wjlkoorey的博客

wjlkoorey.blog.chinaunix.net

wanderlust in the sea...

原创 Linux网络编程：基于UDP的程序开发回顾篇

2012-07-16 23:43:10

分类： LINUX

基于无连接的UDP程序设计

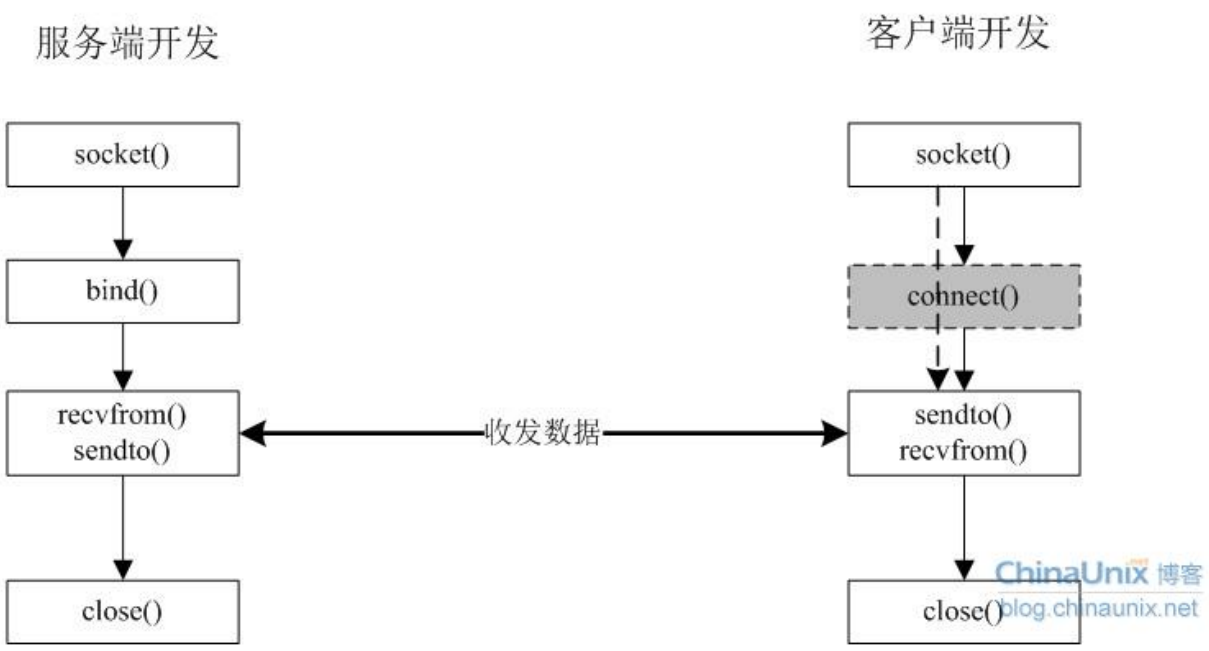
同样，在开发基于UDP的应用程序时，其主要流程如下：

wjlkoorey258

博客访问： 509974
博文数量： 97
博客积分： 671
博客等级： 上尉
技术积分： 10228
用 户 组： 普通用户
注册时间： 2010-12-18 16:08

个人简介

www.5678520.com



对于面向无连接的UDP应用程序在开发过程中服务端和客户端的操作流程基本差不多。对比面向连接的TCP程序，服务端少了listen和accept函数。前面我们也说过listen函数最主要的作用就是将一个socket套接字描述符转为被动监听模式，然后调用accept主要是用于等待客户端(用connect)来连接服务器。connect函数不仅可以用于流式套接字还可用于数据报式套接字。在TCP中，客户端调用connect函数会向服务器端触发一个TCP的3次握手过程，去建立一条TCP连接；而在UDP中，客户端调用该函数主要的作用是告诉后面将要调用的recvfrom函数，仅仅只接受在connect函数中指明的服务器发来的数据，这样当后面调用recvfrom时最后两个参数就可以置为NULL了。也就说对UDP编程来说，客户端调用connect是可选的：如果调用了connect函数，recvfrom就可以省掉最后两个参数；如果不调用connect则recvfrom必须指明从哪儿收数据。

