<http://blog.csdn.net/cabing2005/article/details/53068880>

版权声明：本文为博主原创文章，未经博主允许不得转载。 http://blog.csdn.net/cabing2005/article/details/53068880

目录[(?)](http://blog.csdn.net/cabing2005/article/details/53068880" \o "系统根据文章中H1到H6标签自动生成文章目录)[[+]](http://blog.csdn.net/cabing2005/article/details/53068880)

## linux socket编程以及简单的tcp,udp的例子

通过socket编程大致对网路编程的脉络有个大致的了解，如果有不太懂的地方，   
咱们可以到网络上查找手册

### socket的基本操作

创建socket

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

1. domain 指定使用何种的地址类型

协议 说明

PF\_UNIX/PF\_LOCAL/AF\_UNIX/AF\_LOCAL UNIX 进程通信协议

PF\_INET/AF\_INET Ipv4 网络协议

PF\_INET6/AF\_INET6 Ipv6 网络协议

PF\_IPX/AF\_IPX IPX-Novell 协议

PF\_NETLINK/AF\_NETLINK 核心用户接口装置

PF\_X25/AF\_X25 ITU-T X. 25/ISO-8208 协议

PF\_AX25/AF\_AX25 业余无线AX. 25 协议

PF\_ATMPVC/AF\_ATMPVC 存取原始 ATM PVCs

PF\_APPLETALK/AF\_APPLETALK appletalk (DDP)协议

PF\_PACKET/AF\_PACKET 初级封包接口

PF\_INET/AF\_INET Ipv4 网络协议

PF\_INET/AF\_INET Ipv4 网络协议

2. type的类别。见表：

类型 说明

SOCK\_STREAM 提供双向连续且可信赖的数据流, 即TCP

SOCK\_DGRAM 使用不连续不可信赖的数据包连接

SOCK\_SEQPACKET 提供连续可信赖的数据包连接

SOCK\_RAW 提供原始网络协议存取

SOCK\_RDM 提供可信赖的数据包连接

SOCK\_PACKET 提供和网络驱动程序直接通信

3. protocol 用来指定socket 所使用的传输协议编号，通常为0

成功返回0 失败返回-1

命名socket

int bind(int sockfd, struct sockaddr \* my\_addr, int addrlen);

0 正确

常用的错误：EACCES,EADDRINUSE

监听socket–创建一个监听队列

int liseten(int socketfd,int backlog)

接受连接

accet(int socketfd,struct sockaddr \* my\_addr, int addrlen)

从队列中接受一个连接

发起连接

connect(int socketfd,struct sockaddr \* my\_addr, int addrlen)

关闭连接

close(int fd)

tcp数据读写

recv(int fd,void\*buf,size\_t len,int flags)

send(int fd,const void \*buf,size\_t len,int flags)

flag 有一些特别的参数

持续监听对方的回应

不经过路由表

对socket非阻塞

发送紧急数据

不接受sig信号等等

udp数据读写

recvfrom(int fd,void\*buf,size\_t len,int flags,struct sockaddr \* my\_addr, int addrlen)

sendto(int fd,const void\*buf,size\_t len,int flags,struct sockaddr \* my\_addr, int addrlen)

通用数据读写

recvmsg(int fd,struct msghdr\*msg,int flags);

sendmsg(int fd,struct msghdr\*msg,int flags);

struct msghdr

{

void\* msg\_name;

socklen\_t msg\_namelen;

struct iovec\*msg\_iov;

int msg\_iovlen;

void\* msg\_control;

socklen\_t msg\_controllen;

int msg\_flags;

};

struct iovec

{

void \*iov\_base;

size\_t iov\_len;

};

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17

带外标记判断

int sockatmark(int sockfd);

地址信息函数

int getsockname(int socketfd,struct sockaddr \* my\_addr, int addrlen);//本端的地址信息

int getpeername(int socketfd,struct sockaddr \* my\_addr, int addrlen);//远端的地址信息

socket选项

获取socket选项

getsockopt(int socketfd,int opt\_name,void\*option\_value,socket\_t restrict option\_len)

设置socket选项

getsockopt(int socketfd,int opt\_name,void\*option\_value,socket\_t restrict option\_len)

