

Vistula, IT Faculty, 2016

# **Operating Systems & Systems Programming**

Dmitry A. Zaitsev

<http://daze.ho.ua>

## **Lecture 6:**

**Work with Sockets of OS Linux.  
Communication in TCP/IP networks  
using sockets**

# Socket Concepts

- Communication style
- Units of data transmission: packets or stream
- Namespace – address family
- Protocol – protocol family

# Creating a socket

- `#include <sys/socket.h>`
- `int socket(int domain, int type, int protocol);`
- The *domain* specifies a communication domain and selects the protocol family which will be used for communication: `AF_UNIX`, `AF_LOCAL`, `AF_INET`, `AF_APPLETALK`, `AF_IPX`
- *Type* specifies the communication semantics: `SOCK_STREAM`, `SOCK_DGRAM`, `SOCK_RAW`
- *Protocol* specifies a particular protocol to be used with the socket

# Bind a name to a socket

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

- The socket address structure

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

# Socket Addresses

- Local Namespace - socket addresses are file names
- The Internet Namespace:

```
struct sockaddr_in {  
    sa_family_t    sin_family; /* address family: AF_INET */  
    in_port_t      sin_port;   /* port in network byte order */  
    struct in_addr sin_addr;    /* internet address */  
};
```

/\* Internet address. \*/

```
struct in_addr {  
    uint32_t      s_addr; /* address in network byte order */  
};
```

- Examples

```
INADDR_LOOPBACK (127.0.0.1); INADDR_BROADCAST  
(255.255.255.255)
```

# Socket address conversion

- Get socket name

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

- Conversion of addresses

```
int inet_aton (const char *name, struct in_addr  
*addr);
```

```
uint32_t inet_addr (const char *name);
```

```
char * inet_ntoa (struct in_addr addr);
```

# Host Names

- `struct hostent * gethostbyname (const char *name);`
- `struct hostent {  
char *h_name;  
char **h_aliases;  
int h_addrtype;  
int h_length;  
char **h_addr_list;  
char *h_addr; }`

# Ports & Services

- `struct servent * getservbyname (const char *name, const char *proto);`
- `struct servent {  
char *s_name;  
char **s_aliases;  
int s_port;  
char *s_proto;  
}`



```
make_socket (uint16_t port)
{
    int sock;
    struct sockaddr_in name;
    /* Create the socket. */
    sock = socket (PF_INET, SOCK_STREAM, 0);
    if (sock < 0)
    {
        perror ("socket");
        exit (EXIT_FAILURE);
    }
    /* Give the socket a name. */
    name.sin_family = AF_INET;
    name.sin_port = htons (port);
    name.sin_addr.s_addr = htonl (INADDR_ANY);
    if (bind (sock, (struct sockaddr *) &name, sizeof (name)) < 0)
    {
        perror ("bind");
        exit (EXIT_FAILURE);
    }
    return sock;
}
```

## Make socket example

## Fill-in socket address on host name example

```
void init_sockaddr (struct sockaddr_in *name,  
    const char *hostname, uint16_t port)  
{  
    struct hostent *hostinfo;  
    name->sin_family = AF_INET;  
    name->sin_port = htons (port);  
    hostinfo = gethostbyname (hostname);  
    if (hostinfo == NULL)  
    {  
        fprintf (stderr, "Unknown host %s.\n", hostname);  
        exit (EXIT_FAILURE);  
    }  
    name->sin_addr = *(struct in_addr *) hostinfo->h_addr;  
}
```

# Additional functions

- Closing a Socket

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

- Socket Pairs

```
int socketpair (int namespace, int style, int  
protocol, int fildes[2]);
```

# Making a Connection

- `int connect (int socket, struct sockaddr *addr, socklen_t length);`

# Transferring Data

- Sending Data

```
ssize_t send (int socket, const void *buffer, size_t size, int flags);
```

- Receiving Data

```
ssize_t recv (int socket, void *buffer, size_t size, int flags);
```

- Flags:

# Datagram Socket Operations

- Sending Datagrams

```
ssize_t sendto (int socket, const void *buffer,  
size_t size, int flags, struct sockaddr *addr,  
socklen_t length);
```

- Receiving Datagrams

```
ssize_t recvfrom (int socket, void *buffer, size_t  
size, int flags, struct sockaddr *addr, socklen_t  
*length_ptr);
```

# Connections on Server side

- Listening for Connections

```
int listen (int socket, int n);
```

- Accepting Connections

```
int accept (int socket, struct sockaddr *addr,  
socklen_t *length_ptr);
```

- Who is Connected

```
int getpeername (int socket, struct sockaddr  
*addr, socklen_t *length_ptr);
```

# Examples

- .c files