对于UDP的编程其实主要在数据的收发处理上，而面向无连接的UDP编程中收发数据用到的最多的函数就是recvfrom()和sendto()，其原型如下：

ssize_t recvfrom(int s, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen);

多媒体（7）
网络编程（8）
系统管理（3）
SNMP（2）
Netfilter和iptab（0）
未分配的博文（39）

文章存档

2014年（12）
2013年（21）
2012年（64）

我的朋友

2005227
lbird_11
tyj19891
lawrence
mzh2100
huangba
liucaipi
Zackary1
WbBullFr

最近访客

ssize_t sendto(int s, const void *buf, size_t len, int flags, struct sockaddr *to, socklen_t tolen);

recvfrom函数主要用于从s所指定的套接字中接收数据，并将其存储在buf所指向的缓冲区里。如果from参数不为NULL，那么其中便会携带消息发送端的地址信息，fromlen则指明了信息发送方地址信息结构体的大小。如果接收方对发送发的地址不感兴趣，将from和fromlen置为NULL即可。返回值：小于0，有错误；大于0，实际收到的字节数；等于0，对端主动关闭。

sendto函数，主要是buf所指向的数据发送到套接字描述符s中，len为要发送的数据长度，to中存储了对端的地址信息，即数据该发往何处，tolen为to所占的字节数。返回值：小于0，有错误；大于0，实际发送的字节数。

另外我们还知道，sendto是可以用于面向连接的流式套接字的，在TCP开发章节我们已经提过。这里在罗嗦一点，如果sendto用于面向流式的套接字编程中，to和tolen参数都会被忽略，如果发送数据时连接还未建立相应的提示错误为ENOTCONN。

这里也没有哪个规定说是不准在TCP程序中用sendto，但我们一般都不这么做，自己体会一下就明白了，除非你的项目开发中有特殊需求必须用。一句话：记住sendto和recvfrom既可以用于面向连接的流式套接字中收发数据，也可以用于面向无连接的数据报式套接字。sendto()和recvfrom()一般用在面向无连接的数据报式套接字的程序开发中。

看个小例子：

UDP服务器端代码：udpsrv.c

点击(此处)折叠或打开

```
1.  #include <stdlib.h>
2.  #include <stdio.h>
3.  #include <errno.h>
4.  #include <string.h>
5.  #include <unistd.h>
6.  #include <netdb.h>
```


acorpe 雷锋不谢 y841618


cym0417 猪也有春 米娜拉夜


hmily36 VEGETA phoenixc

订阅

订阅

订阅到 Google

推荐博文

·模仿之中也少不了创新——Leo...
·学习Swift之(二)：swift开发...
·学习Swift之(一)：关于swift...
·实现dup2函数,要求不使用fcnt...
·LR模型的Spark实现
·对Oracle高水位线的研究实践...
·为学习Hadoop使用VMware准备3...
·【故障处理】opmn启动失败及...
·oracle 11g ASM 磁盘组在线扩...
·数据迁移中的数据库检查和建...

热词专题

```
7.  #include <sys/socket.h>
8.  #include <netinet/in.h>
9.  #include <sys/types.h>
10. #include <arpa/inet.h>
11. #define MAX_MSG_SIZE 1024
12.
13. int main(int argc,char** argv){
14.     int skfd,addrlen,ret;
15.     struct sockaddr_in addr,cltaddr;
16.     char buf[MAX_MSG_SIZE]={0};
17.     char sndbuf[MAX_MSG_SIZE]={0};
18.
19.     //创建数据报式套接字skfd
20.     if(0>(skfd=socket(AF_INET,SOCK_DGRAM,0))){
21.         perror("Create Error");
22.         exit(1);
23.     }
24.
25.     bzero(&addr,sizeof(struct sockaddr_in));
26.     addr.sin_family = AF_INET;
27.     addr.sin_addr.s_addr=htonl(INADDR_ANY);
28.     addr.sin_port=htons(atoi(argv[1]));
29.
30.     //将socket文件描述符skfd和本地端口和地址绑定起来
31.     if(0>(bind(skfd,(struct sockaddr*)&addr,sizeof(struct sockaddr_in)))){
32.         perror("Bind Error");
33.         exit(1);
34.     }
35.
36.     //开始收发数据
```

