# Системное программное обеспечение локальных компьютерных сетей

## Программный интерфейс взаимодействия сокетов Беркли (проолжение)

Денис Пынькин

2011 – 2012

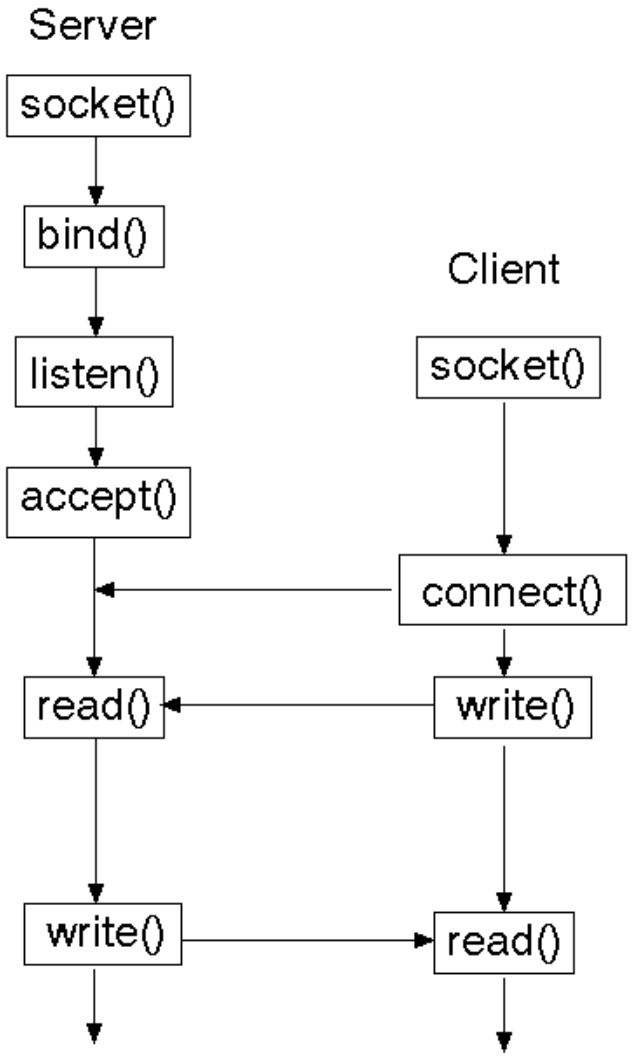
e-mail: denis.pynkin@bsuir.by

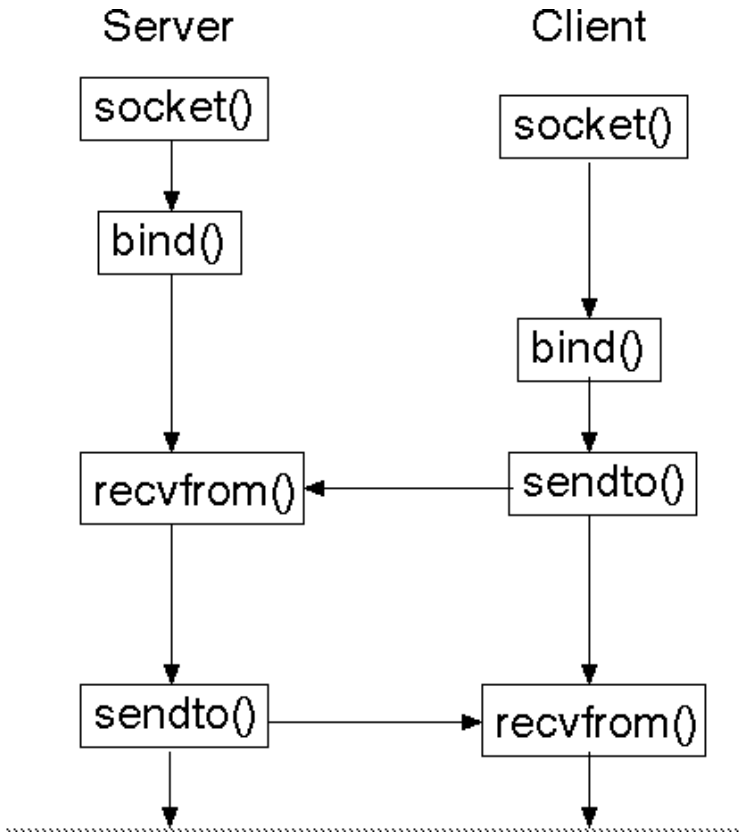
**http://goo.gl/32cTB**

СЧАСТЬЕ ДЛЯ ВСЕХ, ДАРОМ, И ПУСТЬ НИКТО НЕ УЙДЕТ ОБИЖЕННЫЙ!

(c)Стругацкие, Пикник на обочине

# Сетевое взаимодействие



# socket

`socket` – создать оконечную точку коммуникации

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 int socket(int domain, int type, int protocol);
```

Вызов `socket` создает оконечную точку для коммуникации и возвращает её дескриптор.

В случае ошибки возвращается -1; в противном случае возвращается дескриптор, ссылающийся на сокет.

# Домен коммуникации

Параметр `domain` задает “домен” коммуникации; выбирает набор протоколов, которые будут использоваться для коммуникации.
Такие наборы описаны в <sys/socket.h>. Примеры форматов:

- PF_UNIX,PF_LOCAL – Локальная коммуникация
- PF_INET – IPv4, протоколы Интернет
- PF_INET6 – IPv6, протоколы Интернет
- PF_IPX – IPX – протоколы Novell
- PF_NETLINK – Устройство для общения пользователя с ядром
- PF_X25 – Протокол ITU-T X.25 / ISO-8208
- PF_AX25 – Протокол AX.25, любительское радио
- PF_APPLETALK – Appletalk
- PF_PACKET – Низкоуровневый пакетный интерфейс

# Тип сокета

Сокет имеет тип `type`, задающий семантику коммуникации.
Стандарт POSIX определяет следующие типы:

- SOCK_STREAM
- SOCK_SEQPACKET
- SOCK_DGRAM
- SOCK_RAW

