附录-常用数据结构——TCP/IP地址

◆TCP/IP地址结构

```
struct sockaddr_in {
    unsigned short int sin_family;        /* 协议族 */
    unsigned short int sin_port;          /* 端口号 */
    struct in_addr {
      unsigned long int s_addr;
    } sin_addr;                                   /* IP地址 */
    unsigned char sin_zero[8]; /* 零填充 */
};
```

✱长度：2 + 2 + 4 + 8 = 16个字节

✱协议族：AF_INET

✱IP地址：INADDR_ANY表示本机所有IP地址

✱用途：设置用于通信的<IP地址，端口>

附录-常用数据结构——通用地址

◆通用地址结构

```
struct sockaddr {
    unsigned short int sa_family;        /* 协议族 */
    unsigned char sa_data[14];           /* 地址数据 */
};
```

✱长度：2 + 14 = 16个字节

✱协议族：常见协议族有TCP/IP协议族（AF_INET）、TCP/IPv6协议族（AF_INET6）、AF_UNIX、AF_IPX、AF_APPLETALK、AF_BLUETOOTH……

✱用途：向函数传递参数，使函数支持多种协议

附录-常用函数——创建socket

◆定义：

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

◆参数：

family：协议族。实验用AF_INET

type：socket类型。常见有SOCK_STREAM、SOCK_DGRAM、SOCK_RAW等

protocol：具体协议。0表示默认协议

◆返回值：

成功：返回一个描述符（正整数），标识所创建的socket

失败：返回-1，并在全局变量errno中记录错误类型

附录-常用函数——绑定地址

◆定义：

```
#include <sys/types.h>
#include <sys/socket.h>
int bind(int sockfd, const struct sockaddr *addr, int addrlen);
```

◆参数：

sockfd：要绑定的socket的描述符

addr：要绑定的地址。此处为指向通用地址结构的指针，使用时，要进行强制类型转换

addrlen：地址结构大小。可使用sizeof自动计算

◆返回值：

成功：返回0

失败：返回-1，并在全局变量errno中记录错误类型

附录-常用函数——服务器启动监听

◆定义：

```
#include <sys/types.h>
#include <sys/socket.h>
int listen(int sockfd, int backlog);
```

◆参数：

sockfd：要监听的socket的描述符

backlog：该socket上完成队列的最大长度。完成队列是指已完成三次握手(established)，但尚未被服务器接受（accept）的客户机

◆返回值：

成功：返回0

失败：返回-1，并在全局变量errno中记录错误类型

附录-常用函数——服务器接受连接

◆定义：

```
#include <sys/types.h>
#include <sys/socket.h>
int accept(int sockfd, struct sockaddr *addr, int *addrlen);
```

◆参数：

sockfd：用于接受连接的socket的描述符

addr：客户机地址。此处为指向通用地址结构的指针，使用时，要进行强制类型转换。如果不关心客户机地址，可以设为NULL

addrlen：客户机地址结构大小。如果不关心，可以和addr一起设为NULL

◆返回值：

成功：返回一个新建的socket，用于数据传输

失败：返回-1，并在全局变量errno中记录错误类型

◆说明：

参数传入的sockfd专用于监听并接受连接，

服务器使用返回的新建socket和客户机传输数据

附录-常用函数——客户端发起连接

◆定义：

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int sockfd, const struct sockaddr *addr, int addrlen);
```

◆参数：

sockfd：用于发起连接的socket的描述符

addr：服务器地址。此处为指向通用地址结构的指针，使用时，要进行强制类型转换

addrlen：服务器地址结构大小。可使用sizeof自动计算

◆返回值：

成功：返回0

失败：返回-1，并在全局变量errno中记录错误类型

◆说明：

客户端connect之前如果未bind，将使用本机地址和随机端口号自动绑定。

不建议绑定固定端口号

附录-常用函数——发送数据

◆定义：

```
#include <sys/types.h>
#include <sys/socket.h>
int send(int sockfd, const void *buf, int len, int flags);
```

◆参数：

sockfd：用于发送数据的socket的描述符

buf：指向数据的指针

len：发送数据大小

flags：额外选项。本次实验设为0

◆返回值：

成功：返回发送的数据大小

失败：返回-1，并在全局变量errno中记录错误类型

◆说明：

send不将数据直接送到网络上，而是写入发送缓冲区，再由系统发送到网络。如果缓冲区空间不足，发送部分数据，返回值<len；如果缓冲区满，程序将阻塞

附录-常用函数——接收数据

◆定义：

