1. 简述Linux删除文件的原理

Linux是通过link的数量来控制文件删除的，只有当一个文件不存在任何link的时候，这个文件才会被删除。
一般来说，每个文件都有2个link计数器:i_count 和 i_nlink。

i_count的意义是当前文件使用者（或被调用）的数量
当一个文件被某一个进程引用时，对应的这个值就会增加

i_nlink的意义是介质连接的数量（硬链接的数量）
当创建文件的硬链接的时候，这个值就会增加

2. 写出TCP三次握手与四次挥手的过程

三次握手：

初始阶段，客户端是CLOSE状态，服务端需要监听服务所在端口，因此处于LISTEN状态

客户端发来SYN分组，到达了服务器，此时客户端会从CLOSE状态立即变为SYN-SENT状态

SYN到达服务端，在服务器内核中，会将"根据SYN分组内容创建的内核数据结构实例"放入SYN队列中，同时会发送一个SYN+ACK数据分组给客户端，此时服务端从LISTEN状态变为SYN-RECEIVED状态

客户端收到SYN+ACK分组，会给服务端发送ACK分组，并从SYN-SENT状态变为ESTABLISTED状态

服务端收到ACK分组后会从SYN-RECEIVED状态转换为ESTABLISHED状态，但实际在内核中，会把之前放入SYN队列中的数据结构实例移出，放入ACCEPT队列中，然后由应用程序调用accept方法从ACCEPT队列中将该数据结构实例取出，并返回一个新的文件描述符，供应用程序进行后续数据处理和通信。该文件描述符对应的是一个新的 socket，通过它，应用程序可以继续与客户端进行数据收发。

四次挥手：

当客户端决定关闭连接时，它会发送一个 FIN（Finish）包给服务器，表示它不再有数据要发送。发送 FIN 包后，客户端的连接状态从 ESTABLISHED 变为 FIN-WAIT-1，等待服务器的确认。

服务器收到客户端的 FIN 包后，向客户端发送一个 ACK 包，确认收到 FIN。发送 ACK 后，服务器的状态从 ESTABLISHED 变为 CLOSE-WAIT，表示它知道客户端已停止发送数据，但服务器可能还有数据要发送。

当服务器确认不再有数据需要发送时，它会向客户端发送一个 FIN 包，请求关闭连接。发送 FIN 后，服务器的状态从 CLOSE-WAIT 变为 LAST-ACK，等待客户端的最后确认。

客户端收到服务器的 FIN 包后，向服务器发送一个 ACK 包 进行确认

发送完 ACK 包后，客户端的状态变为 TIME-WAIT，并进入一个计时等待阶段（2*MSL），以确保服务器能够收到这个 ACK。
如果在 TIME-WAIT 定时器超时后未收到服务器的重传请求，客户端最终进入 CLOSED 状态，正式关闭连接。

3. 写出ARP协议的作用与原理

作用：ARP（地址解析协议）的主要作用是将网络层地址（如IP地址）转换为数据链路层地址（如MAC地址）。

原理：
显示发送一个包含有IP地址的请求广播给局域网的所有主机，对应IP的主机会返回一个数据包给发送请求的主机，该数据包中含有这个主机的MAC地址，交换机上记载着MAC地址对应的交换机接口，从而实现物理地址的寻址

4. 编写个shell脚本将/usr/local/test目录下大于100K的文件转移到/tmp目录

```
#!/bin/bash

# 源目录
SOURCE_DIR="/usr/local/test"
# 目标目录
TARGET_DIR="/tmp"

# 检查源目录是否存在
if [ ! -d "$SOURCE_DIR" ]; then
    echo "源目录 $SOURCE_DIR 不存在！"
    exit 1
fi

# 确保目标目录存在
if [ ! -d "$TARGET_DIR" ]; then
    mkdir -p "$TARGET_DIR"
    echo "目标目录 $TARGET_DIR 已创建。"
fi

# 查找大于100K的文件并移动到目标目录
find "$SOURCE_DIR" -type f -size +100k -exec mv {} "$TARGET_DIR" \;

echo "转移完成！"
```

5. 使用Linux系统命令统计出establish状态的连接数有多少

```
netstat -an | grep ESTABLISHED | wc -l

ss -t state established | wc -l
```

