1. 叙述Https的工作过程

```
1. 第一步：客户端发起请求
2. 第二步：服务端将证书传给客户端
3. 第三步：客户端使用CA公钥解析验证证书,没问题的话生成一个随机值,使用服务器公钥加密随机值发给服务端
4. 第四步：服务端收到公钥加密的随机值,使用自己的私钥解密,后续客户端和服务端使用这个随机值进行对称加密，机密传输。
```

2. 实验题：创建自签CA，并为用户tom颁发证书

```
# 并更改配置并创建目录
mkdir -pv /etc/pki/CA/{certs,crl,newcerts,private}
touch /etc/pki/CA/index.txt
echo 01 > /etc/pki/CA/serial

vim /etc/ssl/openssl.cnf
[ ca_default ]
dir = /etc/pki/CA
# 创建自签CA
(umask 077; openssl genrsa -out /etc/pki/CA/private/cakey.pem 2048)
openssl req -new -x509 -key /etc/pki/CA/private/cakey.pem -days 3650 -out
/etc/pki/CA/cacert.pem

# 查看自签证书
openssl x509 -in /etc/pki/CA/cacert.pam -noout -text
# tom递交申请
openssl genrsa -out /data/tom.key 2048
openssl req -new -key /data/tom.key -out /data/tom.csr
# CA为tom颁发证书
openssl ca -in /data/tom.csr -out /etc/pki/CA/certs/tom.crt -days 1000
```

3. SCP和Rsync的区别是什么

```
SCP:
使用SSH协议
全量同步
不支持校验文件完整性

Rsync
SSH或daemon
增量同步
支持校验文件完整性
```

4. 简述NFS实现原理

```
NFS 的操作流程
```

挂载过程：
客户端发送挂载请求到服务端的 mountd 守护进程。
服务端验证客户端权限，返回挂载成功的信息。
客户端挂载远程目录到本地目录，更新本地 VFS

文件访问过程：
客户端对挂载目录的文件操作通过 VFS 转发到 NFS 客户端模块。
NFS 客户端模块生成对应的 NFS 协议请求，通过 RPC 发送到服务端。
服务端的 NFS 守护进程解析请求并调用底层文件系统操作。
文件系统返回操作结果，服务端通过 NFS 响应将结果返回客户端。

数据传输：
文件的读写数据通过网络传输，客户端会缓存部分数据以减少网络访问。

5. 写出Web浏览器发起HTTP请求访问网站的过程

- 首先服务器监听打开了443或者80端口
- 浏览器从url中解析出域名
- 根据域名查询DNS，获取域名对应得IP地址
- 浏览器根据ip地址，和服务器三次握手建立TCP链接，https会额外完成TLS/SSL的握手
- 构造HTTP请求，在构造请求的过程中，填充相应的HTTP头部，包括上下文所需要的信息，至头部中
- 通过链接发起HTTP请求
- 服务器接收到HTTP请求后，完成资源的表述，把客户端请求的文件如html页面作为包体返回给浏览器
- 浏览器在渲染引擎中解析响应，根据这个响应中一些其他的超链接资源去构造其他HTTP请求

6. Sessin是什么？Cookie是什么？Session和Cookie有什么关系？

session的定义：
1. Session 是服务器端维护用户状态的一种机制。
2. 每个用户访问服务器时，服务器会为用户创建一个唯一的 Session ID（通常是随机生成的 UUID），用于标识用户会话。

3.Session 是一个数据集合，通常存储在服务器的内存中或某种存储介质（如 Redis 或数据库）中，用于保存用户的临时会话数据，例如：

- 用户登录状态
- 用户偏好设置
- 购物车信息

cookie的定义
1. Cookie 是存储在客户端（浏览器）上的一小段数据。
2. 在基于 Session 的会话中，Cookie 通常用来存储 Session ID，并在后续请求中将这个 ID 发送回服务器，以便服务器识别用户会话。

cookie和session的关系
- Cookie 是 Session 的载体
    - Cookie 中通常包含一个字段（如 sessionid），用来携带 Session ID。
    - 服务器根据这个 Session ID 检索到与之对应的 Session 数据

