[socket编程（1）—— 一对一通信（本地IPC和网络）](http://blog.csdn.net/robot__man/article/details/52302374)

标签： [socket](http://www.csdn.net/tag/socket)[网络编程](http://www.csdn.net/tag/%e7%bd%91%e7%bb%9c%e7%bc%96%e7%a8%8b)

2016-08-24 17:20 84人阅读 [评论](http://blog.csdn.net/robot__man/article/details/52302374#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/robot__man/article/details/52302374#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

Linux下C应用编程（32） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，遵循GPL协议精神，转载请注明出处。

目录[(?)[+]](http://blog.csdn.net/robot__man/article/details/52302374)

一 网络基础知识

1 OSI七层协议模型

　　人和计算机之间交互分了7层(OSI 7层模型)：**物理层、数据链路层、网络层、传输层、会话层、表现层、应用层**。   
　　TCP/IP协议模型中，把应用、表现层和会话层合并为一个大应用层。

2 常见协议

HTTP，FTP，TCP，IP，UDP，收发email的协议   
　　HTTP — 超文本传输协议(互联网)   
　　FTP — 文件传输协议(上传和下载)   
　　TCP — 传输控制协议(一般不翻译)   
　　UDP — 用户数据报协议   
　　IP — Internet协议   
把一组相关的协议，叫做**协议簇(族)**。比如：TCP/IP

3 IP地址、子网掩码、端口

3.1 IP地址

　　IP地址就是计算机在网络中的地址，用4字节/6字节的整数表示，4字节的IPV4，6字节的IPV6。   
　　IP地址 有两种常用的表示方法：   
　　1.计算机底层是支持8位十六进制的32位整数(32位二进制)。   
　　2.人更喜欢点分十进制表示方法，每个字节转成十进制，中间用.隔开， 叫做点分十进制192.168.1.1   
　　两种表示方法底层其实是一样的，比如：   
　　　　点分十进制 172.40.0.10   
　　　　用十六进制可以写成 0xAC28000A   
　　　　192.168.100.20 -> 0xC0A86414

　　查看IP地址的命令：   
　　　　Windows用 ipconfig   
　　　　Unix/[**Linux**](http://lib.csdn.net/base/linux)用 /sbin/ifconfig

　　IP地址又分四级：   
　　　　a类 0   
　　　　b类 10   
　　　　c类 110   
　　　　d类 1110

　　IP地址在网络中可以定位一台计算机。每个网卡在出厂时都有唯一的网卡地址，也叫mac地址(物理地址)，IP地址其实是对应网卡地址然后定位计算机。但IP地址和MAC的绑定是 不固定的。　 

3.2 子网掩码

**子网掩码，主要用于判断是否一个网段。**   
比如：   
　　IP地址： 166.111.160.1 与 166.111.161.45   
　　子网掩码： 255.255.254.0   
IP地址位与子网掩码，如果结果一样，就是同一网段。   
166.111.160.1 & 255.255.254.0 = 166.111.160.0   
166.111.161.45 & 255.255.254.0 = 166.111.160.0

3.3 端口

　　IP地址虽然可以找到计算机，但计算机如果不授权给你，你无法进行任何的操作。计算机用 端口 代表本机的某个进程，访问本机进程必须通过端口进行。

端口是16位的二进制的整数 0 — 65535，其中：   
　　0-1023 基本不用，被系统松散的占用了，比如：   
　　　　80 HTTP的端口(可以改)   
　　　　21 FTP端口 23 TELNET端口   
　　1024 — 48000+ 可用端口，某些软件安装后会占用某些特定端口，比如：   
　　　　1521 [**Oracle**](http://lib.csdn.net/base/oracle)的监听端口   
　　　　8080 Oracle的Http Server   
　　　　7001 weblogic服务器占用的端口([**Java**](http://lib.csdn.net/base/java) 服务器)   
　　　　 ……   
　　48000+ —– 65535 不稳定端口，随时可能被系统拿去做临时的端口

二 socket网络编程

**网络编程就是用IP地址定位计算机，用端口定位对应的进程。**

　　socket 本意就是插座，翻译成套接字。网络编程就是socket编程。   
　　网络编程发展到现在，涵盖的内容已经非常完善，c提供了比较完整的函数和变量支持。

**网络通信包括 一对一(点对点)， 一对多。一对一分为本地通信(IPC)和网络通信。本地通信(两个本地进程) 使用 socket文件做交互媒介，后缀是 .sock，类型为s**

　　整数在存储时，本机有两种可能性，从低字节到高字节和从高字节到低字节，就是所谓的**小端格式**和**大端格式**，但在网络中，字节顺序是唯一的。因此，本机字节顺序和网络字节顺序有可能不一致，htons函数用于整数的本机器格式转网络。