# Протокол сокета

Параметр `protocol` задает конкретный протокол, который используется на сокете. Обычно существует только один протокол, обеспечивающий конкретный тип сокета в заданном наборе протоколов. Однако, возможно существование нескольких таких протоколов – тогда и используется этот параметр. Номер протокола зависит от используемого домена коммуникации.

# socketpair

`socketpair` – создать неименованную пару подключенных сокетов (аналог неименованных каналов через сокеты)

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 int socketpair(int domain,  int type, int protocol, int sv[2]);
```

Вызов `socketpair` создает неименованную пару подключенных оконечных точек для коммуникации и возвращает их дескрипторы.

# bind

`bind` – привязать имя к сокету

```
1  #include <sys/types.h>
2  #include <sys/socket.h>
3
4  int bind(int sockfd,  struct sockaddr *my_addr,  socklen_t addrlen);
```

`bind` привязывает к сокету `sockfd` локальный адрес `my_addr` длиной `addrlen`.

Традиционно, эта операция называется “присваивание сокету имени”.

# listen

`listen` – слушать соединения на сокете (перевести сокет в пассивный режим).

```
1 #include <sys/socket.h>
2 int listen(int s, int backlog);
```

В случае успеха возвращается нуль. При ошибке возвращается -1, а `errno` устанавливается должным образом.

# listen

Системный вызов `listen` применим только к сокетам типа `SOCK_STREAM` или `SOCK_SEQPACKET`.
Параметр `backlog` задает максимальную длину, до которой может расти очередь ожидающих соединений. Если приходит запрос на соединение, а очередь полна, то клиент получит ошибку `ECONNREFUSED` или, если соответствующий протокол поддерживает повторную передачу, запрос может быть игнорирован, чтобы попытаться ответить на повторный запрос.

