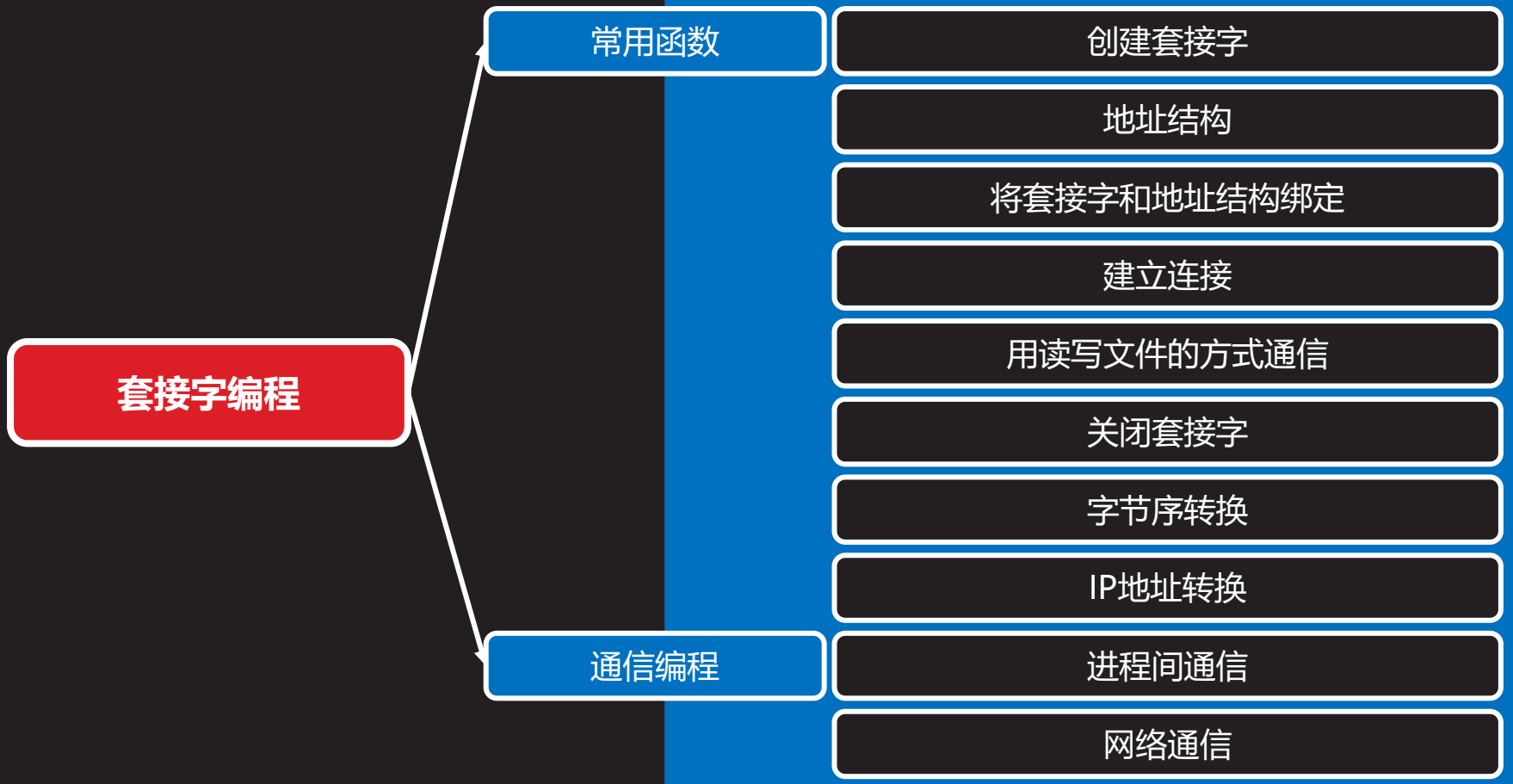
Technology Tarena 达内科技

Unix系统高级编程

套接字编程 Unit24

套接字编程


套接字编程
常用函数
创建套接字
地址结构
将套接字和地址结构绑定
建立连接
用读写文件的方式通信
关闭套接字
字节序转换
IP地址转换
通信编程
进程间通信
网络通信