7. 实验题：某企业有一台服务器 `ServerA`，该服务器上有一个本地目录 `/mnt/nfsdata`，此目录通过 NFS 挂载了远程服务器 `ServerNFS` 上的目录 `/data/shared`。企业需要对 `ServerA` 上的 `/mnt/nfsdata` 中的数据进行实时备份。为此，企业计划在另一台服务器 `ServerB` 上部署 `sersync2`，以监控和同步 `ServerA` 上的 `/mnt/nfsdata` 数据至 `ServerB` 的 `/backup/data` 目录。

```
#ServerNFS: 10.0.0.104
#ServerA: 10.0.0.102
#Backup: 10.0.0.103

#三天服务器关闭防火墙
systemctl --disable now ufw

# ServerNFS
apt install nfs-kernel-server

mkdir /data/shared -p

useradd -u 1111 -s /usr/sbin/nologin -M nfsuser

chown nfsuser:nfsuser /data/shared/ -R

vim /etc/exports
# 添加下行
/data/shared    10.0.0.102(rw,anongid=1111,anonuid=1111

exportfs -r

# Backup
apt install -y rnync

vim /etc/rsyncd.conf

uid=root
gid=root
max connection=0
log file=/var/log/rsyncd.log
pid file=/var/run/rsyncd.pid
lock file=/var/run/rsyncd.lock

[dir1]
path=/backup/data
comment=rsync dir1
read only=no
auth users=rsyncer
secrets file=/etc/rsyncd.pwd

vim /etc/rsyncd.pwd
rsyncer:123456

chmod 600 /etc/rsyncd.pwd

systemctl restart rsync

# ServerA
```

```
# 挂载NFS磁盘
mount 10.0.0.104:/data/shared /mnt/nfsdata

# 下载并部署sersync2
wget https://storage.googleapis.com/google-code-archive-
downloads/v2/code.google.com/sersync/sersync2.5.4_64bit_binary_stable_final.tar.g
z

tar xf sersync2.5.4_64bit_binary_stable_final.tar.gz

vim GNU-Linux-x86/confxml.xml

# 修改下面几行
<sersync>
            <localpath watch="/mnt/nfsdata/">
            <remote ip="10.0.0.103" name="dir1"/>
            <!--<remote ip="192.168.8.39" name="tongbu"/>-->
            <!--<remote ip="192.168.8.40" name="tongbu"/>-->
        </localpath>
        <rsync>
            <commonParams params="-artuz"/>
            <auth start="true" users="rsyncer" passwordfile="/etc/rsyncd.pwd"/>
            <userDefinedPort start="false" port="874"/><!-- port=874 -->
            <timeout start="false" time="100"/><!-- timeout=100 -->
            <ssh start="false"/>
        </rsync>
        <failLog path="/tmp/rsync_fail_log.sh" timeToExecute="60"/><!--default
every 60mins execute once-->
        <crontab start="false" schedule="600"><!--600mins-->
            <crontabfilter start="false">
                <exclude expression="*.php"></exclude>
                <exclude expression="info/*"></exclude>
            </crontabfilter>
        </crontab>
        <plugin start="false" name="command"/>
    </sersync>

echo 123456 > /etc/rsyncd.pwd

chmod 600 /etc/rsyncd.pwd

cd  GNU-Linux-x86

./sersync2 -dro ./confxml

# 测试
在ServerA /mnt/nfsdata目录下创建文件，观察NFSServer上的/data/shared和Backup上
的/backup/data上是否都有文件
```

8. 某企业希望在其 IT 基础设施中实现集中式日志管理，以便及时发现和排查问题。公司有两台服务器：
    - **ServerA**：应用服务器，负责运行业务应用，生成日志。

- **ServerB**：日志服务器，用于接收和存储来自多台应用服务器的日志。

企业要求通过 **rsyslog** 实现以下功能：

1. **ServerA** 将其应用日志 `/var/log/app.log` 和系统日志（如 `auth.log`）实时转发到 **ServerB**。
2. ServerB

   需要对接收到的日志进行分类存储：

   - 系统日志保存到 `/var/log/remote/system_logs.log`。
   - 应用日志保存到 `/var/log/remote/app_logs.log`。
3. 配置日志压缩与自动清理机制，确保日志存储空间不被耗尽：
   - 每天转储一次
   - 做多保存7个压缩日志

提示：可以使用下面的命令在ServerA上进行验证

```
logger -t app "This is a test message for app"
logger -t auth "This is a test message for auth"
```

```
# 服务器准备
ServerA: 10.0.0.211(Rocky)
ServerB: 10.0.0.231(Ubuntu)

# ServerA:
vim /etc/rsyslog.conf

*.* @@10.0.0.231:514

systemctl restart rsyslog

# ServerB:
mkdir -p /var/log/remote
chown syslog:syslog /var/log/remote
chmod 755 /var/log/remote

vim /etc/resyslog.d/remote.conf

:programname, isequal, "app"  /var/log/remote/app_logs.log
:programname, isequal, "auth" /var/log/remote/system_logs.log

vim /etc/rsyslog.conf

# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")

vim /etc/logrotate.d/remote_logs
```

```
/var/log/remote/*.log {
    daily
    rotate 7
    compress
    delaycompress
    missingok
    notifempty
    create 0640 root root
    postrotate
        /usr/bin/systemctl restart rsyslog > /dev/null
    endscript
}

systemctl restart rsyslog
systemctl restart logrotate


# ServerA上测试
logger -t app "This is a test message for app"
logger -t auth "This is a test message for auth"
```