Socket 扩展编程

Andrew Huang< bluedrum@163.com>

内容

- Winsock编程
- 特殊情况处理
  - 特殊Socket参数的设置和处理
  - UDP广播的实现
- 通用socket的设计

windows下Socket编程

Windows下Socket编程

- 由于BSD Socket已经成为事实的标准,所以Windows下也采用相同编程接口进行编程.
- 在Linux下的Socket程序基本上不需要经过太多改动即可在Windows下执行.
- 但是在细微的地方还是有差异.主要体现在如下方面
  - 头文件有个别不同
  - 链接库不一样
  - Windows Socket必须有独立初始化代码
  - Socket数据类型不一样
  - 错误返回值不是-1,SOCKET_ERROR
  - 关闭socket,用closesocket,而不是close
  - Windows下不能用write,read进行收发
  - Windows没有socklen_t类型,直接采用int

WinSock的使用

- Socket不是采用int表示,而是采用SOCKET来表示
- 必须包含Winsock.h的头文件
- 必须链接 ws2_32.lib
- 在使用任何一个SOCKET函数之前,必须要初始化.
  - 只需初始化一次即可

```
do{
WORD wVersionRequested = MAKEWORD(1,1);
WSADATA wsaData;
int nRet;
nRet = WSAStartup(wVersionRequested, &wsaData);
      if (wsaData.wVersion != wVersionRequested)
      {
            fprintf(stderr,"\n Wrong version\n");
            return;
      }
}while(0);
```

WinSock的使用

- 关闭SOCKET用closesocket()而不是close
- 当不需要SOCKET函数时,需要调用一次清除函数.
  - 但不是必须,程序退出时,也会自动调用
  - WSACleanup();
- Socket函数的错误的返回值,都不是-1,而是SOCKET_ERROR
- 创建Socket失败的返回值,不是-1,是INVALID_SOCKET

特殊情况处理

设置和获得套接口选项

- 获得套接口选项
  - int getsockopt (int sockfd, int level, int optname, void * optval, socklen_t *opteln)
- 设置套接口选项：
  - int setsockopt (int sockfd, int level, int optname, const void * optval, socklen_t *opteln)
- 参数含意
  - sockfd(套接字): 指向一个打开的套接口描述字
  - level:(级别)： 指定选项代码的类型。
    - SOL_SOCKET: 基本套接口
    - IPPROTO_IP: IPv4套接口
    - IPPROTO_IPV6: IPv6套接口
    - IPPROTO_TCP: TCP套接口
  - optname(选项名)： 选项名称
  - optval(选项值): 是一个指向变量的指针 类型：整形，套接口结构， 其他结构类型:linger{}, timeval{ }
  - optlen(选项长度) ：optval 的大小
- 返回值：标志打开或关闭某个特征的二进制选项

- 用于设置SOCKET细节
  SO_REUSEADDR 重用地址
- 当打开某一端口的程序非正常退出,可能端口仍被占用,第二次执行程序就会报"Addr in use"无法使用这一端口
- 用SO_REUSEADDR 可以防止这一问题.服务器的socket,最好用这一选项.

```
n = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
val = 1;
setsockopt(n, SOL_SOCKET, SO_REUSEADDR, (char *) &val, sizeof (val));
// some code ...
if ((bind(n, (struct sockaddr *) &sin, sizeof (sin)) < 0)
    || (listen(n, QLEN) < 0))
    exit(1);
```

