

Síťové sockety

TCP – před samotnou komunikací se naváže spojení, Všechna odeslaná data se potvrzují a na konec je nutné spojení ukončit (uzavřít). TCP paket obsahuje svou hlavičku a samotná data, která přenáší. TCP paket bude vložen do IP datagramu (jako data IP datagramu) a odeslán.

UDP – jedná se tzv. nespojovanou službu. To znamená, že nedochází k navázání spojení. Prostě odešleme data na stanovenou IP adresu a daný UDP port a nevíme, zda data dorazila a zda se nepoškodila nebo nedorazila v jiném spojení.

1.1. Funkce pro práci se sockety

- Definice formátů adres (<sys/socket.h>):

- AF_UNIX – interní protokol UNIXu,
- AF_INET – Internet. adresy,
- AF_ISO – protokoly ISO.

- Typ socketu:

- SOCK_STREAM – (TCP) sekvenční, spolehlivá, dvoustranná proudová komunikace, mechanismus přenosu *out-of-band*,
- SOCK_DGRAM – (UDP) podpora datagramů (nespojované, nespolehlivé zprávy malé délky).

- Specifikace jména socketu:

```
struct sockaddr {
    /* Rodina adres AF_xxx */
    unsigned short sa_family;
    /* až 14bytová přímá adresa */
    char sa_data[14]; };
```

- Volba adresy a portu:

```
struct sockaddr_in {
    /* Rodina adres AF_xxx */
    short int sin_family;
    /* Číslo portu */
    unsigned short int sin_port;
    /* Internetová adresa */
    struct in_addr sin_addr;
    /* nepoužito */
    unsigned char sin_zero[8]; };
```

- Nastavení adresy:

```
struct in_addr {
    /* 32 bitová adresa */
    unsigned long s_addr; };
```

- Pro převod adresy z textového tvaru (192.168.0.1) do binárního tvaru:

```
int inet_aton(const char *NAME,
             struct in_addr *ADDR)
```

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

- převod pořadí bytu mezi sítí (big endian, MSB první) a hostem, např. na x86 je LSB byte první v pořadí (little endian).

- Host → Network:
uint32_t htonl(uint32_t hostlong),
uint16_t htons(uint16_t hostshort).
- Network → Host:
uint32_t ntohl((uint32_t netlong),
uint16_t ntohs((uint16_t hostshort)).

- **socket()** – tvoří socket daného typu a vrátí file-descriptor, protokol 0x0, domain AF_xxx

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

- **connect()** – slouží k navazování spojení

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

- **bind()** – sváže socket se jménem, v kombinaci s listen() určí čísla portu, kde server poslouchá

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

- **listen()** – zavolat po bind(), slouží ke konfiguraci pro příjmem dat., backlog udává max. počet nepokrytých spojení ve frontě.

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

- **accept()** – čeká na příchozí spojení, vrátí parametry prvního spojení ve frontě a nový socket, který slouží ke komunikaci s druhou stranou. Původní socket sockfd se tak může opět použít pro čekání na další spojení.

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

- **close(), shutdown()** – zavírá socket.

• Posílání zpráv do socketu

- **send()** – pouze jeli socket spojený, neobsahuje indikaci správného doručení, s flagem 0 je ekv. write().

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

- **sendto()** – pouze pro SOCK_STREAM

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

Definice struktury msghdr:

```
struct msghdr {
    /* optional address */
    void * msg_name;
    /* size of address */
    socklen_t msg_namelen;
    /* scatter/gather array */
    struct iovec * msg_iov;
    /* # elements in msg_iov */
    size_t msg_iovlen;
    /* ancillary data, see below */
    void * msg_control;
    /* ancillary data buffer len */
    socklen_t msg_controllen;
    /* flags on received message */
    int msg_flags;
};
```

- **sendmsg()** –

```
int sendmsg(int s, const struct msghdr *msg,
            unsigned int flags);
```

