实验1：私有CA证书签发的简单实现

# 实验要求

1、熟悉Openssl工具的使用

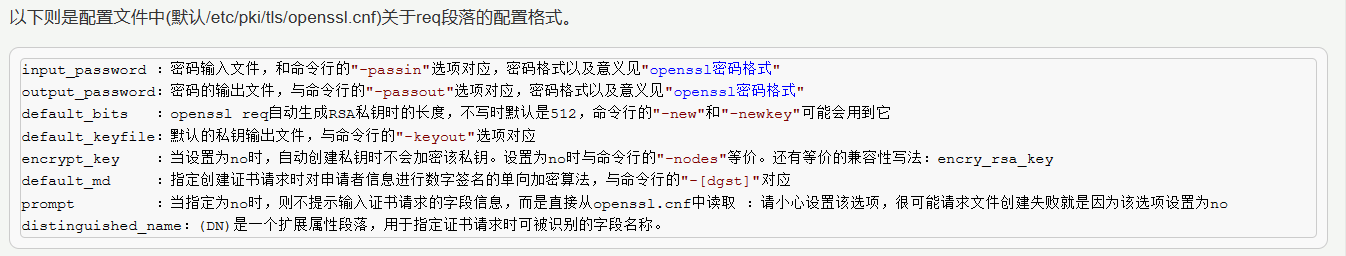
2、搭建私有CA并生成根证书

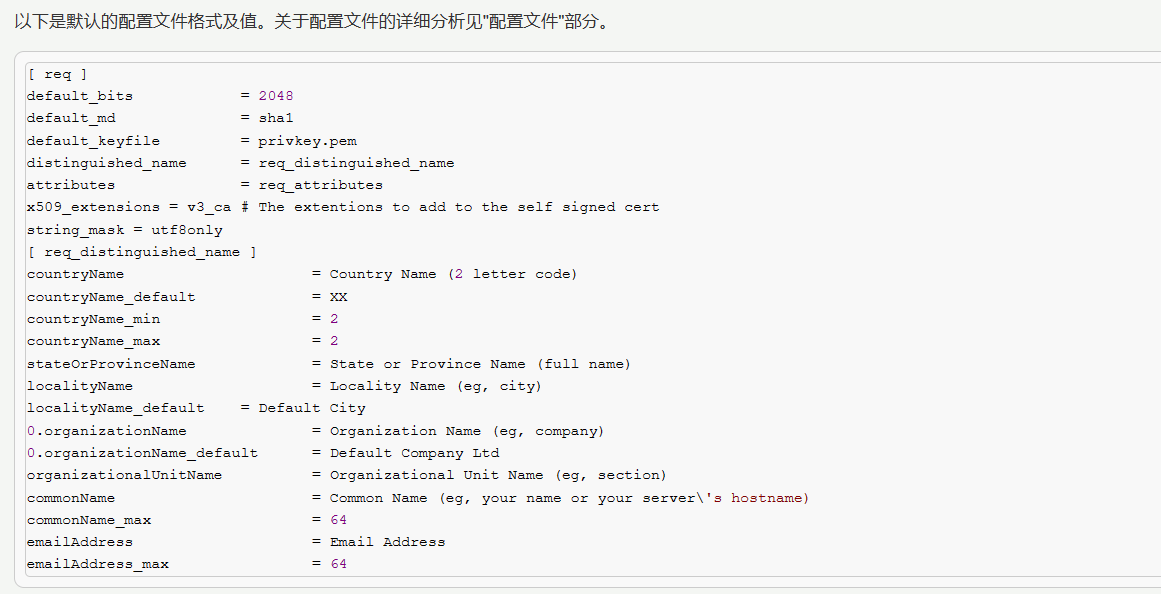
3、完成证书签发、吊销流程的简单实现

# 实验准备

## OpenSSL req







参考网址：

1.[(22条消息) 基于openssl工具完成自建CA以及为server,client颁发证书\_tutu-hu的博客-CSDN博客\_ca.crt server.crt 自建证书](https://blog.csdn.net/weixin_42700740/article/details/117527769)

2.[openssl req(生成证书请求和自建CA) - 骏马金龙 - 博客园 (cnblogs.com)](https://www.cnblogs.com/f-ck-need-u/p/7113610.html)