5. 简述TCP和UDP的优缺点和各自的使用场景

TCP（传输控制协议）
优点
可靠性：TCP提供可靠的数据传输，确保数据包按顺序到达，并且没有丢失或损坏。
流量控制：TCP使用滑动窗口机制来控制数据流，防止发送方发送数据过快导致接收方无法处理。
拥塞控制：TCP能够检测网络拥塞并调整发送速率，以避免网络过载。
连接导向：TCP是面向连接的协议，通信双方在传输数据之前需要建立连接。
缺点
速度较慢：由于TCP需要进行握手、确认、重传等操作，因此传输速度相对较慢。
开销较大：TCP的头部开销较大，包含了许多控制信息。
复杂性：TCP的实现较为复杂，涉及多种机制如拥塞控制、流量控制等。
使用场景

文件传输：如FTP（文件传输协议）。
电子邮件：如SMTP（简单邮件传输协议）。
网页浏览：如HTTP/HTTPS。
远程登录：如SSH（安全外壳协议）。
数据库操作：如SQL查询。

UDP（用户数据报协议）
优点
速度快：UDP没有连接建立、确认、重传等机制，因此传输速度较快。
开销小：UDP的头部开销较小，只包含基本的源端口、目的端口、长度和校验和。
简单性：UDP的实现较为简单，没有复杂的控制机制。
无连接：UDP是无连接的协议，不需要建立连接即可发送数据。
缺点
不可靠：UDP不保证数据包的可靠传输，数据包可能会丢失、重复或乱序。
无流量控制：UDP没有流量控制机制，发送方可以以任意速率发送数据。
无拥塞控制：UDP没有拥塞控制机制，可能会导致网络拥塞。
使用场景
实时通信：如VoIP（网络电话）、视频会议。
在线游戏：需要低延迟的实时交互。
DNS查询：域名系统查询通常使用UDP。
广播和多播：UDP支持广播和多播，适合一次性向多个接收者发送数据。
流媒体：如视频流和音频流，允许一定程度的丢包。

5. 以下是 6 个 IP 地址及其子网掩码：

- 192.168.10.5/23
- 192.168.11.10/24
- 192.168.10.130/25
- 192.168.11.128/26
- 192.168.10.70/22
- 192.168.11.200/30
  要求：找出哪些 IP 地址属于同一个网段。公共网段是什么？哪些IP地址是独立IP

```
# 网络范围判断
92.168.10.5/23

11000000.10101000.00001010.00000101   ------  192.168.10.5
11111111.11111111.11111110.00000000  --------    /23
--------------------------------------------------------------------------
11000000.10101000.00001010.00000000  --------   192.168.10.0  -------网络地址范围
（192.168.10.0~192.168.11.255）


192.168.11.10/24

11000000.10101000.00001011.00001010  ------- 192.168.11.10
11111111.11111111.11111111.00000000  -------  /24
--------------------------------------------------------------------------
11000000.10101000.00001011.00000000 ------  192.168.11.0  ----------- 网络地址范围(
198.168.11.0 ~ 192.168.11.255)
```

192.168.10.130/25

11000000.10101000.00001010.10000010 --------- 192.168.10.130
11111111.11111111.11111111.10000000 --------- /25

11000000.10101000.00001010.10000000 --------- 192.168.10.128 网络地址范围(192.168.10.128~182.168.10.255)

192.168.11.128/26

11000000.10101000.00001011.10000000 -------- 192.168.11.128
11111111.11111111.11111111.11000000 --------   /26

11000000.10101000.00001011.10000000 -------- 192.168.11.128

192.168.10.70/22

11000000.10101000.00001010.01000110 ----- 192.168.10.70
11111111.11111111.11111100.00000000 ----- /22

11000000.10101000.00001010.00000000 ------ 192.168.10.0 网络地址范围(192.168.10.128~182.168.10.255)

192.168.11.200/30
11000000.10101000.00001011.11001000 ------- 192.168.11.200
11111111111111111111111111111100 ------- /30

11000000.10101000.00001011.11001000 ------- 192.168.11.200
通过比较范围，以下 IP 地址属于同一个网段：

192.168.10.5/23 和 192.168.10.70/22
公共网段：192.168.8.0 ~ 192.168.11.255

192.168.11.10/24 和 192.168.10.5/23（在 192.168.10.0/23 的范围中）
公共网段：192.168.10.0 ~ 192.168.11.255