- **write()** – systémová funkce <unistd.h>

```
ssize_t write(int fd, const void *buf,
              size_t count);
```

- Hodnoty parametry flags:

- * MSG_OOB – zpracuj out-of-band data
- * MSG_DONTROUTE – obejdi routing, použij přímé rozhraní

• Příjem zprávy ze socketu

- **recv()** – čtení dat bez odstranění z fronty

```
ssize_t recv(int s, void *buf,
             size_t len, int flags);
```

- **recvfrom()** –

```
ssize_t recvfrom(int s, void *buf,
                 size_t len, int flags, struct
                 sockaddr *from,
                 socklen_t *fromlen);
```

- **recvmsg()** –

```
ssize_t recvmsg(int s, struct msghdr *msg,
                int flags);
```

- **read()** – systémová funkce <unistd.h>, čte data bez odstranění z fronty

```
ssize_t read(int fd, void *buf, size_t count);
```

- Chybové návraty

- EBADF – špatný descriptor,
- ECONNREFUSED – odmítnutí spojení,
- ENOTCONN – nepřipojený socket,
- ENOTSOCK – argument není socket,
- EAGAIN – čas vypršel před doručením,
- EINTR – příjem přerušeno,
- EFAULT – RX ukazatel mimo adresový prostor,
- EINVAL – vložen špatný argument,
- ENOMEM – nelze alokovat paměť pro recvmsg.

1.2. Příklad doménových jmen

- Definice funkcí <netdb.h>

- Struktura hostent pro char. vzdáleného počítače

```
struct hostent {
    /* Řetězec s názvem počítače */
    char *h_name;
    /* Ukazatel na pole aliasu, ukončený NULL */
    char **h_aliases;
    /* Typ adresy, např. AF_INET */
    int h_addrtype;
    /* Délka adresy v bytech */
    int h_length;
    /* Ukazatel na pole ukazatelů ukončené NULL,
    ukazatel obsahuje adresu pole typu char
    délky h_length, obsahující adresu v síťovém
    pořadí */
    char **h_addr_list
};
```

- **gethostbyname()** – vrátí informace o zadaném doménovém jméně.

```
struct hostent *gethostbyname(const char *name);
```

- **gethostbyaddr()** – vrátí info o zadané adrese

```
struct hostent *gethostbyaddr(const char *addr,
                              int len, int type);
```

- Struktura protent

```
struct protoent {
    char *p_name; /* official protocol name */
    char **p_aliases; /* alias list */
    int p_proto; /* protocol number */
};
```

- **getprotobyname(), getprotobynumber()** – vrátí protent pro daný řádek z /etc/protocols pro daný protokol nebo jeho číslo.

```
struct protoent *getprotobyname(const char *name);
struct protoent *getprotobynumber(int proto);
```

- Struktura servent

```
struct servent {
    char *s_name; /* official service name */
    char **s_aliases; /* alias list */
    int s_port; /* port number */
    char *s_proto; /* protocol to use */
};
```

- **getservbyname(), getservbyport()** – vrací strukturu servent pro konkrétní službu nebo číslo z /etc/services

```
struct servent *getservbyname(const char *name,
                              const char *proto);
struct servent *getservbyport(int port,
                              const char *proto);
```

1.3. Kompilace

- Doporučené hlavičkové soubory: <stdio.h> <stdlib.h>, <unistd.h>, <netinet/in.h>, <netdb.h>, <sys/socket.h>, <string.h>

- Kompilace na GNU/Linux:

```
gcc -Wall -o program zdrojak.c
```

- Kompilace na Solaris vyžaduje doplnění knihoven:

```
gcc -Wall -lnsl -lsocket -o program zdrojak.c
```

- Příklad Makefile

```
all: program
```

```
program.o: zdrojak.c
    gcc -c -o program.o zdrojak.c
```

```
program: program.o
    gcc -Wall -o program program.o
```