

=====

第八课 网络通信

=====

一、基本概念

1. ISO/OSI七层网络协议模型

~~~~~

|       |              |    |
|-------|--------------|----|
| 应用层   | Application  | ^  |
| 表示层   | Presentation |    |
| 会话层   | Session      | v  |
| 传输层   | Transport    | ^  |
| 网络层   | Network      |    |
| 数据链路层 | Data Link    | 低层 |
| 物理层   | Physical     | v  |

2. TCP/IP协议族

~~~~~

- 1) TCP (Transmission Control Protocol, 传输控制协议)
- 面向连接的服务。
- 2) UDP (User Datagram Protocol, 用户数据报文协议)
- 面向无连接的服务。
- 3) IP (Internet Protocol, 互联网协议)
- 信息传递机制。

图示: tcpip.bmp

3. TCP/IP协议与ISO/OSI模型的对比

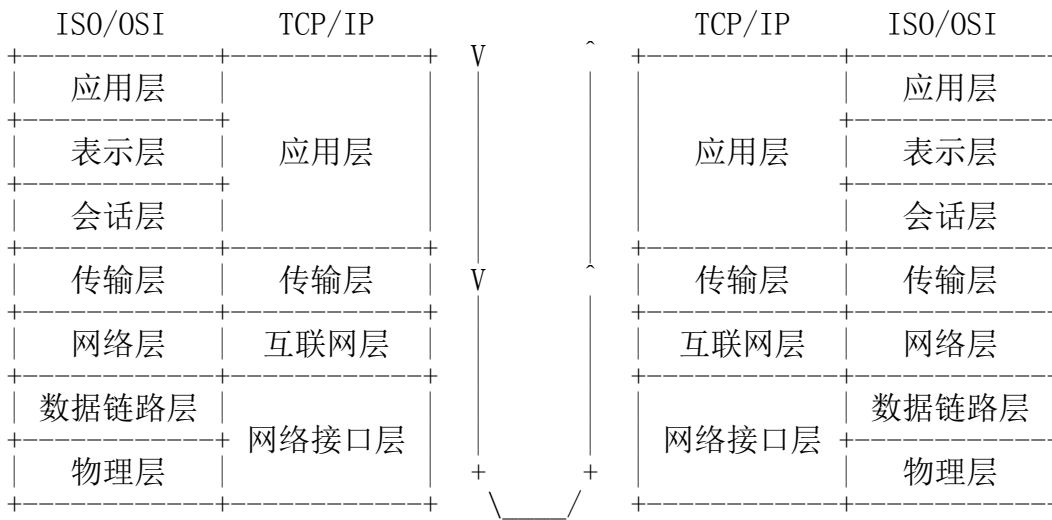
~~~~~

| ISO/OSI | TCP/IP |                 |
|---------|--------|-----------------|
| 应用层     | 应用层    | TELNET/FTP/HTTP |
| 表示层     |        |                 |
| 会话层     |        |                 |
| 传输层     | 传输层    | TCP/UDP         |

|       |       |       |
|-------|-------|-------|
| 网络层   | 互联网层  | IP/路由 |
| 数据链路层 | 网络接口层 | 驱动/设备 |
| 物理层   |       |       |

图示: osi.bmp

#### 4. 消息流



#### 5. 消息包

|                 |
|-----------------|
| TELNET/FTP/HTTP |
| TCP/UDP         |
| IP              |
| ETHERNET        |

从上至下, 消息包逐层递增, 从下至上, 消息包逐层递减。

#### 6. IP地址

- 1) IP地址是Internet中唯一的地址标识
  - A. 一个IP地址占32位, 正在扩充至128位。
  - B. 每个Internet包必须带IP地址。

#### 2) 点分十进制表示法

0x01020304 -> 1. 2. 3. 4, 高数位在左, 低数位在右。

### 3) IP地址分级

A级: 0 + 7位网络地址 + 24位本地地址  
B级: 10 + 14位网络地址 + 16位本地地址  
C级: 110 + 21位网络地址 + 8位本地地址  
D级: 1110 + 28位多播 (Multicast) 地址

### 4) 子网掩码

IP地址 & 子网掩码 = 网络地址

IP地址: 192.168.182.48  
子网掩码: 255.255.255.0  
网络地址: 192.168.182  
本地地址: 48

## 二、套接字(Socket)

### 1. 接口