这个可以查看手册。

选项名称　　　　　　　　说明　　　　　　　　　　　　　　　　　　数据类型

========================================================================

　　　　　　　　　　　　SOL\_SOCKET

------------------------------------------------------------------------

SO\_BROADCAST　　　　　　允许发送广播数据　　　　　　　　　　　　int

SO\_DEBUG　　　　　　　　允许调试　　　　　　　　　　　　　　　　int

SO\_DONTROUTE　　　　　　不查找路由　　　　　　　　　　　　　　　int

SO\_ERROR　　　　　　　　获得套接字错误　　　　　　　　　　　　　int

SO\_KEEPALIVE　　　　　　保持连接　　　　　　　　　　　　　　　　int

SO\_LINGER　　　　　　　 延迟关闭连接　　　　　　　　　　　　　　struct linger

SO\_OOBINLINE　　　　　　带外数据放入正常数据流　　　　　　　　　int

SO\_RCVBUF　　　　　　　 接收缓冲区大小　　　　　　　　　　　　　int

SO\_SNDBUF　　　　　　　 发送缓冲区大小　　　　　　　　　　　　　int

SO\_RCVLOWAT　　　　　　 接收缓冲区下限　　　　　　　　　　　　　int

SO\_SNDLOWAT　　　　　　 发送缓冲区下限　　　　　　　　　　　　　int

SO\_RCVTIMEO　　　　　　 接收超时　　　　　　　　　　　　　　　　struct timeval

SO\_SNDTIMEO　　　　　　 发送超时　　　　　　　　　　　　　　　　struct timeval

SO\_REUSERADDR　　　　　 允许重用本地地址和端口　　　　　　　　　int

SO\_TYPE　　　　　　　　 获得套接字类型　　　　　　　　　　　　　int

SO\_BSDCOMPAT　　　　　　与BSD系统兼容　　　　　　　　　　　　　 int

========================================================================

　　　　　　　　　　　　IPPROTO\_IP

------------------------------------------------------------------------

IP\_HDRINCL　　　　　　　在数据包中包含IP首部　　　　　　　　　　int

IP\_OPTINOS　　　　　　　IP首部选项　　　　　　　　　　　　　　　int

IP\_TOS　　　　　　　　　服务类型

IP\_TTL　　　　　　　　　生存时间　　　　　　　　　　　　　　　　int

========================================================================

　　　　　　　　　　　　IPPRO\_TCP

------------------------------------------------------------------------

TCP\_MAXSEG　　　　　　　TCP最大数据段的大小　　　　　　　　　　 int

TCP\_NODELAY　　　　　　 不使用Nagle算法　　　　　　　　　　　　 int

========================================================================

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33

获取主机信息

根据名字获取主机信息

struct hostent\* gethostbyname(const char \*name)

根据ip获取主机信息

struct hostent\* gethostbyaddr(const void \*addr,size\_t len,int type)

struct hostent

{

char \*h\_name;//主机名

char\*\* h\_aliases;//主机别名，可能有多个

int h\_addrtype;//地址类型

int h\_length;// 地址长度

char \*\* h\_addr\_list//按照网络字节序列列出idp地址列表

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

获取服务信息

//根据 名字，端口号获取服务信息

struct servent\*getservbyname(const char \*name,const char \*proto);

//根据服务类型，端口号获取服务信息

struct servent\*getservbyport(int port,const char \*proto);

struct servent{

char \*h\_name;//主机名

char\*\* h\_aliases;//主机别名，可能有多个

int s\_port;//端口号

char \*s\_proto;//服务类型，tcp或者upd等

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

通过主机名获取ip地址

int getaddrinfo( const char \*hostname, const char \*service, const struct addrinfo \*hints, struct addrinfo \*\*result );

通过socket地址获取主机名

int getnameinfo (const struct sockaddr \*sockaddr, socklen\_t addrlen, char \*host, socklen\_t hostlen, char \*serv, socklen\_t servlen, int flags) ;

tcp编程的例子server

//

// main.cpp

// linux\_socket\_api

//

// Created by bikang on 16/11/2.

// Copyright (c) 2016年 bikang. All rights reserved.