·李体育老师荣获“杰出传承人...
·string.h
·安装oracle
·VMware VDP
·Android + 系统属性

```
37.     while(1){
38.         ret=recvfrom(skfd,buf,MAX_MSG_SIZE,0,(struct sockaddr*)&cltaddr,&addrlen)
    ;
39.         if(ret < 0){
40.             printf("recv data from %s:%d error!",inet_ntoa(cltaddr.sin_addr),ntohs(cltaddr
    .sin_port));
41.         }else if(ret == 0){
42.             perror("client has been closing socket!");
43.         }else{
44.             printf("From %s:%d,%s",inet_ntoa(cltaddr.sin_addr),ntohs(cltaddr.sin_port),b
    uf);
45.             memset(sndbuf,0,MAX_MSG_SIZE);
46.             switch(buf[0]){
47.                 case 'a':
48.                     strcpy(sndbuf,"After u ,lady...");
49.                 break;
50.                 case 'b':
51.                     strcpy(sndbuf,"Before u ,sir...");
52.                 break;
53.                 case 'c':
54.                     strcpy(sndbuf,"Can u?");
55.                     break;
56.                 default:
57.                     strcpy(sndbuf,"I dont't know what u want!");
58.             }
59.             sendto(skfd,sndbuf,strlen(sndbuf),0,(struct sockaddr*)&cltaddr,addrlen);
60.         }
61.         memset(buf,0,MAX_MSG_SIZE);
62.     }
63.     return 0;
```

```
64.  }
```