3.[(22条消息) Linux实现搭建私有CA服务器和证书申请颁发吊销\_白-胖-子的博客-CSDN博客\_linux policy/match](https://blog.csdn.net/timonium/article/details/116150945)

# 实验环境

Ubuntu 20.04+OpenSSL 1.1

# 实验内容

## 1、搭建私有CA

### （1）创建私有CA所需要的文件目录，保存CA的相关信息

mkdir myCA //创建CA根文件夹

cd myCA //进入CA根文件夹

mkdir newcerts private conf //创建三个文件夹，用来存放新发放证书、私钥和配置文件

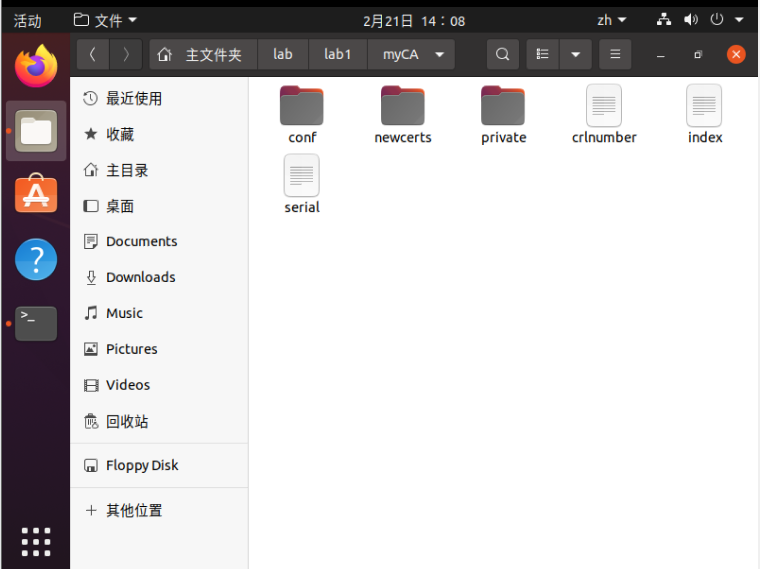
chmod g-rwx,o-rwx private //设置private文件夹的操作权限

touch index crlnumber //创建证书信息数据库、crl编号列表

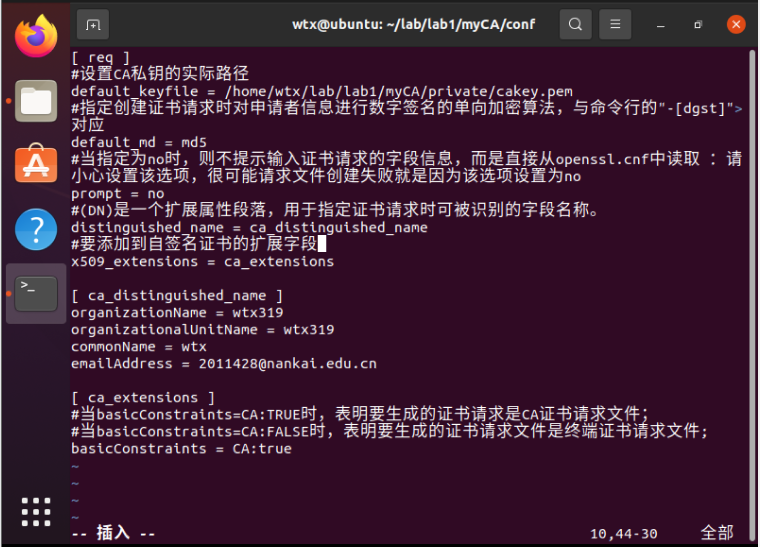
echo 01 > serial //初始化证书的序列号

echo 01 > crlnumber //初始化吊销证书列表序号

结果如下：



### 创建生成CA自签名证书的配置文件

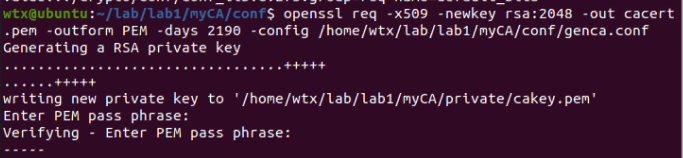


### （3）生成私有CA的私钥和自签名证书（根证书）

openssl req -x509 -newkey rsa:2048 -out /home/wtx/lab/lab1/myCA/newcarts/cacert.pem -outform PEM -days 2190 -config /home/wtx/lab/lab1/myCA/conf/genca.conf