```
PutTY -> telnet \
LeapFTP -> ftp -> socket -> TCP/UDP -> IP -> 网卡驱动 -> 网卡硬件
IE -> http /
应用程序 -----+
```

图示: bsd.bmp

### 2. 异构

Java @ UNIX -> socket <----> socket <- C/C++ @ Windows

### 3. 模式

- 1) 点对点 (Peer-to-Peer, P2P): 一对一的通信。
- 2) 客户机/服务器 (Client/Server, C/S): 一对多的通信。

### 4. 绑定

先要有一个套接字描述符, 还要有物理通信载体,  
然后将二者绑定在一起。

### 5. 函数

#### 1) 创建套接字

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

unix\_c\_08.txt

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

domain - 域/地址族, 取值:

AF\_UNIX/AF\_LOCAL/AF\_FILE: 本地通信(进程间通信);  
AF\_INET: 基于TCP/IPv4(32位IP地址)的网络通信;  
AF\_INET6: 基于TCP/IPv6(128位IP地址)的网络通信;  
AF\_PACKET: 基于底层包接口的网络通信。

type - 通信协议, 取值:

SOCK\_STREAM: 数据流协议, 即TCP协议;  
SOCK\_DGRAM: 数据报协议, 即UDP协议。

protocol - 特别通信协议, 一般不用, 置0即可。

成功返回套接字描述符, 失败返回-1。

套接字描述符类似于文件描述符, UNIX把网络当文件看待, 发送数据即写文件, 接收数据即读文件, 一切皆文件。

## 2) 准备通信地址

### A. 基本地址类型

```
struct sockaddr {  
    sa_family_t sa_family; // 地址族  
    char        sa_data[14]; // 地址值  
};
```

### B. 本地地址类型

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

```
struct sockaddr_un {  
    sa_family_t sun_family; // 地址族  
    char        sun_path[]; // 套接字文件路径  
};
```

### C. 网络地址类型

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

```
struct sockaddr_in {  
    // 地址族  
    sa_family_t sin_family;  
  
    // 端口号  
    // unsigned short, 0-65535  
    // 逻辑上表示一个参与通信的进程  
    // 使用时需要转成网络字节序  
    // 0-1024端口一般被系统占用  
    // 如: 21-FTP、23-Telnet、80-WWW  
    in_port_t sin_port;  
  
    // IP地址
```

```
    struct in_addr sin_addr;
};
```

```
struct in_addr {
    in_addr_t s_addr;
};
```

```
typedef uint32_t in_addr_t;
```

IP地址用于定位主机，端口号用于定位主机上的进程。

### 3) 将套接字和通信地址绑定在一起

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

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

成功返回0，失败返回-1。

### 4) 建立连接

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

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

成功返回0，失败返回-1。

### 5) 用读写文件的方式通信：read/write

### 6) 关闭套接字：close

### 7) 字节序转换

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

```
// 32位无符号整数，主机字节序 -> 网络字节序
uint32_t htonl (uint32_t hostlong);
```

```
// 16位无符号整数，主机字节序 -> 网络字节序
uint16_t htons (uint16_t hostshort);
```

```
// 32位无符号整数，网络字节序 -> 主机字节序
uint32_t ntohl (uint32_t netlong);
```

```
// 16位无符号整数，网络字节序 -> 主机字节序
uint16_t ntohs (uint16_t netshort);
```

主机字节序因处理器架构而异，有的采用小端字节序，有的采用大端字节序。网络字节序则固定采用大端字节序。

### 8) IP地址转换

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

```

                                unix_c_08.txt
// 点分十进制字符串 -> 网络字节序32位无符号整数
in_addr_t inet_addr (const char* cp);

// 点分十进制字符串 -> 网络字节序32位无符号整数
int inet_aton (const char* cp, struct in_addr* inp);

// 网络字节序32位无符号整数 -> 点分十进制字符串
char* inet_ntoa (struct in_addr in);

```

## 6. 编程

### 1) 本地通信

服务器：创建套接字(AF\_LOCAL)->准备地址(sockaddr\_un)并绑定->接收数据->关闭套接字  
 客户机：创建套接字(AF\_LOCAL)->准备地址(sockaddr\_un)并连接->发送数据->关闭套接字

范例：locsvr.c、loccli.c

### 2) 网络通信

服务器：创建套接字(AF\_INET)->准备地址(sockaddr\_in)并绑定->接收数据->关闭套接字  
 客户机：创建套接字(AF\_INET)->准备地址(sockaddr\_in)并连接->发送数据->关闭套接字

范例：netsvr.c、netcli.c

## 三、基于TCP协议的客户机/服务器模型

### 1. 基本特征

- 1) 面向连接。
- 2) 可靠，保证数据的完整性和有序性。

ABCDEF