192.168.10.130/25 不与其他范围重叠。

192.168.11.128/26 和 192.168.11.10/24。
公共网段：192.168.11.0 ~ 192.168.11.255

192.168.11.200/30 不与其他范围重叠。

最终结论

属于同一个网段的 IP 地址有：

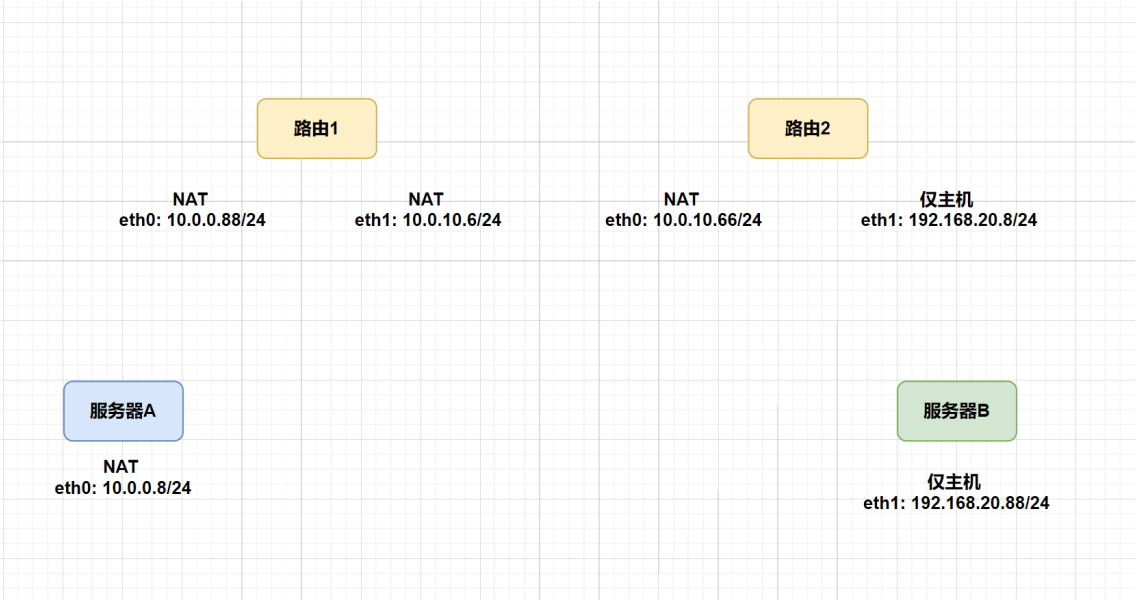
192.168.10.5/23、192.168.11.10/24 和 192.168.10.70/22
公共网段：192.168.10.0/23

192.168.11.10/24 和 192.168.11.128/26
公共网段：192.168.11.0/24。

```
独立的 IP 地址:
192.168.10.130/25
192.168.11.200/30
```