# accept

## accept – прием соединений

```
1  #include <sys/socket.h>
2  int accept (int sd,  struct sockaddr *restrict address,
3          socklen_t *restrict address_len);
```

Этот системный вызов возвращает -1 в случае ошибки. При успешном завершении возвращается неотрицательное целое, являющееся дескриптором сокета.

# connect

`connect` – инициирует соединение на сокете.

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
```

Если соединение или привязка прошла успешно, возвращается нуль. При ошибке возвращается -1, а `errno` устанавливается должным образом.

# connect

Файловый дескриптор `sockfd` должен ссылаться на сокет. Если сокет имеет тип `SOCK_DGRAM`, значит, адрес `serv_addr` является адресом по умолчанию, куда посылаются датаграммы, и единственным адресом, откуда они принимаются. Если сокет имеет тип `SOCK_STREAM` или `SOCK_SEQPACKET`, то данный системный вызов попытается установить соединение с другим сокетом. Другой сокет задан параметром `serv_addr`, являющийся адресом длиной `addrelen` в пространстве коммуникации сокета. Каждое пространство коммуникации интерпретирует параметр `serv_addr` по-своему.

# connect

Обычно сокеты с протоколами, основанными на соединении, могут устанавливать соединение только один раз; сокеты с протоколами без соединения могут использовать `connect` многократно, чтобы изменить адрес назначения. Сокеты без поддержки соединения могут прекратить связь с другим сокетом, установив член `sa_family` структуры sockaddr в `AF_UNSPEC`.

# send

`send, sento, sendmsg` – отправить сообщение в сокет

```
1  #include <sys/types.h>
2  #include <sys/socket.h>
3
4  int send(int s, const void *msg, size_t len, int flags);
5  int sendto(int s, const void *msg, size_t len, int flags,
6  const struct sockaddr *to, socklen_t tolen);
```

`send`, `sendto`, и `sendmsg` используются для пересылки сообщений на другой сокет. `send` можно использовать, только если сокет находится в соединенном состоянии, тогда как `sendto` и `sendmsg` можно использовать в любое время.

Эти системные вызовы возвращают количество отправленных символов или -1, если произошла ошибка.

# recv

`recv`, `recvfrom`, `recvmsg` – получить сообщение из сокета

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3
4 int recv(int s, void *buf, size_t len, int flags);
5 int recvfrom(int s, void *buf, size_t len, int flags,
6 struct sockaddr *from, socklen_t *fromlen);
```

Системные вызовы `recvfrom` и `recvmsg` используются для получения сообщений из сокета, и могут использоваться для получения данных, независимо от того, является ли сокет ориентированным на соединения или нет.

# Флаги

- MSG_OOB – Запрашиваются экстренные данные. Трактовка этого понятия зависит от протокола.
- MSG_DONTROUTE (send) – не использовать маршрутизацию при отправке пакета
- MSG_DONTWAIT (send) – включает неблокирующий режим
- MSG_TRUNC (recv) – Возвращает реальную длину пакета (пакетные протоколы)
- MSG_PEEK (recv) – Не удалять прочитанные данные.
- MSG_WAITALL (recv) – для сокетов типа SOCK_STREAM флаг означает, что вызывающий процесс блокируется до получения всего запрошенного объема данных.

# close

`close` – закрыть файловый дескриптор

```
1 #include <unistd.h>
2
3 int close(int fd);
```

возвращает ноль при успешном завершении или -1, если произошла ошибка.

# shutdown

`shutdown` – закрыть файловый дескриптор

```
1 #
2 #include <sys/socket.h>
3 int shutdown (int sd,  int how);
```

возвращает ноль при успешном завершении или -1, если произошла ошибка.
Значение аргумента how показывает, что именно завершается: `SHUT_RD` прекращает прием, `SHUT_WR` – отправку, `SHUT_RDWR` – и то, и другое.