//生成x509的CA证书，过程中需要输入CA私钥的保护密码（cakey），请牢记。

//CA会按照genca.conf文件中配置的规则自签名生成证书



在newcarts文件夹下出现了cacert.pem文件。



## 2、私有CA为服务器签发证书

### （1）创建用来为其他请求签发证书的配置文件

reca.conf

####################################

[ ca ]

default\_ca = testca # 按需修改

[ testca ] # 按需修改

dir = /home/wtx/lab/lab1/myCA # top dir，按实际给出

database = $dir/index # index file.

new\_certs\_dir = $dir/newcerts # new certs dir

certificate = $dir/private/cacert.pem # The CA cert

serial = $dir/serial # serial no file

private\_key = $dir/private/cakey.pem # CA private key

RANDFILE = $dir/private/.rand # random number file

default\_days = 365 # how long to certify for

default\_crl\_days= 30 # how long before next CRL

default\_md = md5 # message digest method to use

unique\_subject = no # Set to 'no' to allow creation of

# several ctificates with same subject.

policy = policy\_any # default policy

[ policy\_any ]

countryName = optional

stateOrProvinceName = optional

localityName = optional

organizationName = optional

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

########################################

### （2）模拟服务器，生成私钥与证书申请的请求文件

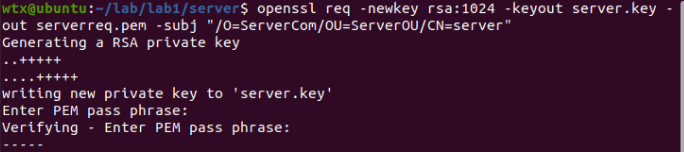
### （3）CA根据服务器的证书请求文件生成证书并将其返回给服务器

在与myCA同级，创建文件夹server。

生成server的私钥server.key及证书申请的请求文件serverreq.pem：

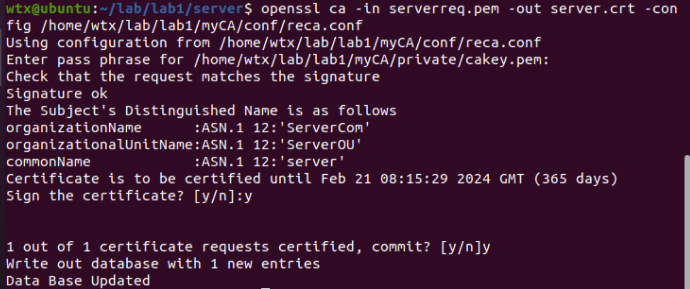
openssl req -newkey rsa:1024 -keyout server.key -out serverreq.pem -subj "/O=ServerCom/OU=ServerOU/CN=server"

私钥为：server

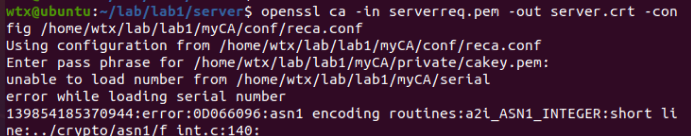


提交serverreq.pem向CA申请证书并生成证书server.crt：

openssl ca -in serverreq.pem -out server.crt -config /home/wtx/lab/lab1/myCA/conf/reca.conf