Technology Tarena 达内科技

常用函数

创建套接字

- 创建套接字

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

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

成功返回套接字描述符，失败返回-1

- *domain*：通信域，即协议族，可取以下值

AF_LOCAL/AF_UNIX	- 本地通信，即进程间通信
AF_INET	- 基于IPv4的网络通信
AF_INET6	- 基于IPv6的网络通信
AF_PACKET	- 基于底层包接口的网络通信

知识讲解

创建套接字（续1）

- 创建套接字
  - ***type***：套接字类型，可取以下值
    SOCK_STREAM - 流式套接字，即使用TCP协议的套接字
    SOCK_DGRAM - 数据报套接字，即使用UDP协议的套接字
    SOCK_RAW - 原始套接字，即使用IP协议的套接字
  - ***protocol***：特殊协议，通常不用，取0即可
- socket函数所返回的套接字描述符类似于文件描述符，Unix系统把网络也看成是文件，发送数据即写文件，接收数据即读文件，一切皆文件

知识讲解

创建套接字（续2）

- 例如
  - int sockfd = socket (AF_LOCAL, SOCK_DGRAM, 0);
    if (sockfd == -1) {
        perror ("socket"); exit (EXIT_FAILURE);
    }
  - int sockfd = socket (AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        perror ("socket"); exit (EXIT_FAILURE);
    }
  - int sockfd = socket (AF_INET, SOCK_DGRAM, 0);
    if (sockfd == -1) {
        perror ("socket"); exit (EXIT_FAILURE);
    }

知识讲解

地址结构

- 套接字接口库通过地址结构定位一个通信主体，可以是一个文件，可以是一台远程主机，也可以是执行者自己
- 基本地址结构，本身没有实际意义，仅用于泛型化参数
  -
```
struct sockaddr {
    sa_family_t sa_family;   // 地址族
    char        sa_data[14]; // 地址值
};
```
- 本地地址结构，用于AF_LOCAL/AF_UNIX域的本地通信
  -
```
#include <sys/un.h>
struct sockaddr_un {
    sa_family_t sun_family; // 地址族(AF_LOCAL)
    char        sun_path[]; // 套接字文件路径
};
```

知识讲解

地址结构（续1）

- 网络地址结构，用于AF_INET域的IPv4网络通信
  -
```
#include <netinet/in.h>
struct sockaddr_in {
    sa_family_t    sin_family; // 地址族(AF_INET)
    in_port_t      sin_port;   // 端口号
    struct in_addr sin_addr;   // IP地址
};
```
  -
```
struct in_addr {
    in_addr_t s_addr;
};
```
  -
```
typedef uint16_t in_port_t; // 无符号短整型
typedef uint32_t in_addr_t; // 无符号长整型
```

知识讲解

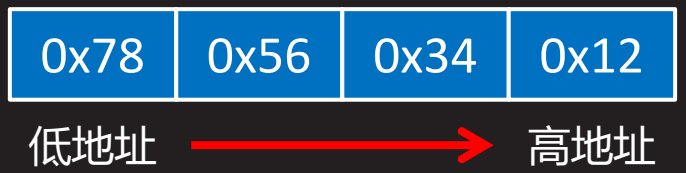
地址结构（续2）

知识讲解

- 如前所述，通过IP地址可以定位网络上的一台主机，但一台主机上可能同时有多个网络应用在运行，究竟想跟哪个网络应用通信呢？这就需要靠所谓的端口号来区分，因为不同的网络应用会使用不同的端口号。用IP地址定位主机，再用端口号定位运行在这台主机上一个具体的网络应用，这样一种对通信主体的描述才是唯一确定的
- 套接字接口库中的端口号被定义为一个16位的无符号整数，其值介于0到65535，其中0到1024已被系统和一些网络服务占据，比如21端口用于ftp服务、23端口用于telnet服务、80端口用于www服务等，因此一般应用程序最好选择1024以上的端口号，以避免和这些服务冲突

