这是图片

linux C ioctl设置，获取网关，路由信息

过客
每周更新一篇以上关于音视频的技术博文!

linux c ioctl 设置本地ip 子网掩码网络信息

在日常开发中除了设置网络信息外，路由的设置也是不可避免的，同样仍然使用ioctl万能函数设置，获取设备属性，首先认识下路由属性核心结构：

```
struct rtentry {
        unsigned long   rt_pad1;
        struct sockaddr rt_dst;         /* 目标地址*/
        struct sockaddr rt_gateway;     /* 网关*/
        struct sockaddr rt_genmask;     /* 子网掩码*/
        unsigned short  rt_flags;
        short           rt_pad2;
        unsigned long   rt_pad3;
        void            *rt_pad4;
        short           rt_metric;      /* +1 为了二进制兼容性 */
        char __user     *rt_dev;        /* 强制添加设备     */
        unsigned long   rt_mtu;         /* 路由MTU*/
#ifndef __KERNEL__
#define rt_mss  rt_mtu                  /* 兼容性*/
#endif
        unsigned long   rt_window;
        unsigned short  rt_irtt;        /* 最初的RTT*/
};
```

这个结构由SIOCADDRT和SIOCDELRT调用传递，白话就是这个结构就是在添加路由和删除路由的时候被调用。SIOCADDRT：添加路由，SIOCDELRT：删除路由，就是ioctl调用时使用的请求码。

具体的代码实现如下：

下面代码的参数RouteItem结构如下：

```
typedef struct
{
    std::string ethName;//网卡名
    std::string dest_ip;//目的地址
    std::string mask;//子网掩码
    std::string nexthop;//下一跳(网关)
} RouteItem;
```

添加路由：

```
/*
描述:
    添加路由项
参数:
    item:路由项实例
返回值:
    成功:true 失败:false
*/

bool NetHelper::AddRouteItem(RouteItem &item)
{

    int sockfd;
    struct rtentry rt;//创建结构体变量
    //创建套接字
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd == -1)
    {
        perror("socket creation failed\n");
        return false;
    }
    //设置网关，又名下一跳:转到下个路由的路由地址
    struct sockaddr_in *sockinfo = (struct sockaddr_in *)&rt.rt_gateway;
    sockinfo->sin_family = AF_INET;
    sockinfo->sin_addr.s_addr = inet_addr(item.nexthop.c_str());

    //设置目的地址
    sockinfo = (struct sockaddr_in *)&rt.rt_dst;
    sockinfo->sin_family = AF_INET;
    sockinfo->sin_addr.s_addr = inet_addr(item.dest_ip.c_str());

    //设置子网掩码
    sockinfo = (struct sockaddr_in *)&rt.rt_genmask;
    sockinfo->sin_family = AF_INET;
    sockinfo->sin_addr.s_addr = inet_addr(item.mask.c_str());

    //设置网卡设备名
    rt.rt_flags = RTF_UP | RTF_GATEWAY;
    rt.rt_dev = (char *)item.ethName.c_str();

    //ioctl接口进行路由属性设置
    if (ioctl(sockfd, SIOCADDRT, &rt) < 0)
    {
        perror("ioctl:");
        return false;
    }

    return true;
}
```

删除路由代码：

```
/*
描述:
    删除路由项
参数:
    item:路由对象
```

```
返回值:
    成功:true 失败:false
*/

bool NetHelper::DelRouteItem(RouteItem &item)
{
    int fd;
    struct sockaddr_in _sin;
    struct sockaddr_in *sin = &_sin;
    struct rtentry rt;

    fd = socket(AF_INET, SOCK_DGRAM, 0);
    if (fd < 0)
    {
        perror("create fd of socket error:");
        return false;
    }
    // 要删除的网关信息, 网关信息可以不填充, 有ip 子网掩码即可删除路由
    memset(&rt, 0, sizeof(struct rtentry));
    memset(sin, 0, sizeof(struct sockaddr_in));
    sin->sin_family = AF_INET;
    sin->sin_port = 0;
    if (inet_aton(item.nexthop.c_str(), &sin->sin_addr) < 0)
    {
        perror("gateWay inet_aton error:");
        close(fd);
        return false;
    }
    memcpy(&rt.rt_gateway, sin, sizeof(struct sockaddr_in));
    //要删除的ip信息
    ((struct sockaddr_in *)&rt.rt_dst)->sin_family = AF_INET;
    if (inet_aton(item.dest_ip.c_str(), &((struct sockaddr_in *)&rt.rt_dst)->sin_addr
    {
        perror("dest addr inet_aton error:");
        close(fd);
        return false;
    }
    //要删除的子网掩码
    ((struct sockaddr_in *)&rt.rt_genmask)->sin_family = AF_INET;
    if (inet_aton(item.mask.c_str(), &((struct sockaddr_in *)&rt.rt_genmask)->sin_add
    {
        perror("mask inet_aton error:");
        close(fd);
        return false;
    }
    //网卡设备名
    rt.rt_dev = (char *)item.ethName.c_str();
    rt.rt_flags = RTF_UP | RTF_GATEWAY;

    if (ioctl(fd, SIOCDELRT, &rt) < 0)
    {
        perror("ioctl SIOCADDRT error : ");
        close(fd);
        return false;
    }

    close(fd);

    return true;
}
```

