp = gethostbyname("uiits.miem.edu.ru");
bcopy(hp->h_addr, &serv_addr.sin_addr,hp->h_length);
serv_addr.sin_family = hp-> h_addrtype;
```

Функции htons, ntohs

unsigned short int ***htons****(unsigned short int hostshort)*

- Приведение в соответствие порядка следования байтов в структуре данных (для хоста и сети они различны – фактически, происходит примерно следующее: исходное число циклически сдвигается на 4 разряда влево).

Например:

#define PORTNUM 123

serv_addr.sin_port = htons(PORTNUM);

Обратное преобразование осуществляется функцией **ntohs**

unsigned short int ***ntohs****(unsigned short int netshort);*

Прототипы этих функций описаны в <netinet/in.h>. Там же описываются аналогичные функции htonl и ntohl:

unsigned long int ***htonl****(unsigned long int hostlong)*

unsigned long int ***ntohl****(unsigned long int netlong)*

Получение/установка имени хоста

int ***gethostname****(char *name, size_t len)*

int ***sethostname****(const char *name, size_t len)*

Для установки имени хоста надо иметь права суперпользователя.

int ***getsockname****(int s, struct sockaddr *name, int *namelen)*

Получить имя сокета. Функция возвращает текущее имя для указанного параметром *s* сокета. Параметр namelen должен быть проинициализирован (определен) перед вызовом функции. После вызова этот параметр будет содержать реальный размер адреса (в байтах).

Примеры приложений "клиент-сервер"

Общая схема сервера:

```
sockfd = socket(…);
bind(sockfd,…);
listen(sockfd,…);
for(;;)
{       newsockfd = accept(sockfd,…);
        if(fork()==0)
        {
                close(sockfd);
                …
                exit(0);
        }
        close(newsockfd);
        …
}
```

При компиляции в Linux: cc -l socket -l unix program.c