地址结构（续3）

- 网络应用与单机应用不同，经常需要在具有不同硬件架构和操作系统的计算机之间交换数据，因此编程语言里一些多字节数据类型的字节序问题就需要特别予以关注
  - 假设一台小端机器里有一个32位整数：0x12345678，它在内存中按照小端字节序低位低地址的规则存放：



  - 现在，这个整数通过网络被传送到一台大端机器里，内存中的形态不会有丝毫差别，但在大端机器看来地址越低的字节数位应该越高，因此它会把这4个字节解读为：0x78563412，而这显然有悖于发送端的初衷

知识讲解

地址结构（续4）

- 为了避免字节序带来的麻烦，套接字接口库规定凡是在网络中交换的多字节整数(short、int、long、long long和它们的unsigned版本)一律采用网络字节序传输。而所谓网络字节序，其实就是大端字节序。也就是说，发数据时，先从主机字节序转成网络字节序，然后发送；收数据时，先从网络字节序转成主机字节序，然后使用
  - 小端机A，0x12345678，主机序[0x78,0x56,0x34,0x12]转成网络序[0x12,0x34,0x56,0x78]，发送给B和C
  - 大端机B，接收网络序[0x12,0x34,0x56,0x78]，转成主机序[0x12,0x34,0x56,0x78]，0x12345678
  - 小端机C，接收网络序[0x12,0x34,0x56,0x78]，转成主机序[0x78,0x56,0x34,0x12]，0x12345678

知识讲解

地址结构（续5）

- 网络地址结构sockaddr_in中表示端口号的sin_port成员和表示IP地址的sin_addr.s_addr成员，分别为2字节和4字节的无符号整数，同样需要用网络字节序来表示
  - struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = inet_addr ("192.168.182.48");
  - 其中htons函数将一个16位整数形式的端口号从主机序转成网络序，而inet_addr函数则将一个点分十进制字符串形式的IP地址转成网络序的32位整数形式

知识讲解

将套接字和地址结构绑定

- 将套接字对象和自己的地址结构绑定在一起

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

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

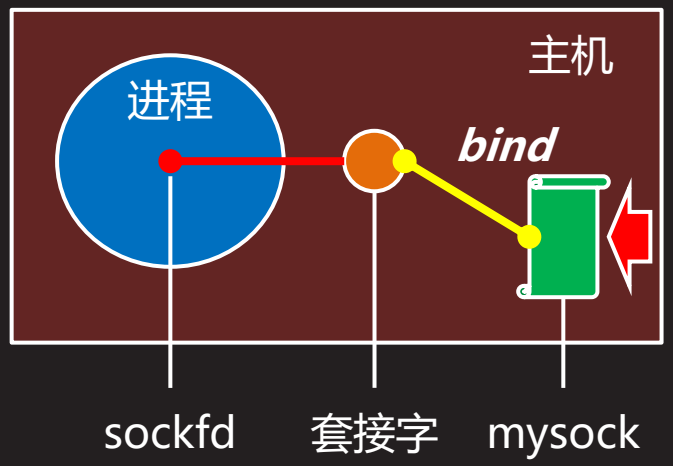
成功返回0，失败返回-1

  - ***sockfd***：套接字描述符
  - ***addr***：自己的地址结构
  - ***addrlen***：地址结构长度(以字节为单位)