8. 写一个初始化服务器的脚本

- 要求:(Rocky8以上和Ubuntu2204以上通用)
  - 关闭selinux
  - 关闭防火墙
  - 初始化网卡名eth0
  - 给服务器配置相应的国内软件源(阿里，清华，中科大等随意选择)
- 使用这个初始化脚本给四台服务器初始化，按下列图例配置软路由使A服务器和B服务器可以成功通信




```
#!/bin/bash

. /etc/os-release

eth_num=$[ `ip a|grep -P "^\d:" |wc -l` - 1 ]

# 关闭SELinux
rocky_selinuxoff() {
        sed -ir 's@SELINUX=enforcing@SELINUX=disabled@' /etc/selinux/config
}

# 关闭防火墙
ubuntu_firewalloff() {
        systemctl disable --now ufw
}

rocky_firewalldoff() {
        systemctl disable --now firewalld.service
}


# 初始化网卡
```

```
# rocky9添加udev/file

rocky9_udevfile() {


        for (( i=0; i<$eth_num; i++ )); do
                j=$[ $i + 2 ]
                mac=`ip a|grep -A1 "$j:"|tail -n1|awk '{print $2}'`
                echo "SUBSYSTEM==\"net\", ACTION==\"add\",
ATTR{address}==\"$mac\", NAME=\"eth$i\"" >> /etc/udev/rules.d/10-network.rules
        done

}


# rocky8,9更改grub

rocky_grub() {

        sed -ri 's@(^GRUB_CMDLINE.*)\"@\1 net.ifnames=0 biosdevname=0\"@'
/etc/default/grub
        grub2-mkconfig -o /boot/grub2/grub.cfg

}


Ubuntu_grub() {

        sed -i.bak '/^GRUB_CMDLINE_LINUX=/s#"$#net.ifnames=0"#' /etc/default/grub

}


# 换源
ubuntu2404_repo() {
    cat > /etc/apt/sources.list << EOF
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ noble main restricted universe
multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ noble main restricted
universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ noble-updates main restricted
universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ noble-updates main
restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ noble-backports main restricted
universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ noble-backports main
restricted universe multiverse
EOF

}

ubuntu2204_repo() {
    cat > /etc/apt/sources.list << EOF
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe
multiverse
```

```
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted
universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted
universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main
restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted
universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main
restricted universe multiverse
EOF
}

rocky9_repo() {
    mkdir /etc/yum.repos.d/backup -p
    mv /etc/yum.repos.d/*.repo /etc/yum.repos.d/backup/
    cat > /etc/yum.repos.d/BaseOS.repo << EOF
[BaseOS-ali]
name=BaseOS-ali-repo
baseurl=https://mirrors.aliyun.com/rockylinux/9/BaseOS/x86_64/os/
enabled=1
gpgcheck=0
EOF

    cat > /etc/yum.repos.d/Appstream.repo << EOF
[AppStream-ali]
name=AppStream-ali-repo
baseurl=https://mirrors.aliyun.com/rockylinux/9/AppStream/x86_64/os/
enabled=1
gpgcheck=0
EOF

    cat > /etc/yum.repos.d/extras.repo << EOF
[extras-ali]
name=extras-ali-repo
baseurl=https://mirrors.aliyun.com/rockylinux/9/extras/x86_64/os/
enabled=1
gpgcheck=0
EOF

    cat > /etc/yum.repos.d/epel.repo << EOF
[epel-ali]
name=epel-ali-repo
baseurl=https://mirrors.aliyun.com/epel/9/Everything/x86_64/
enabled=1
gpgcheck=0
EOF

}


rocky8_repo() {
    mkdir /etc/yum.repos.d/backup -p
    mv /etc/yum.repos.d/*.repo /etc/yum.repos.d/backup/
    cat > /etc/yum.repos.d/BaseOS.repo << EOF
[BaseOS-ali]
```

```
name=BaseOS-ali-repo
baseurl=https://mirrors.aliyun.com/rockylinux/8/BaseOS/x86_64/os/
enabled=1
gpgcheck=0
EOF

    cat > /etc/yum.repos.d/Appstream.repo << EOF
[AppStream-ali]
name=AppStream-ali-repo
baseurl=https://mirrors.aliyun.com/rockylinux/8/AppStream/x86_64/os/
enabled=1
gpgcheck=0
EOF

    cat > /etc/yum.repos.d/extras.repo << EOF
[extras-ali]
name=extras-ali-repo
baseurl=https://mirrors.aliyun.com/rockylinux/8/extras/x86_64/os/
enabled=1
gpgcheck=0
EOF

    cat > /etc/yum.repos.d/epel.repo << EOF
[epel-ali]
name=epel-ali-repo
baseurl=https://mirrors.aliyun.com/epel/8/Everything/x86_64/
enabled=1
gpgcheck=0
EOF

}


echo "初始化过程中...."


if [[ "$NAME" == Rocky* ]];then

        rocky_selinuxoff
        rocky_firewalldoff

        if [[ $VERSION_ID == 8* ]];then
        # 网卡初始化
            rocky_grub
            rocky8_repo
            yum makecache
        fi
        if [[ $VERSION_ID == 9* ]];then
        # 网卡初始化
            rocky_grub
            rocky9_udevfile
            rocky9_repo
            yum makecache
        fi
fi
```

```
if [[ "$NAME" == Ubuntu* ]];then

        ubuntu_firewalloff
        Ubuntu_grub

        if [[ $VERSION_ID == 22* ]];then
            ubuntu2204_repo
            apt update
        fi
        if [[ $VERSION_ID == 24* ]];then
            ubuntu2404_repo
            apt update
        fi
fi


# 重启
reboot
```

```
10.0.0.8服务器：
route add -net 192.168.20.0/24 gw 10.0.0.88 dev eth0

route add -net 192.168.20.0/24 gw 10.0.10.66 dev eth1

route add -net 10.0.0.0/24 gw 10.0.10.6 dev eth0

route add -net 10.0.0.0/24 gw 192.168.20.0 dev eth1

echo 1 > /proc/sys/net/ipv4/ip_forward

echo 1 > /proc/sys/net/ipv4/ip_forward
```