//

#include <iostream>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <signal.h>

#include <unistd.h>

#include <stdlib.h>

#include <assert.h>

#include <stdio.h>

#include <string.h>

//using namespace std;

#define BUFFER\_SIZE 1024

void tsocket(int argc, const char \* argv[]);

int main(int argc, const char \* argv[]) {

tsocket(argc,argv);

return 0;

}

void tsocket(int argc, const char \* argv[]){

if(argc < 3){

exit(-1);

}

const char\* ip = argv[1];

int port = atoi(argv[2]);

int backlog = atoi(argv[3]);

std::cout << "ip=" << ip << " port="<<port << " backlog=" << backlog << std::endl;

int fd;

int check\_ret;

fd = socket(PF\_INET,SOCK\_STREAM , 0);

assert(fd >= 0);

struct sockaddr\_in address;

bzero(&address,sizeof(address));

//转换成网络地址

address.sin\_port = htons(port);

address.sin\_family = AF\_INET;

//地址转换

inet\_pton(AF\_INET, ip, &address.sin\_addr);

//设置socket buffer大小

int recvbuf = 4096;

int len = sizeof( recvbuf );

setsockopt( fd, SOL\_SOCKET, SO\_RCVBUF, &recvbuf, sizeof( recvbuf ) );

getsockopt( fd, SOL\_SOCKET, SO\_RCVBUF, &recvbuf, ( socklen\_t\* )&len );

printf( "the receive buffer size after settting is %d\n", recvbuf );

//绑定ip和端口

check\_ret = bind(fd,(struct sockaddr\*)&address,sizeof(address));

assert(check\_ret >= 0);

//创建监听队列，用来存放待处理的客户连接

check\_ret = listen(fd, backlog);

assert(check\_ret >= 0);

struct sockaddr\_in addressClient;

socklen\_t clientLen = sizeof(addressClient);

//接受连接，阻塞函数

int connfd = accept(fd, (struct sockaddr\*)&addressClient, &clientLen);

if(connfd < 0){

std::cout << "accept error";

}else{

//打印客户端信息

char showData[INET\_ADDRSTRLEN];

std::cout <<inet\_ntop(AF\_INET,&addressClient.sin\_addr,showData,INET\_ADDRSTRLEN)<<":" <<ntohs(addressClient.sin\_port)<<std::endl;

//接受数据

const int BUF\_LEN = 1024;

char sockBuf[BUF\_LEN];

size\_t ret;

memset(sockBuf, '\0', BUF\_LEN);

ret = recv(connfd, sockBuf, BUF\_LEN-1, 0);

printf("ret=%ld,msg=%s\n",ret,sockBuf);

memset(sockBuf, '\0', BUF\_LEN);

ret = recv(connfd, sockBuf, BUF\_LEN-1, MSG\_OOB);

printf("ret=%ld,msg=%s\n",ret,sockBuf);

memset(sockBuf, '\0', BUF\_LEN);

ret = recv(connfd, sockBuf, BUF\_LEN-1, 0);

printf("ret=%ld,msg=%s\n",ret,sockBuf);

//获取本地socket信息

struct sockaddr\_in tmpAddress;

clientLen = sizeof(tmpAddress);

getsockname(fd, (struct sockaddr\*)&tmpAddress, &clientLen);

std::cout <<inet\_ntop(AF\_INET,&tmpAddress.sin\_addr,showData,INET\_ADDRSTRLEN)<<":" <<ntohs(tmpAddress.sin\_port)<<std::endl;

//获取远端socket信息

getpeername(connfd,(struct sockaddr\*)&tmpAddress, &clientLen );

std::cout <<inet\_ntop(AF\_INET,&tmpAddress.sin\_addr,showData,INET\_ADDRSTRLEN)<<":" <<ntohs(tmpAddress.sin\_port)<<std::endl;

close(connfd);

}

close(fd);

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74
* 75
* 76
* 77
* 78
* 79
* 80
* 81
* 82
* 83
* 84
* 85
* 86
* 87
* 88
* 89
* 90
* 91
* 92
* 93
* 94
* 95
* 96
* 97
* 98
* 99
* 100
* 101
* 102
* 103
* 104
* 105
* 106
* 107
* 108
* 109
* 110
* 111
* 112
* 113
* 114
* 115
* 116
* 117
* 118
* 119

tcp编程的例子client

//

// main.cpp

// linux\_socket\_api\_client

//

// Created by bikang on 16/11/2.

// Copyright (c) 2016年 bikang. All rights reserved.