- 套接字接口库中的很多函数都用到地址结构，但为了同时支持不同的地址结构类型，其参数往往都会选择更一般化的sockaddr类型的指针，使用时需要强制类型转换

将套接字和地址结构绑定（续1）

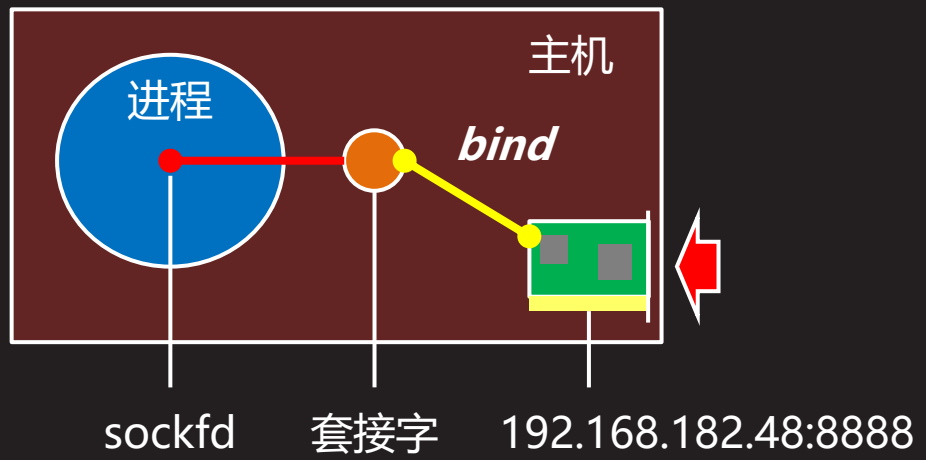
- 例如
  - struct sockaddr_un addr;
    addr.sun_family = AF_LOCAL;
    strcpy (addr.sun_path, "mysock");
    if (bind (sockfd, (struct sockaddr*)&addr,
    sizeof (addr)) == -1) {
    perror ("bind"); exit (EXIT_FAILURE); }
- 在上面例子中，将套接字绑定到一个本地文件上，此后所有针对这个本地文件的操作都可以反映到这个套接字之上



将套接字和地址结构绑定（续2）

知识讲解

- 例如
  - struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = inet_addr ("192.168.182.48");
    if (bind (sockfd, (struct sockaddr*)&addr,
        sizeof (addr)) == -1) {
        perror ("bind"); exit (EXIT_FAILURE); }
- 在上面例子中，将套接字绑定到一个IP和端口上，此后所有针对这个IP和端口的操作都可以反映到这个套接字之上



将套接字和地址结构绑定（续3）

- 例如
  - struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = INADDR_ANY;
    if (bind (sockfd, (struct sockaddr*)&addr,
        sizeof (addr)) == -1) {
        perror ("bind"); exit (EXIT_FAILURE); }
- 在上面的例子中，IP地址使用了INADDR_ANY宏，该宏的值被定义为0，表示任意IP。这样的绑定操作主要用于服务器端，假设服务器主机配置了多个IP地址，无论客户机用哪个IP地址发起通信，服务器套接字都能感觉到

