

# Работа с сокетами

## Часть 1

# Определение

- *Сокет* (socket) - это конечная точка сетевых коммуникаций.
- Сокет идентифицируется дескриптором (int)

# Атрибуты сокета

- Домен
  - локальный сокет (Unix-домен)
  - сетевой сокет (Internet-домен)
- Тип
  - с установкой соединения
  - без установки соединения
  - низкоуровневый
- Протокол
  - TCP
  - UDP
  - низкоуровневое задание протокола

# Создание сокета

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

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

- *int* domain:
  - AF\_UNIX
  - AF\_INET
- *int* type:
  - SOCK\_STREAM
  - SOCK\_DGRAM
  - SOCK\_RAW
- *int* protocol:
  - 0 (by default)

# Адреса

- **Unix-домен:**

текстовая строка – имя файла, через который происходит обмен данными

- **Internet-домен:**

комбинация IP-адреса и номера порта

# Связывание сокета

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

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

```
int bind(int sockfd, struct sockaddr *addr, int addrlen);
```

```
struct sockaddr {  
    unsigned short  sa_family;  
    char           sa_data[14]  
};
```

- *sa\_family* – семейство адресов, AF\_xxx
- *sa\_data* – 14 байтов для хранения адреса

# Задание адреса

```
struct sockaddr_in {  
    short int     sin_family; // Семейство адресов  
    unsigned short int sin_port; // Номер порта  
    struct in_addr sin_addr; // IP-адрес  
    unsigned char  sin_zero[8]; // "Дополнение" до  
    размера структуры sockaddr  
};
```

```
struct in_addr {  
    unsigned long s_addr;  
};
```

# Порядок хранения байтов

- Существуют два порядка хранения байтов в слове и двойном слове:
  - порядок хоста (host byte order)
  - сетевой порядок (network byte order)
- Функции:
  - *htons* (Host TO Network Short)
  - *htonl* (Host TO Network Long)
  - *ntohs* (Network TO Host Short)
  - *ntohl* (Network TO Host Long)



# Установка соединения (сервер)

- Создание сокета
- Привязка в локальному адресу  
(INADDR\_ANY – любой локальный адрес, 0 – любой порт)
- Запустить ожидание запросов
- Обслуживание поступивших запросов

# Режим ожидания запросов

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

- *sockfd* – дескриптор сокета
- *backlog* – размер очереди запросов

# Обслуживание запросов

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

```
int accept(int sockfd, void *addr, int *addrlen);
```

функция создает новый сокет для общения с клиентом и возвращает его дескриптор

- *sockfd* – дескриптор слушающего сокета
- *addr* – адрес сокета клиента
- *addrlen* – размер структуры (длина, которая реально была использована)

# Установка соединения (клиент)

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

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

```
int connect(int sockfd, struct sockaddr *serv_addr, int  
addrlen);
```

- *sockfd* – дескриптор сокета для обмена данными с сервером
- *serv\_addr* – указатель на структуру с адресом сервера
- *addrlen* – длина этой структуры

Функция сама привязывает сокет к адресу.

# Обмен данными

- *send()*
- *recv()*

Можно использовать (но не желательно)

- *read()*
- *write()*

# Отправка данных

```
int send(int sockfd, const void *msg, int len, int flags);
```

- *sockfd*
- *msg* – указатель с на буфер с данными
- *len* – длина буфера в байтах
- *flags* – набор битовых флагов

Возвращает число байтов, которые реально были отправлены (или -1 в случае ошибки).

# Отправка буфера целиком

```
int sendall(int s, char *buf, int len, int flags)
{
    int total = 0;
    int n;

    while(total < len)
    {
        n = send(s, buf+total, len-total, flags);
        if(n == -1) { break; }
        total += n;
    }

    return (n == -1 ? -1 : total);
}
```

# Чтение данных

```
int recv(int sockfd, void *buf, int len, int flags);
```

- Все аналогично send()
- Если соединение разорвано – возвращает 0



# Заккрытие сокета

```
#include <unistd.h>
```

```
int close(int fd);
```

- Можно так же запретить передачу данных в каком-то одном направлении:

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

- *how*:
  - 0 — запретить чтение из сокета
  - 1 — запретить запись в сокет
  - 2 — запретить и то и другое

# Интерфейс внутренней петли

- INADDR\_LOOPBACK

# Задание 1

- Написать две программы (клиент и сервер):

клиент посылает сообщение серверу и  
выводит на экран его ответ

сервер читает все, что передает ему клиент, а  
затем просто отправляет полученные данные  
обратно

# Обмен датаграммами

- Создать сокет
- Привязать адрес
- Отправлять/принимать:

```
int sendto(int sockfd, const void *msg, int len, unsigned  
int flags, const struct sockaddr *to, int tolen);
```

```
int recvfrom(int sockfd, void *buf, int len, unsigned int  
flags, struct sockaddr *from, int *fromlen);
```

Можно сделать и через *connect-send/recv*.

# Задание 2

- Обмен датаграммами:

первая программа отправляет два сообщения

вторая получает их и печатает на экране

попробовать через обычный и  
присоединенный сокет

# Преобразование IP-адреса

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

```
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
```

```
int inet_aton(const char *cp, struct in_addr  
*inp);
```

# Задание 3

- Реализовать чат:

для двух компьютеров

для нескольких компьютеров