//

#include <iostream>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <signal.h>

#include <unistd.h>

#include <stdlib.h>

#include <assert.h>

#include <stdio.h>

#include <string.h>

void tserver(int argc, const char \* argv[]);

int main(int argc, const char \* argv[]) {

tserver(argc,argv);

return 0;

}

void tserver(int argc, const char \* argv[]){

std::cout << "t server" << std::endl;

if(argc < 3){

exit(-1);

}

const char\* ip = argv[1];

int port = atoi(argv[2]);

int backlog = atoi(argv[3]);

std::cout << "ip=" << ip << " port="<<port << " backlog=" << backlog << std::endl;

int fd;

int check\_ret;

fd = socket(PF\_INET,SOCK\_STREAM , 0);

assert(fd >= 0);

int sendbuf = 4096;

int len = sizeof( sendbuf );

setsockopt( fd, SOL\_SOCKET, SO\_SNDBUF, &sendbuf, sizeof( sendbuf ) );

getsockopt( fd, SOL\_SOCKET, SO\_SNDBUF, &sendbuf, ( socklen\_t\* )&len );

printf( "the send buffer size after settting is %d\n", sendbuf );

struct sockaddr\_in address;

bzero(&address,sizeof(address));

//转换成网络地址

address.sin\_port = htons(port);

address.sin\_family = AF\_INET;

//地址转换

inet\_pton(AF\_INET, ip, &address.sin\_addr);

check\_ret = connect(fd, (struct sockaddr\*) &address, sizeof(address));

assert(check\_ret >= 0);

//发送数据

const char\* oob\_data = "abc";

const char\* normal\_data = "my boy!";

send(fd, normal\_data, strlen(normal\_data), 0);

send(fd, oob\_data, strlen(oob\_data), MSG\_OOB);

send(fd, normal\_data, strlen(normal\_data), 0);

close(fd);

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73

udp编程的例子server

#include <iostream>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <signal.h>

#include <unistd.h>

#include <stdlib.h>

#include <assert.h>

#include <stdio.h>

#include <string.h>

//using namespace std;

#define BUFFER\_SIZE 1024

void tsocket(int argc, const char \* argv[]);

int main(int argc, const char \* argv[]) {

tsocket(argc,argv);

return 0;

}

void tsocket(int argc, const char \* argv[]){

if(argc < 3){

exit(-1);

}

const char\* ip = argv[1];

int port = atoi(argv[2]);

int backlog = atoi(argv[3]);

std::cout << "ip=" << ip << " port="<<port << " backlog=" << backlog << std::endl;

int fd;

int check\_ret;

fd = socket(PF\_INET,SOCK\_DGRAM , 0);

assert(fd >= 0);

struct sockaddr\_in address;

bzero(&address,sizeof(address));

//转换成网络地址

address.sin\_port = htons(port);

address.sin\_family = AF\_INET;

//地址转换

inet\_pton(AF\_INET, ip, &address.sin\_addr);

//绑定ip和端口

check\_ret = bind(fd,(struct sockaddr\*)&address,sizeof(address));

assert(check\_ret >= 0);

while(1){

char buffer[BUFFER\_SIZE];

struct sockaddr\_in addressClient;

socklen\_t clientLen = sizeof(addressClient);

memset(buffer, '\0', BUFFER\_SIZE);

//获取信息

if(recvfrom(fd, buffer, BUFFER\_SIZE-1,0,(struct sockaddr\*)&addressClient, &clientLen) == -1)

{

perror("Receive Data Failed:");

exit(1);

}

printf("buffer=%s\n", buffer);

}

close(fd);

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74

udp编程的例子client

//

// main.cpp

// linux\_socket\_api\_client

//

// Created by bikang on 16/11/2.

// Copyright (c) 2016年 bikang. All rights reserved.

//

#include <iostream>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <signal.h>

#include <unistd.h>

#include <stdlib.h>

#include <assert.h>

#include <stdio.h>

#include <string.h>

void tserver(int argc, const char \* argv[]);

int main(int argc, const char \* argv[]) {

tserver(argc,argv);

return 0;

}

void tserver(int argc, const char \* argv[]){

std::cout << "t server" << std::endl;

if(argc < 3){

exit(-1);

}

const char\* ip = argv[1];

int port = atoi(argv[2]);

int backlog = atoi(argv[3]);

std::cout << "ip=" << ip << " port="<<port << " backlog=" << backlog << std::endl;

int fd;

int check\_ret;

fd = socket(PF\_INET,SOCK\_DGRAM , 0);

assert(fd >= 0);

struct sockaddr\_in address;

bzero(&address,sizeof(address));

//转换成网络地址

address.sin\_port = htons(port);

address.sin\_family = AF\_INET;

//地址转换

inet\_pton(AF\_INET, ip, &address.sin\_addr);

//发送数据

const char\* normal\_data = "my boy!";

if(sendto(fd, normal\_data, strlen(normal\_data),0,(struct sockaddr\*)&address,sizeof(address)) < 0)

{

perror("Send File Name Failed:");

exit(1);

}

close(fd);

}