知识讲解

建立连接

- 将套接字对象和对方的地址结构连接在一起

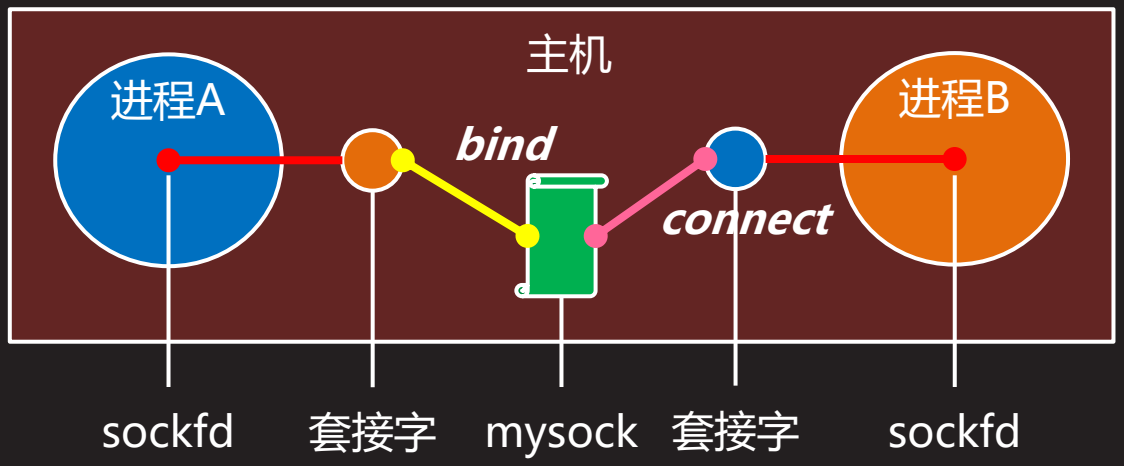
```
#include <sys/socket.h>

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

成功返回0，失败返回-1

  - ***sockfd***：套接字描述符
  - ***addr***：对方的地址结构
  - ***addrlen***：地址结构长度(以字节为单位)

建立连接（续1）

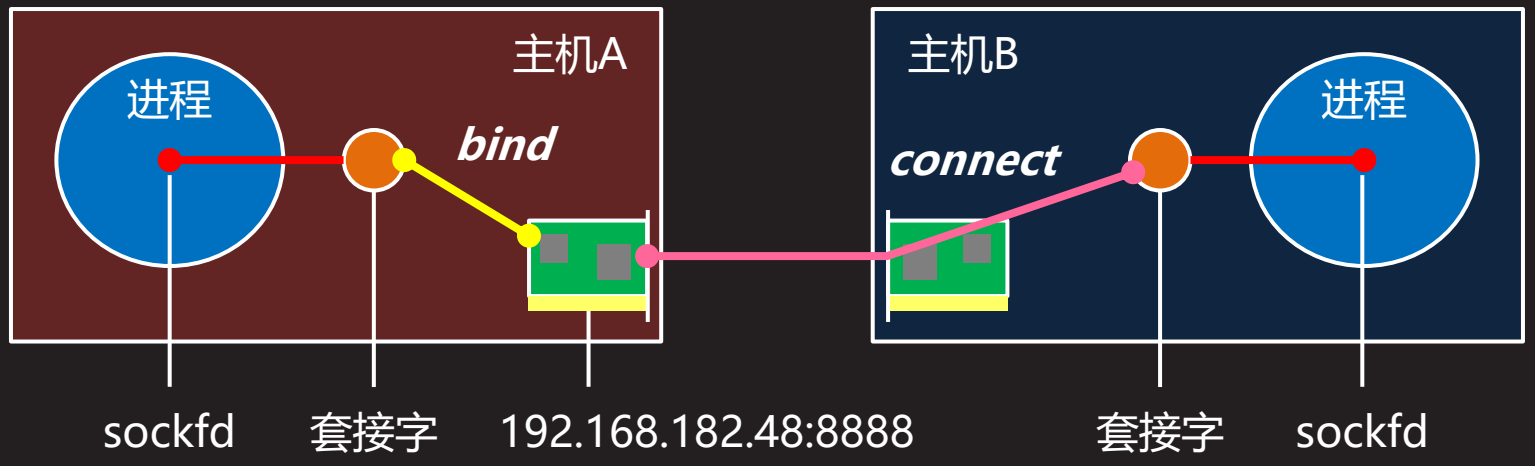
- 例如
  - struct sockaddr_un addr;
    addr.sun_family = AF_LOCAL;
    strcpy (addr.sun_path, "mysock");
    if (connect (sockfd, (struct sockaddr*)&addr,
    sizeof (addr)) == -1) {
    perror ("connect"); exit (EXIT_FAILURE); }



知识讲解

建立连接（续2）

- 例如
  - struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (8888);
    addr.sin_addr.s_addr = inet_addr ("192.168.182.48");
    if (connect (sockfd, (struct sockaddr*)&addr,
    sizeof (addr)) == -1) {
    perror ("connect"); exit (EXIT_FAILURE); }



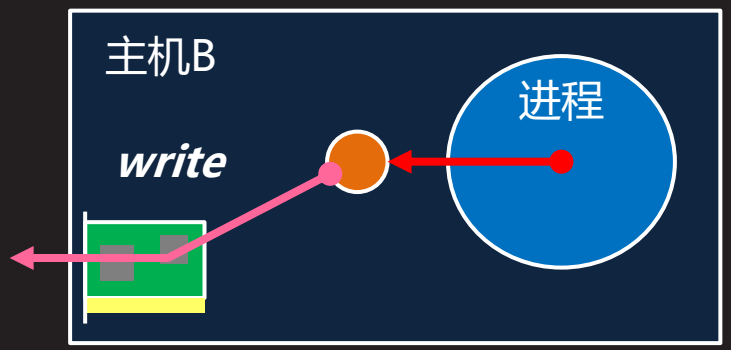
知识讲解

用读写文件的方式通信

- 通过套接字发送字节流