UDP客户端代码：udpclt.c

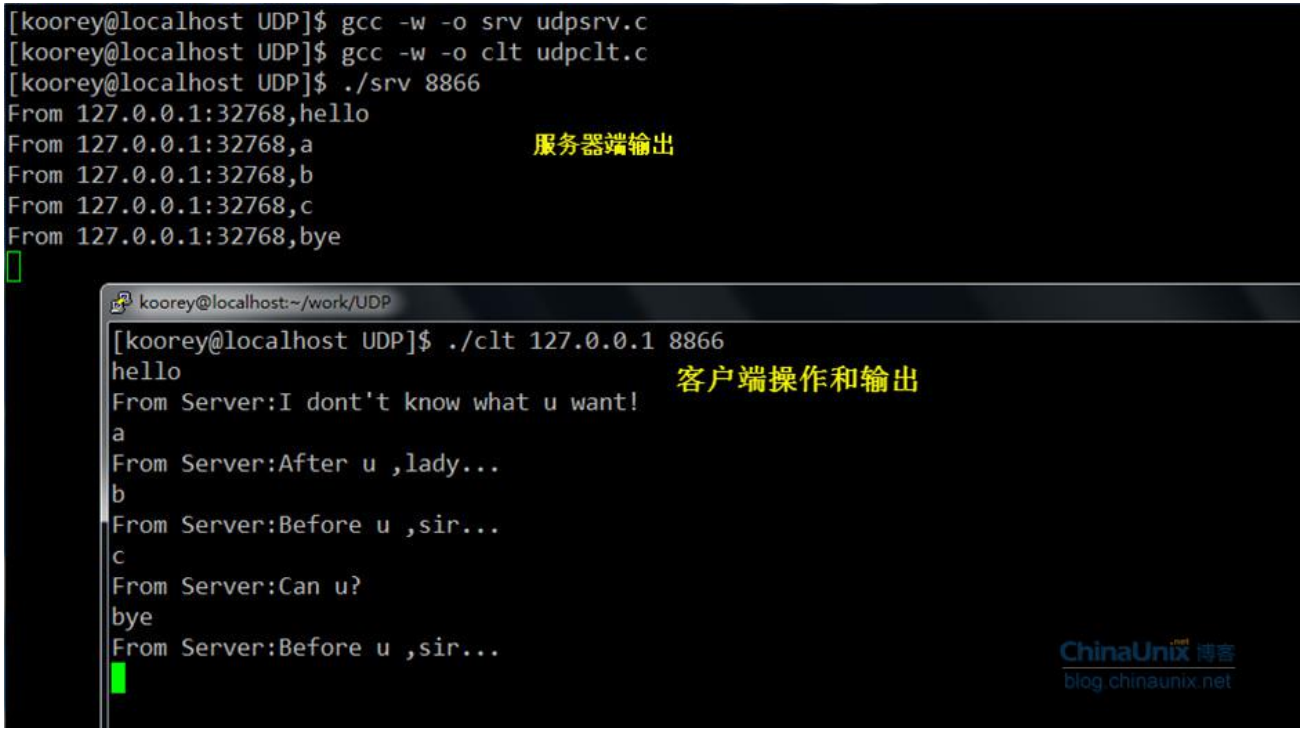
点击(此处)折叠或打开

```
1.  #include <stdlib.h>
2.  #include <stdio.h>
3.  #include <errno.h>
4.  #include <string.h>
5.  #include <unistd.h>
6.  #include <netdb.h>
7.  #include <sys/socket.h>
8.  #include <netinet/in.h>
9.  #include <sys/types.h>
10. #include <arpa/inet.h>
11. #define MAX_MSG_SIZE 1024
12.
13. int main(int argc,char** argv){
14.     int skfd,ret,len;
15.     struct sockaddr_in srvaddr;
16.     char buf[MAX_MSG_SIZE]={0};
17.     char sndbuf[MAX_MSG_SIZE]={0};
18.     struct in_addr addr;
19.
20.     //创建数据报式套接字skfd
21.     if(0>(skfd=socket(AF_INET,SOCK_DGRAM,0))){
22.         perror("Create Error");
23.         exit(1);
24.     }
25.
26.     if(0 == inet_aton(argv[1],&addr)){
```

```
27.         perror("server addr invalid!");
28.         exit(1);
29.     }
30.
31.     bzero(&srvaddr,sizeof(struct sockaddr_in));
32.     srvaddr.sin_family = AF_INET;
33.     srvaddr.sin_addr=addr;
34.     srvaddr.sin_port=htons(atoi(argv[2]));
35.
36.     //我们的客户端只接收从服务器地址是srvaddr的主机发来的数据
37.     if(0>(connect(skfd,(struct sockaddr*)&srvaddr,sizeof(struct sockaddr_in)))){
38.         perror("Connect Error");
39.         exit(1);
40.     }
41.
42.     //开始收发数据
43.     while(1){
44.         memset(sndbuf,0,MAX_MSG_SIZE);
45.         len=read(0,sndbuf,MAX_MSG_SIZE);
46.         ret=sendto(skfd,sndbuf,strlen(sndbuf),0,(struct sockaddr*)&srvaddr,sizeof(struc
    t sockaddr));
47.         if(ret == len){
48.             memset(buf,0,MAX_MSG_SIZE);
49.             //我们已经知道服务器地址信息了，所以最后两个参数为NULL
50.             ret=recvfrom(skfd,buf,MAX_MSG_SIZE,0,NULL,NULL);
51.
52.             if(ret < 0){
53.                 perror("read error from server!");
54.             }else if(ret == 0){
55.                 perror("server has been closing socket!");
```

```
56.             }else{
57.                 buf[ret]='\0';
58.                 printf("From Server:%s\n",buf);
59.             }
60.         }
61.     }
62.     return 0;
63. }
```

测试结果：



我们客户端接收用户命令行输入的指令，然后将其发给UDP服务器端；服务器端收到不同的指令后给客户端予以不同的提示信息，整个流程如上所示。

udpclt.c的示例代码中我是调用了connect，勤奋好学的童鞋可以动手试验哈不调用connect然后将程序调通吧。

上一篇：Linux网络编程：基于TCP的程序开发回顾篇
下一篇：Linux网络编程：原始套接字的魔力【上】

相关热门文章

热 轻量级web server Tornado代码...
荐 linux 常见服务端口
热 kernel 报错l701.exel[16922]:...
热 2013年世界科技发展回顾...
荐 【ROOTFS搭建】busybox的httpd...
热 C语言 如何在一个整型左边补0...
热 经纬财富:南充金银探底回升 利...
荐 xmanager 2.0 for linux配置
热 python无法爬取阿里巴巴的数据...
热 经纬财富:安庆金银大跌 千三关...
荐 什么是shell
热 linux-2.6.28 和linux-2.6.32....
热 Android日志系统驱动程序Logg...
荐 linux socket的bug??
热 linux su - username -c 命...

给主人留下些什么吧！~~

评论热议

请登录后评论。

登录 注册