```

A ->          -
B ->          |
C ->          +- 时间窗口
D ->          |
E -> <- A OK  -
F -> <- B OK
      <- C OK
      <- D OK
      <- E OK
      <- F OK

```

每个发送都有应答，若在时间窗口内没有收到A的应答，则从A开始重发。

### 2. 编程模型

| unix_c_08.txt |       |        |         |       |    |
|---------------|-------|--------|---------|-------|----|
| 步骤            | 服务器   |        | 客户机     |       | 步骤 |
| 1             | 创建套接字 | socket | socket  | 创建套接字 | 1  |
| 2             | 准备地址  | ...    | ...     | 准备地址  | 2  |
| 3             | 绑定套接字 | bind   |         | ----- |    |
| 4             | 监听套接字 | listen |         | ----- |    |
| 5             | 接受连接  | accept | connect | 建立链接  | 3  |
| 6             | 接收请求  | recv   | send    | 发送请求  | 4  |
| 7             | 发送响应  | send   | recv    | 接收响应  | 5  |
| 8             | 关闭套接字 | close  | close   | 关闭套接字 | 6  |

图示: handshake.bmp、tcpcs.bmp

### 3. 常用函数

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

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

将sockfd参数所标识的套接字标记为被动模式，使之可用于接受连接请求。

backlog参数表示未决连接请求队列的最大长度，即最多允许同时有多少个未决连接请求存在。若服务器端的未决连接数已达此限，则客户机端的connect()函数将返回-1，且errno为ECONNREFUSED。

成功返回0，失败返回-1。

图示: listen.bmp

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

从sockfd参数所标识套接字的未决连接请求队列中，提取第一个连接请求，同时创建一个新的套接字，用于在该连接中通信，返回该套接字的描述符。

addr和addrlen参数用于输出连接请求发起者的地址信息。

成功返回通信套接字描述符，失败返回-1。

图示: accept.bmp

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

通过sockfd参数所标识的套接字，期望接收len个字节到buf所指向的缓冲区中。

成功返回实际接收到的字节数，失败返回-1。

```
ssize_t send (int sockfd, const void* buf,
```

```
size_t len, int flags);
```

通过sockfd参数所标识的套接字，  
从buf所指向的缓冲区中发送len个字节。

成功返回实际被发送的字节数，失败返回-1。

图示：concurrent.bmp

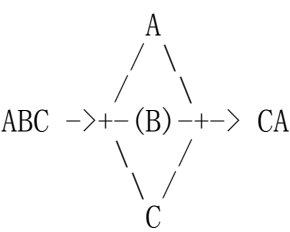
范例：tcpsvr.c、tcpcli.c

图示：inetd.bmp

四、基于UDP协议的客户机/服务器模型

1. 基本特征

- 1) 无连接。
- 2) 不可靠，不保证数据的完整性和有序性。



效率高速度快。

2. 编程模型

| 步骤 | 服务器   |          | 客户机      |       | 步骤 |
|----|-------|----------|----------|-------|----|
| 1  | 创建套接字 | socket   | socket   | 创建套接字 | 1  |
| 2  | 准备地址  | ...      | ...      | 准备地址  | 2  |
| 3  | 绑定套接字 | bind     |          |       |    |
| 4  | 接收请求  | recvfrom | sendto   | 发送请求  | 3  |
| 5  | 发送响应  | sendto   | recvfrom | 接收响应  | 4  |
| 6  | 关闭套接字 | close    | close    | 关闭套接字 | 5  |

图示：udpcs.bmp

3. 常用函数

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

```
ssize_t recvfrom (int sockfd, void* buf, size_t len,  
int flags, struct sockaddr* src_addr,
```



```
socklen_t* addrlen);
```

通过sockfd参数所标识的套接字，  
期望接收len个字节到buf所指向的缓冲区中。

若src\_addr和addrlen参数不是空指针，  
则通过这两个参数输出源地址结构及其长度。  
注意在这种情况下，  
addrlen参数的目标应被初始化为，  
src\_addr参数的目标数据结构的大小。

成功返回实际接收到的字节数，失败返回-1。

```
ssize_t sendto (int sockfd, const void* buf,  
                size_t len, int flags,  
                const struct sockaddr* dest_addr,  
                socklen_t addrlen);
```

通过sockfd参数所标识的套接字，  
从buf所指向的缓冲区中发送len个字节。

发送目的的地址结构及其长度，  
通过dest\_addr和addrlen参数输入。

成功返回实际被发送的字节数，失败返回-1。

范例：udpsvr.c、udpcli.c

图示：tcp\_udp.bmp

127.0.0.1：回绕地址，表示本机，不依赖网络。

练习：基于TCP协议的网络银行。

代码：bank/