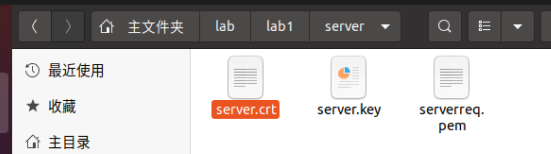


此时遇到报错：

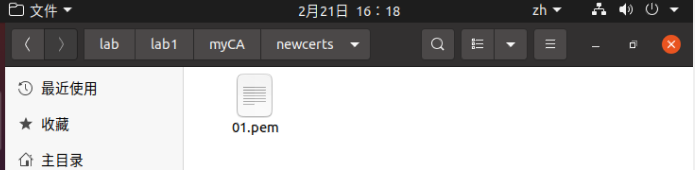


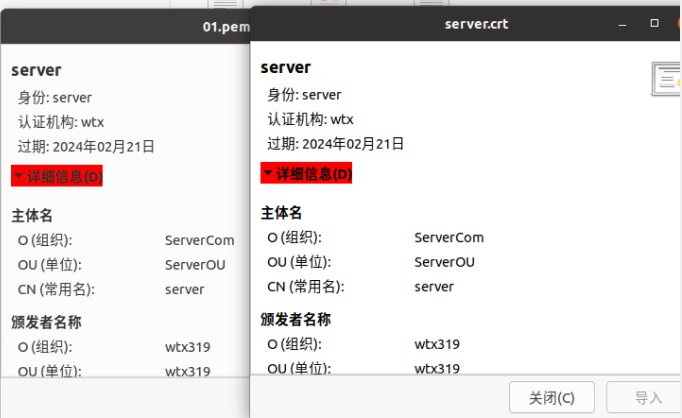
解决方案：echo "01" > /home/wtx/lab/lab1/myCA/serial

结果：



同时，在CA目录下newcerts目录下也生成了该证书的备份：01.pem，这和备份的内容和生成的证书server.crt完全一致。



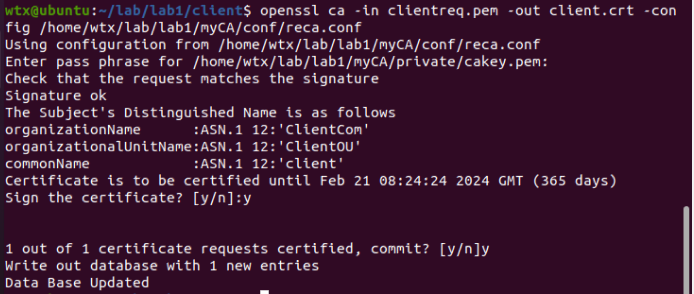


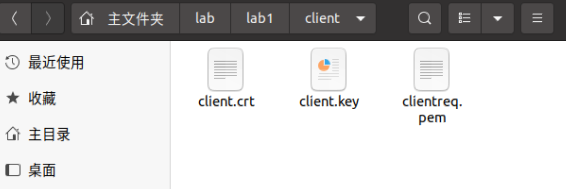
## 私有CA为客户端签发证书

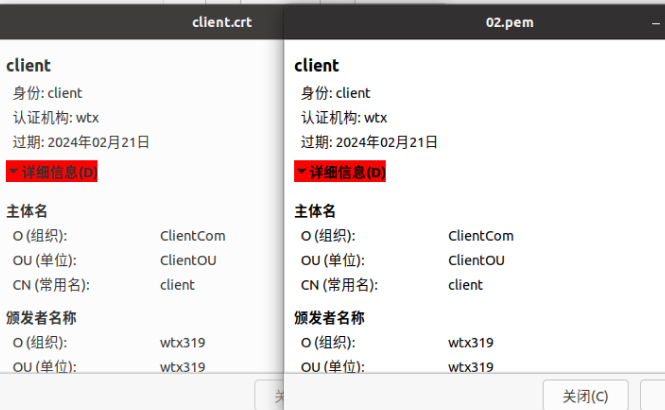
整体步骤同server类似。

openssl req -newkey rsa:1024 -keyout client.key -out clientreq.pem -subj "/O=ClientCom/OU=ClientOU/CN=client"（密钥：client）

openssl ca -in clientreq.pem -out client.crt -config /home/wtx/lab/lab1/myCA/conf/reca.conf



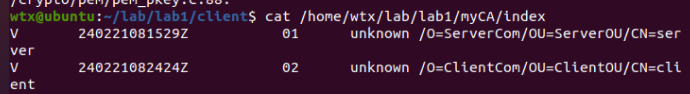




## 4、CA吊销用户证书

### （1）找到证书对应的编号

cat /home/wtx/lab/lab1/myCA/index



### （2）在存放证书的文件夹下找到编号对应的证书，对其完成吊销

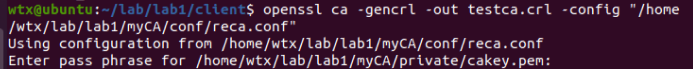
openssl ca -revoke /home/wtx/lab/lab1/myCA/newcerts/02.pem -config "/home/wtx/lab/lab1/myCA/conf/reca.conf"