SO_BROADCAST UDP广播选项

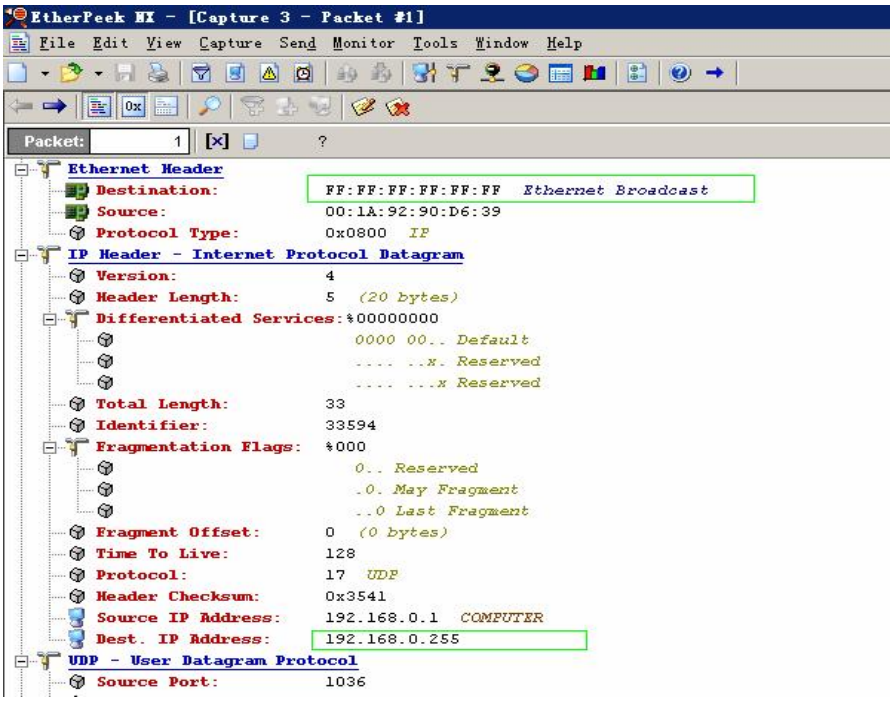
- 一般在发送UDP数据报的时候，希望该socket发送的数据具有广播特性
- 用sendto发送时,广播地址可以写
  - from.sin_addr.s_addr=INADDR_BROADCAST;
  - 或是根据**IP**地址和掩码算出的子网的广播地址

```
int bBroadcast=1;
setsockopt(s，SOL_SOCKET，SO_BROADCAST，(const char*)&bBroadcast，sizeof(int));
```

UDP广播实例

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <errno.h>
#include <stdlib.h>
#include <arpa/inet.h>

int main(int argc, char **argv)
{
struct sockaddr_in s_addr;
int sock;
int addr_len;
int len;
char buff[128];
int yes;
```

```
/*接上一页*/

/* 创建 socket */
if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
perror("socket");
exit(errno);
} else
printf("create socket.\n\r");

/* 设置通讯方式对广播，即本程序发送的一个消息，网络上所有主机均可以收到 */
yes = 1;
setsockopt(sock, SOL_SOCKET, SO_BROADCAST, &yes, sizeof(yes));
/* 唯一变化就是这一点了 */

/* 设置对方地址和端口信息 */
s_addr.sin_family = AF_INET;
if (argv[2])
s_addr.sin_port = htons(atoi(argv[2]));
else
s_addr.sin_port = htons(7838);
//Windows
if (argv[1])
s_addr.sin_addr.s_addr = inet_addr(argv[1]);
else {
printf("消息必须有一个接收者！\n");
exit(0);
}

/* 发送 UDP 消息 */
addr_len = sizeof(s_addr);
strcpy(buff, "hello i'm here");
len = sendto(sock, buff, strlen(buff), 0,
(struct sockaddr *) &s_addr, addr_len);
if (len < 0) {
printf("\n\rsend error.\n\r");
return 3;
}

printf("send success.\n\r");
return 0;
}
```



课堂练习

- 用os_lib库改造tcp echo和udp echo程序,使用其能在两个操作系统下编译,测试通过
- 在os_lib库加入进程锁支持,Linux用有名信号量,Windows用Mutex

- 用getpeername()把聊天程序的广播消息加入IP和端口信息.
- 请将原有简单文件下载服务器,由一次下载一个文件,升级为一次可下多个文件的.
  - 多个文件名可由配置文件或直接写在源码里
- 请将广播和聊天服务器代码移植到Linux上