```
#include <unistd.h>

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

        成功返回实际发送的字节数(0表示未发送)，失败返回-1
```

  - ***sockfd***：套接字描述符
  - ***buf***：内存缓冲区
  - ***count***：期望发送的字节数



- 例如
  - ssize_t written = write (sockfd, text, towrite);
    if (written == -1) { perror ("write"); exit (EXIT_FAILURE); }

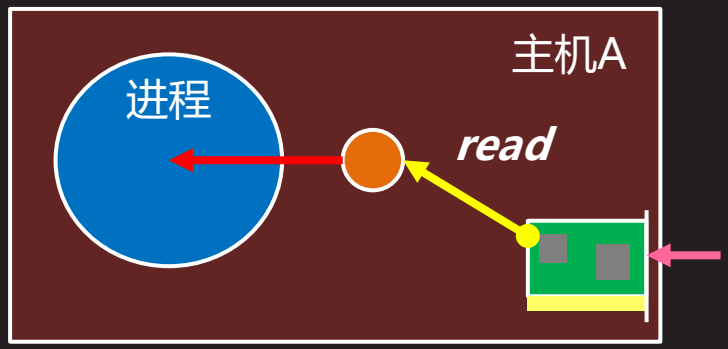
知识讲解

用读写文件的方式通信（续1）

- 通过套接字接收字节流

```
#include <unistd.h>

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

    成功返回实际接收的字节数(0表示连接已关闭)，失败返回-1
```

  - ***sockfd***：套接字描述符
  - ***buf***：内存缓冲区
  - ***count***：期望读取的字节数



- 例如
  - ssize_t readed = read (sockfd, text, toread);
    if (readed == -1) { perror ("read"); exit (EXIT_FAILURE); }

知识讲解

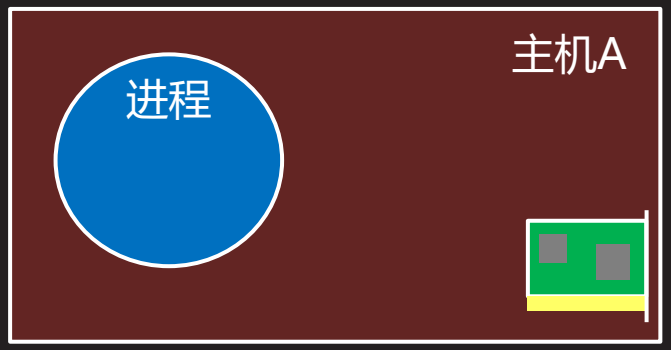
关闭套接字

- 关闭处于打开状态的套接字描述符

```
#include <unistd.h>

int close (int sockfd);
```

成功返回0，失败返回-1

  - *sockfd*：套接字描述符



- 例如
  - if (close (sockfd) == -1) {
    perror ("close");
    exit (EXIT_FAILURE);
    }

知识讲解

字节序转换

- 将主机或网络字节序的长短整数转换为网络或主机字节序

