Работа с сокетами

Часть 2

Структура сообщения

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:*
  - IPPROTO_TCP
  - IPPROTO_UDP
  - IPPROTO_RAW

Заголовок UDP-сообщения

Порт отправителя (16 бит)	Порт получателя (16 бит)
Длина (заголовок+данные) (16 бит)	Контрольная сумма (16 бит)

```
struct UdpHeader
{
        short int targ_port;      //порт отправителя
        short int length;         //длина сообщения
        int checksum;             //контрольная сумма
}
```

Задание 1

- Написать отправитель UDP-сообщений с использованием низкоуровневых сокетов

Работа с адресами

Преобразование IP-адреса:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int inet_aton(const char *cp, struct in_addr *in_p);
unsigned long int inet_addr(const char *cp);
char *inet_ntoa(struct in_addr in);
```

Работа с DNS

```
#include <netdb.h>

struct hostent *gethostbyname(const char *name);

struct hostent {
    char    *h_name;
    char    **h_aliases;
    int     h_addrtype;
    int     h_length;
    char    **h_addr_list;
};
#define h_addr h_addr_list[0]
```

Переменные структуры

- **h_name**. Имя хоста.
- **h_aliases**. Массив строк, содержащих псевдонимы хоста. Завершается значением NULL.
- **h_addrtype**. Тип адреса. Для Internet-домена - **AF_INET**.
- **h_length**. Длина адреса в байтах.
- **h_addr_list**. Массив, содержащий адреса всех сетевых интерфейсов хоста. Завершается нулём. Байты каждого адреса хранятся в сетевом порядке, поэтому **htonl** вызывать не нужно.

Еще полезные функции

- Узнать имя локального хоста

```
#include <unistd.h>

int gethostname(char *hostname, size_t size);
```

- Адрес сокета на другом конце соединения

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

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

Параллельное обслуживание клиентов

- Способ 1: использование нового процесса для обработки каждого соединения

- Задание 2: реализовать tcp-сервер с параллельной обработкой

Параллельное обслуживание клиентов

- Способ 2: Неблокирующие сокеты

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

sockfd = socket(AF_INET, SOCK_STREAM, 0);
fcntl(sockfd, F_SETFL, O_NONBLOCK);
```

Функция select

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

int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);

FD_CLR(int fd, fd_set *set);        //удаляет дескриптор fd из множества set
FD_ISSET(int fd, fd_set *set);      //проверяет, содержится ли дескриптор fd в множестве set
FD_SET(int fd, fd_set *set);        //добавляет дескриптор fd в множество set
FD_ZERO(int fd);                    // очищает множество set

struct timeval {
   int tv_sec;     // секунды
   int tv_usec;    // микросекунды
};
```