1 一对一通信

一对一通信步骤：   
1 服务端   
1.1 创建一个socket，函数socket()可以创建socket，返回socket描述符。   
　　**int socket(int domain, int type, int protocol)**   
　　参数domain: 域 用于选择协议簇，本地通信/网络   
　　AF\_UNIX/ AF\_LOCAL/ AF\_FILE —- 本地通信   
　　AF\_INET —- 网络通信(IPV4)   
　　AF\_INET6 —- 网络通信(IPV6)   
　　注：AF换成PF也是一样的   
　　参数type： 选择通信方式，同时圈定协议   
　　 SOCK\_STREAM — 数据流，针对TCP协议   
　　SOCK\_DGRAM — 数据报，针对UDP协议   
（数据流数据传送的过程中不间断，数据报在数据传送的过程中将数据打包，分别针对不同的协议TCP和UDP）   
　　参数protocol基本用不到，因为协议前面的2个参数已经确定了，一般给0即可。   
　　函数返回 socket描述符，失败返回-1，socket文件描述符用法和文件描述符一样。

1.2 准备通信地址(本地是文件/网络是IP+端口)   
通信地址有三个结构组成：   
　　struct sockaddr{   
　　　　int sa\_family; //协议簇   
　　　　char sa\_data[];//通信地址   
　　};   
这个结构里面的成员无需记忆，因为这个结构只用来做函数参数类型，从不用来储存数据。这个结构的类型用于兼容 本地通信地址和网络通信地址。   
　　本地通信地址用：

#include <sys/un.h>

struct sockaddr\_un{

int sun\_family;//协议簇 ，和domain保持一致

char sun\_path[];//带路径的socket文件名

};

　　网络通信地址用：

#include<netinet/in.h>

struct sockaddr\_in{

int sin\_family;

short sin\_port;//端口号

struct in\_addr sin\_addr;//IP地址

};

struct in\_addr{

in\_addr\_t s\_addr;

};

1.3 绑定socket和通信地址   
**bind(int sockfd, addr, sizeof(addr))**   
1.4 读写(通信) — read write   
1.5 关闭 —- close

2 客户端   
　　2.1 创建socket   
　　2.2 准备通信地址，如果是网络，准备的是服务器通信地址。   
　　2.3 链接服务器，函数connect()，用法和bind()完全一样   
　　2.4 通信 – 和服务端要交叉进行。   
　　2.5 关闭

　　一对一本地通信：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/un.h>

int main(){

int sockfd = socket(AF\_UNIX,SOCK\_DGRAM,0);

if(sockfd==-1) perror("socket"),exit(-1);

struct sockaddr\_un addr;

addr.sun\_family = AF\_UNIX;

strcpy(addr.sun\_path,"a.sock");

int res = bind(sockfd,

(struct sockaddr\*)&addr,sizeof(addr));

if(res==-1) perror("bind"),exit(-1);

printf("绑定成功\n"); char buf[100] = {};

res = read(sockfd,buf,100);

printf("读到了%d字节，内容：%s\n",res,buf);

close(sockfd);

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/un.h>

int main(){

int sockfd = socket(AF\_UNIX,SOCK\_DGRAM,0);

if(sockfd==-1) perror("socket"),exit(-1);

struct sockaddr\_un addr;

addr.sun\_family = AF\_UNIX;

strcpy(addr.sun\_path,"a.sock");

int res = connect(sockfd,

(struct sockaddr\*)&addr,sizeof(addr));

if(res==-1) perror("connect"),exit(-1);

printf("连接成功\n");

write(sockfd,"hello",5);

close(sockfd);

}

35

　　网络通信地址结构中的sin\_addr是十六进制方式。点分十进制转换十六进制的函数 inet\_addr()。   
　　一对一网络通信：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>//

#include <arpa/inet.h>

int main(){

int sockfd = socket(AF\_INET,SOCK\_DGRAM,0);//

if(sockfd==-1) perror("socket"),exit(-1);

struct sockaddr\_in addr;//

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(2222);//本机转网络

addr.sin\_addr.s\_addr = //点分十进制转

inet\_addr("172.40.0.10");//十六进制

int res = bind(sockfd,

(struct sockaddr\*)&addr,sizeof(addr));

if(res==-1) perror("bind"),exit(-1);

printf("绑定成功\n"); char buf[100] = {};

res = read(sockfd,buf,100);

printf("读到了%d字节，内容：%s\n",res,buf);

close(sockfd);

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int main(){

int sockfd = socket(AF\_INET,SOCK\_DGRAM,0);

if(sockfd==-1) perror("socket"),exit(-1);

struct sockaddr\_in addr;

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(2222);//连接端口

addr.sin\_addr.s\_addr = //都是服务器的

inet\_addr("172.40.0.10");//改成连接IP

int res = connect(sockfd,

(struct sockaddr\*)&addr,sizeof(addr));

if(res==-1) perror("connect"),exit(-1);

printf("连接成功\n");

write(sockfd,"hello",5);

close(sockfd);

}