# select

Функции `select` и `pselect` ждут изменения статуса нескольких файловых дескрипторов

```
1 #include <sys/time.h>
2 #include <sys/types.h>
3 #include <unistd.h>
4
5 int select(int n, fd_set *readfds, fd_set *writefds,
6   fd_set *exceptfds, struct timeval *timeout);
```

# select

`n` на единицу больше самого большого номера дескриптора из всех наборов. `timeout` – это верхняя граница времени, которое

пройдет перед возвратом из select. Можно использовать ноль, при этом select завершится немедленно. (Это полезно для периодического опроса.) Если timeout равен NULL (нет тайм-аута), то select будет ожидать изменений неопределенное время.

# fd_set

Стандарт POSIX-2001 определяет тип `fd_set` как абстрактный.

```
1 FD_SET(int fd, fd_set *set);
2 FD_CLR(int fd, fd_set *set);
3 FD_ISSET(int fd, fd_set *set);
4 FD_ZERO(fd_set *set);
```

# Опции сокетов

С сокетами могут быть ассоциированы опции, влияющие на их функционирование. Значения этих опций можно опросить или изменить с помощью функций `getsockopt` и `setsockopt`.

```
1  #include <sys/socket.h>
2  int getsockopt (int sd,  int level,
3    int option_name,
4    void *restrict option_value,
5    socklen_t *restrict option_len);
6
7  int setsockopt (int sd,  int level,
8    int option_name,
9    const void *option_value,
10   socklen_t option_len);
```

# Уровни опций

`level` – указывает к какому уровню относится опция

- SOL_SOCKET – опции сокета
- SOL_TCP – опции для протокола TCP
- SOL_IP – опции для протокола IP

# SOL_SOCKET

| Имя опции | Тип | Описание |
|---|---|---|
| SO_ERROR | int | Статус ошибок (после опроса очищается) |
| SO_TYPE | int | Тип сокета |
| SO_PROTOCOL | int | Протокол |
| SO_DEBUG | bool | Отладочная информация о работе сокета. |
| SO_ACCEPTCONN | bool | Указывает, является ли сокет слушающим. |
| SO_SNDBUF | int | Размер буфера для передаваемых данных (выходного |
| SO_RCVBUF | int | Размер входного буфера. |
| SO_RCVLOWATM | int | Минимальное число байт, обрабатываемых при вводе. |
| SO_SNDLOWAT | int | Минимальное число байт, обрабатываемых при выводе |
| SO_RCVTIMEO | timeval | Длительность ожидания поступления данных при ввод |
| SO_SNDTIMEO | timeval | Длительность ожидания отправки данных при выводе. |
| SO_TIMESTAMP | bool | Включить передачу отметок времени |
| SO_BROADCAST | bool | Переводит сокет в широковещательный режим передач |
| SO_OOBINLINE | bool | Если установлена, то данные out-of-band помещаются |
| SO_REUSEADDR | bool | Использование "занятого"адреса (для bind) |

# SOL_SOCKET: SO_LINGER

Определяет, блокировать ли процесс при закрытии дескриптора sd до передачи буферизованных данных, и если блокировать, то на какой срок.

```
1 struct linger {
2   int l_onoff;    /* linger active */
3   int l_linger;   /* how many seconds to linger for */
4 };
```

# SOL_TCP

| Имя опции | Тип | Описание |
|---|---|---|
| TCP_NODELAY | bool | Отключает алгоритм Нэгла (Nagle) |
| TCP_MAXSEG | int | устанавливает или сообщает максимальный размер сегмента |
| TCP_CORK | bool | При включении этой опции, перестают отсылаться частичны |

# SOL_IP

| Имя опции | Тип | Описание |
|---|---|---|
| IP_HDRINCL | bool | Включение этого флага означает, что пользователь ух |
| IP_OPTIONS | | Устанавливает или возвращает те опции IP, которые п |
| IP_TTL | byte | Устанавливает или получает текущее значение поля T |
| IP_TOS | byte | Устанавливает или получает значение поля Тип-Серви |
| IP_PMTU_DISCOVER | int | Устанавливает или возвращает значение опции Path M |
| IP_MTU | int | Возвращает используемое в данный момент значение |

# Спасибо за внимание!
# Вопросы?