```
#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);
```

h：主机字节序
n：网络字节序
l：长整数(32位)
s：短整数(16位)

返回网络或主机字节序的长短整数

- ***hostlong***：主机字节序长整数
- ***hostshort***：主机字节序短整数
- ***netlong***：网络字节序长整数
- ***netshort***：网络字节序短整数

知识讲解

IP地址转换

- 点分十进制字符串->网络字节序32位无符号整数

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

in_addr_t inet_addr (const char* cp);

                    返回网络字节序32位无符号整数形式的IP地址
```

  - *cp*：点分十进制字符串形式的IP地址
- 例如
  - struct sockaddr_in addr;
    addr.sin_addr.s_addr = inet_addr ("192.168.182.48");

知识讲解

IP地址转换（续1）

- 点分十进制字符串->网络字节序32位无符号整数

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

int inet_aton (const char* cp, struct in_addr* inp);

                                成功返回0，失败返回-1
```

  - *cp*：点分十进制字符串形式的IP地址
  - *inp*：输出包含网络字节序32位无符号整数形式IP地址的in_addr结构

- 例如
  - struct sockaddr_in addr;
    if (inet_aton ("192.168.182.48", &addr.sin_addr) == -1) {
        perror ("inet_aton"); exit (EXIT_FAILURE); }

知识讲解

IP地址转换（续2）

- 网络字节序32位无符号整数->点分十进制字符串

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

char* inet_ntoa (struct in_addr in);

                        返回点分十进制字符串形式的IP地址
```

  - *in*：包含网络字节序32位无符号整数形式IP地址的in_addr结构
- 例如
  - struct sockaddr_in addr;
    inet_aton ("192.168.182.48", &addr.sin_addr);
    printf ("%s\n", inet_ntoa (addr.sin_addr));

知识讲解

Technology Tarena 达内科技

通信编程

进程间通信

- 基于套接字实现进程间通信的编程模型

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	准备地址结构	sockaddr_un	sockaddr_un	准备地址结构	2
3	绑定地址	bind	connect	建立连接	3
4	接收请求	read	write	发送请求	4
5	发送响应	write	read	接收响应	5
6	关闭套接字	close	close	关闭套接字	6

知识讲解

- 创建套接字时使用AF_LOCAL域
  - int sockfd = socket (AF_LOCAL, ...);

进程间通信（续1）

- 准备地址结构时使用sockaddr_un结构体类型
  - struct sockaddr_un addr;
    addr.sun_family = AF_LOCAL;
    strcpy (addr.sun_path, "mysock");
- 套接字文件如不再用需要显式删除
  - unlink ("mysock")

知识讲解

基于套接字的进程间通信

【参见：locsvr.c、loccli.c】

课堂练习

- 基于套接字的进程间通信

网络通信

- 基于套接字实现网络通信的编程模型

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	准备地址结构	sockaddr_in	sockaddr_in	准备地址结构	2
3	绑定地址	bind	connect	建立连接	3
4	接收请求	read	write	发送请求	4
5	发送响应	write	read	接收响应	5
6	关闭套接字	close	close	关闭套接字	6

- 创建套接字时使用AF_INET域
  - int sockfd = socket (AF_INET, ...);

知识讲解

网络通信（续1）

- 准备地址结构时使用sockaddr_in结构体类型
  - 服务器

```
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons (8888);
addr.sin_addr.s_addr = INADDR_ANY;
```

  - 客户机

```
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons (8888);
addr.sin_addr.s_addr = inet_addr ("192.168.182.48");
```

  - 客户机连本机服务器可以使用本地环回地址

```
addr.sin_addr.s_addr = inet_addr ("127.0.0.1");
```

知识讲解

基于套接字的网络通信

【参见：netsvr.c、netcli.c】

课堂练习

- 基于套接字的网络通信

Technology Tarena 达内科技

总结和答疑