0212socket

预备知识

两个机子在通信
不是真正机子在通信
而是两个机子上层的软件的通信

本质就是：客户端进程和服务器进程进行通信

结论：真正的网络通信过程，本质其实是进程间通信！
所以本质是：将数据在主机之间进行转发仅仅是手段，及其收到之后，需要将数据交付给指定的进程！

客户端机子有这么多进程，那客户端机子收到数据之后，怎么知道应该交给哪一个进程呢？

所以这里就涉及到了**端口号！**

端口号是标识特定主机上**进程**的唯一性

所以一个IP+端口号，是不是就是全网唯一的一个进程呢

任何一个发出的报文，里面肯定有ip+port

端口号(port)是传输层协议的内容.

- 端口号是一个2字节16位的整数;
- 端口号用来标识一个进程, 告诉操作系统, 当前的这个数据要交给哪一个进程来处理;
- IP地址 + 端口号能够标识网络上的某一台主机的某一个进程;
- 一个端口号只能被一个进程占用.

反过来可以

{SRC_IP, SRC_PORT}

{DST_IP, DST_PORT}

它们两个都叫做套接字

认识TCP协议

此处我们先对TCP(Transmission Control Protocol 传输控制协议)有一个直观的认识; 后面我们再详细讨论TCP的一些细节问题.

> - 传输层协议
> - 有连接
> - 可靠传输
> - 面向字节流

认识UDP协议

此处我们也是对UDP(User Datagram Protocol 用户数据报协议)有一个直观的认识; 后面再详细讨论.

> - 传输层协议
> - 无连接
> - 不可靠传输
> - 面向数据报

网络规定，所有网络数据都必须是大端！

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

uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

socket编程接口

socket 常见API

```
// 创建 socket 文件描述符 (TCP/UDP, 客户端 + 服务器)
int socket(int domain, int type, int protocol);

// 绑定端口号 (TCP/UDP, 服务器)
int bind(int socket, const struct sockaddr *address,
         socklen_t address_len);

// 开始监听socket (TCP, 服务器)
int listen(int socket, int backlog);

// 接收请求 (TCP, 服务器)
int accept(int socket, struct sockaddr* address,
         socklen_t* address_len);

// 建立连接 (TCP, 客户端)
int connect(int sockfd, const struct sockaddr *addr,
         socklen_t addrlen);
```

sockaddr结构

socket API是一层抽象的网络编程接口,适用于各种底层网络协议,如IPv4、IPv6,以及后面要讲的UNIX Domain Socket. 然而, 各种网络协议的地址格式并不相同.

常见的套接字：
1. 域间socket
2. 原始套接字
3. **网络套接字**

理论上，是三种应用场景
对应的应该是三套接口。

但是！Linux不想多设计太多的借口

所以所有的接口都是被统一了