```
#include <sys/types.h>
#include <sys/socket.h>
int recv(int sockfd, void *buf, int len, int flags);
```

◆参数：

sockfd：用于接收数据的socket的描述符

buf：指向数据的指针

len：接收数据大小

flags：额外选项。本次实验设为0

◆返回值：

成功：返回接收的数据大小

失败：如果对方已关闭连接，返回0；其他错误返回-1，并在全局变量errno中记录错误类型

◆说明：

recv从接收缓冲区读取指定大小数据。如果缓冲区数据不足，接受部分数据，返回值<len；如果缓冲区无数据可读，程序将阻塞

附录-常用函数——关闭连接

◆定义：

```
#include <unistd.h>
int close(int sockfd);
```

◆参数：

sockfd：要关闭的socket的描述符

◆返回值：

成功：返回0

失败：返回-1，并在全局变量errno中记录错误类型

◆说明：

如果有多个进程同时访问该socket，close只关闭本进程对该socket的访问，不影响其他进程，当所有进程都close后，该socket才被彻底清除

附录-常用函数——错误代码及处理

◆定义：

```
#include <stdio.h>
#include <string.h>
#include <error.h>
void perror(const char *s);
char *strerror(int errnum);
```

◆说明：

函数调用失败时，将返回-1，同时在系统变量errno中保存错误代码。通过perror()和strerror()，可获得错误代码对应的错误信息。perror直接显示具体错误信息，strerror则将错误代码转换为错误信息字串。

◆例如：

在判断bind()返回-1后执行代码：

perror(“bind failed”）;

将显示：

bind failed：xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

等价于：

printf(“bind failed：%s\n”, strerror(errno));

附录-常用函数——字节顺序转换

◆定义：

```
#include <arpa/inet.h>
unsigned long int htonl(unsigned long int integer);
unsigned short int htons(unsigned short int integer);
unsigned long int ntohl(unsigned long int integer);
unsigned short int ntohs(unsigned short int integer);
```

◆参数：

integer：要转换的整型数据

◆返回值：

转换后的整型数据

◆说明：

主机对于多字节的整型数据的表示称为主机字节顺序（HBO），

网络通信使用的表示法称为网络字节顺序（NBO）。

HBO有两种，little-endian和big-endian，NBO只有一种，big-endian。为了让通信双方能正常使用各自的HBO，对于整型数据，发送方调用htonx，接收方调用ntohx。

单字节数据和非整型数据不受影响。

附录-关于Little-Endian和Big-Endian

- Little-Endian小字节序，就是低位字节排放在内存的低地址端、
  高位字节排放在内存的高地址端。
- Big-Endian大字节序，就是高位字节排放在内存的低地址端、
  低位字节排放在内存的高地址端。

- 比如 int a = 0x05060708，在Little-Endian的情况下存放为：

字节号	0	1	2	3
数 据	08	07	06	05

int a = 0x05060708，在Big-Endian的情况下存放为：

字节号	0	1	2	3
数 据	05	06	07	08

- 主机字节序和CPU有关：Intel的x86系列采用Little-Endian，
  其他如PowerPC 、SPARC和Motorola处理器则采用Big-Endian
- 网络字节序：TCP/IP各层协议将字节序定义为Big-Endian

附录-常用函数——地址的数/串转换

◆定义：

#include <arpa/inet.h>

inet_addr(const char *str);

◆参数：

字符串，一个点分十进制的**IP**地址，如： **inet_addr("127.0.0.1")**

◆返回值：

如果正确执行将返回一个无符号长整数型数；如果传入的字符串不是一个合法的IP地址，将返回INADDR_NONE。

◆说明：

inet_addr()作用是将一个IP字符串转化为一个网络字节序的整数值，用于sockaddr_in.sin_addr.s_addr

附录-常用函数——地址的数/串转换

- 定义：

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int inet_aton(const char *str, struct in_addr *num);
                    //将前面字符串转换后写入后面的地址结构体
char *inet_ntoa(struct in_addr num); //将传入的地址结构体转换成字符串
```

- 参数：

str：IP地址字符串形式。如“172.16.13.161”

num：IP地址的整数形式。如“172.16.13.161”，其4字节整数为0xAC100DA1

- 返回值：

inet_aton：成功返回非0值，失败返回0

inet_ntoa：返回转换后的字符串形式

- 说明：

调用这两个函数时，也要做相应的HBO和NBO的字节顺序转换。

inet_ntoa的结果保存在静态缓冲区中，再次调用将覆盖上次的结果

附录-常用函数——字符串转换成整数

◆定义：

```
#include <stdlib.h>
int atoi(const char *str );
```

◆参数：

str：要进行转换的字符串

◆返回值：

每个函数返回 int 值，此值由将输入字符作为数字解析而生成。

如果该输入无法转换为该类型的值，则atoi的返回值为 0。

◆说明：

使用该函数时要注意atoi返回的是int类型，注意输入str的范围不要超出int类型的范围。