路由表用到的指令吗目前只有这两个:

```
/* Routing table calls. */
#define SIOCADDRT       0x890B          /* 添加路由      */
#define SIOCDELRT       0x890C          /* 删除路由      */
```

所以目前还没办法通过ioctl获取路由表信息，一般我们都是通过/proc/net/route文件获取路由表信息，具体路由表的获取方法:

```
/*
描述:
    获得路由表
参数:
    routeTable:保存路由表的容器
返回值:
    成功:true 失败:false
*/

bool NetHelper::GetRouteTabel(RouteTable &routeTable)
{
    // ubuntu
    FILE *fp;
    char devname[64];
    unsigned long d, g, m;
    int r = 0;
    int flgs, ref, use, metric, mtu, win, ir;
    uint8_t gate[4] = {0};
    uint8_t ip[4] = {0};
    uint8_t mask[4] = {0};
    std::string gateway;
    uint8_t ip_str[15] = {0};
    uint8_t mask_str[15] = {0};
    uint8_t gate_str[15] = {0};
    RouteItem item;

    fp = fopen("/proc/net/route", "r");
    /* Skip the first line. */
    r = fscanf(fp, "%*[^\n]\n");
    if (r < 0)
    {
        /* Empty line, read error, or EOF. Yes, if routing table
         * is completely empty, /proc/net/route has no header.
         */
        fclose(fp);
        return false;
    }

    // ubuntu
    while (1)
    {
        r = fscanf(fp, "%63s%lx%lx%X%d%d%d%lx%d%d%d\n",
                   devname, &d, &g, &flgs, &ref, &use, &metric, &m,
                   &mtu, &win, &ir);

        if (r != 11)
        {
            if ((r < 0) && feof(fp))
            { /* EOF with no (nonspace) chars read. */
```

```
                break;
            }
        }
        //        printf("%63s %lx %lx %X %d %d %d %lx %d %d %d\n", devname, d, g, fl
        // RTF_UP表示该路由可用，RTF_GATEWAY表示该路由为一个网关，组合在一起就是3，表示一个可用
        if ((flgs & RTF_GATEWAY) &&
            (flgs & RTF_UP) &&
            g != 0)
        {
            memcpy(ip, &d, 4);
            memcpy(mask, &m, 4);
            memcpy(gate, &g, 4);
            sprintf((char *)gate_str, "%d.%d.%d.%d", gate[0], gate[1], gate[2], gate[
            sprintf((char *)ip_str, "%d.%d.%d.%d", ip[0], ip[1], ip[2], ip[3]);
            sprintf((char *)mask_str, "%d.%d.%d.%d", mask[0], mask[1], mask[2], mask[
            item.ethName = devname;
            item.dest_ip = (char *)ip_str;
            item.mask = (char *)mask_str;
            item.nexthop = (char *)gate_str;
            routeTable.push_back(item);
        }
    }

    printf("gate : %d.%d.%d.%d\n", gate[0], gate[1], gate[2], gate[3]);
    printf("ip : %d.%d.%d.%d\n", ip[0], ip[1], ip[2], ip[3]);
    printf("mask : %d.%d.%d.%d\n", mask[0], mask[1], mask[2], mask[3]);

    std::vector<RouteItem>::iterator iter;
    for (iter = routeTable.begin(); iter != routeTable.end(); iter++)
    {
        std::cout << iter->ethName << "  " << iter->dest_ip << "  " << iter->mask <<
    }

    fclose(fp);
    return true;
}
```

该代码获取了路由表的所有ip 子网掩码，网关（下一跳），并且将获取到的信息保存在RouteTable容器中，想要获取路由信息可以参考该代码。

温馨提示：路由设置时ip不最后一个字节要为0才能设置成功，及ip：XXX.XXX.XXX.0 。

路由类似于一个转换器，能够连通不同网段的IP，所以最后一个字节无关紧要，为0即可。

发布于 2023-04-02 20:56 · IP 属地北京

Linux 网关 路由

写下你的评论...

还没有评论，发表第一个评论吧

文章被以下专栏收录

嵌入式linux
linux的技术积累

推荐阅读

在 Linux 上检查网络连接的更多方法

Linux... 发表于Linux...

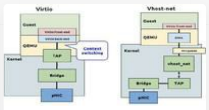
linux进程间通信----IPC篇(一)----共享内存初识篇

先给自己打个广告，本人的微信公众号正式上线了，搜索:张笑生的地盘，主要关注嵌入式软件开发，股票基金定投，足球等等，希望大家多多关注，有问题可以直接留言给我，一定尽心尽力回答大家...

ryan

linux内核与用户之间的通信方式——虚拟文件系统、...

CPP加油... 发表于Linux...



简单干货分享：Linux内核中网络设备连接状态监测

Linux嵌入式

×
登录即可查看 超5亿 专业优质内容
超 5 千万创作者的优质提问、专业回答、深度文章和精彩视频尽在知乎。
立即登录/注册

赞同 1 ▼ 添加评论 分享 喜欢 收藏 申请转载 ···