### 

### 更新吊销证书列表

**生成证书吊销列表文件（CRL）**

openssl ca -gencrl -out testca.crl -config "/home/wtx/lab/lab1/myCA/conf/reca.conf"

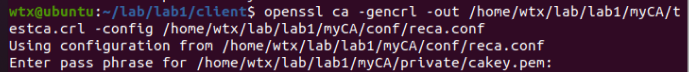


**指定第一个吊销证书的编号,注意：第一次更新证书吊销列表前，才需要执行**

echo 01 > /home/wtx/lab/lab1/myCA/crlnumber

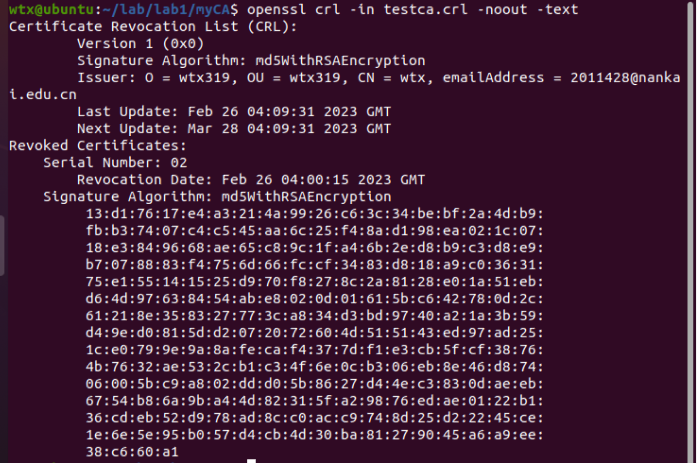
**更新证书吊销列表**

openssl ca -gencrl -out /home/wtx/lab/lab1/myCA/testca.crl -config /home/wtx/lab/lab1/myCA/conf/reca.conf



**查看吊销crl文件：**

openssl crl -in testca.crl -noout -text



# 实验结果

## •CA如何验证证书的有效性？

**证书的签发过程：**

1. 服务方 S 向第三方机构CA提交公钥、组织信息、个人信息(域名)等信息并申请认证；
2. CA 通过线上、线下等多种手段验证申请者提供信息的真实性，如组织是否存在、企业是否合法，是否拥有域名的所有权等；
3. 如信息审核通过，CA 会向申请者签发认证文件-证书。证书包含以下信息：申请者公钥、申请者的组织信息和个人信息、签发机构 CA 的信息、有效时间、证书序列号等信息的明文，同时包含一个签名；签名的产生算法：首先，使用散列函数计算公开的明文信息的信息摘要，然后，采用 CA 的私钥对信息摘要进行加密，密文即签名；
4. 客户端 C 向服务器 S 发出请求时，S 返回证书文件；
5. 客户端 C 读取证书中的相关的明文信息，采用相同的散列函数计算得到信息摘要，然后，利用对应 CA 的公钥解密签名数据，对比证书的信息摘要，如果一致，则可以确认证书的合法性，即公钥合法；
6. 客户端然后验证证书相关的域名信息、有效时间等信息；
7. 客户端会内置信任 CA 的证书信息(包含公钥)，如果CA不被信任，则找不到对应 CA 的证书，证书也会被判定非法。

在这个过程注意几点：

1.申请证书不需要提供私钥，确保私钥永远只能服务器掌握；

2.证书的合法性仍然依赖于非对称加密算法，证书主要是增加了服务器信息以及签名；

3.内置 CA 对应的证书称为根证书，颁发者和使用者相同，自己为自己签名，即自签名证书；

4.证书=公钥+申请者与颁发者信息+签名；

## •需要考虑到哪些方面？

1. 证书是否被吊销及证书是否失效
2. 用户信息
3. 通过hash值判断证书是否被篡改