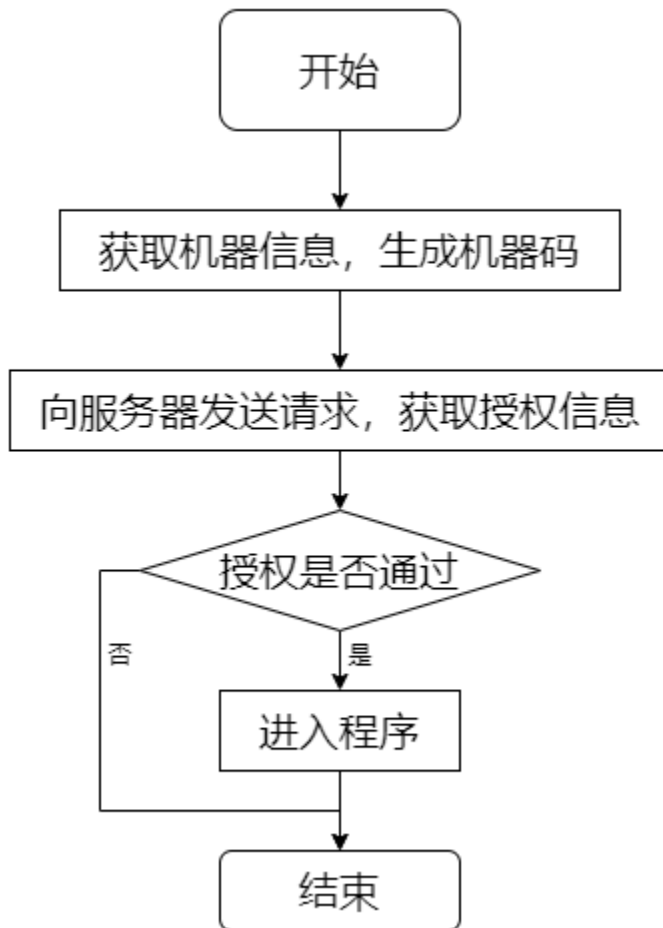


授权码机制 V3.0

客户端程序授权验证逻辑，如图所示。



客户端程序启动时向服务器发送验证请求，并根据服务器返回的验证结果做相应的操作。

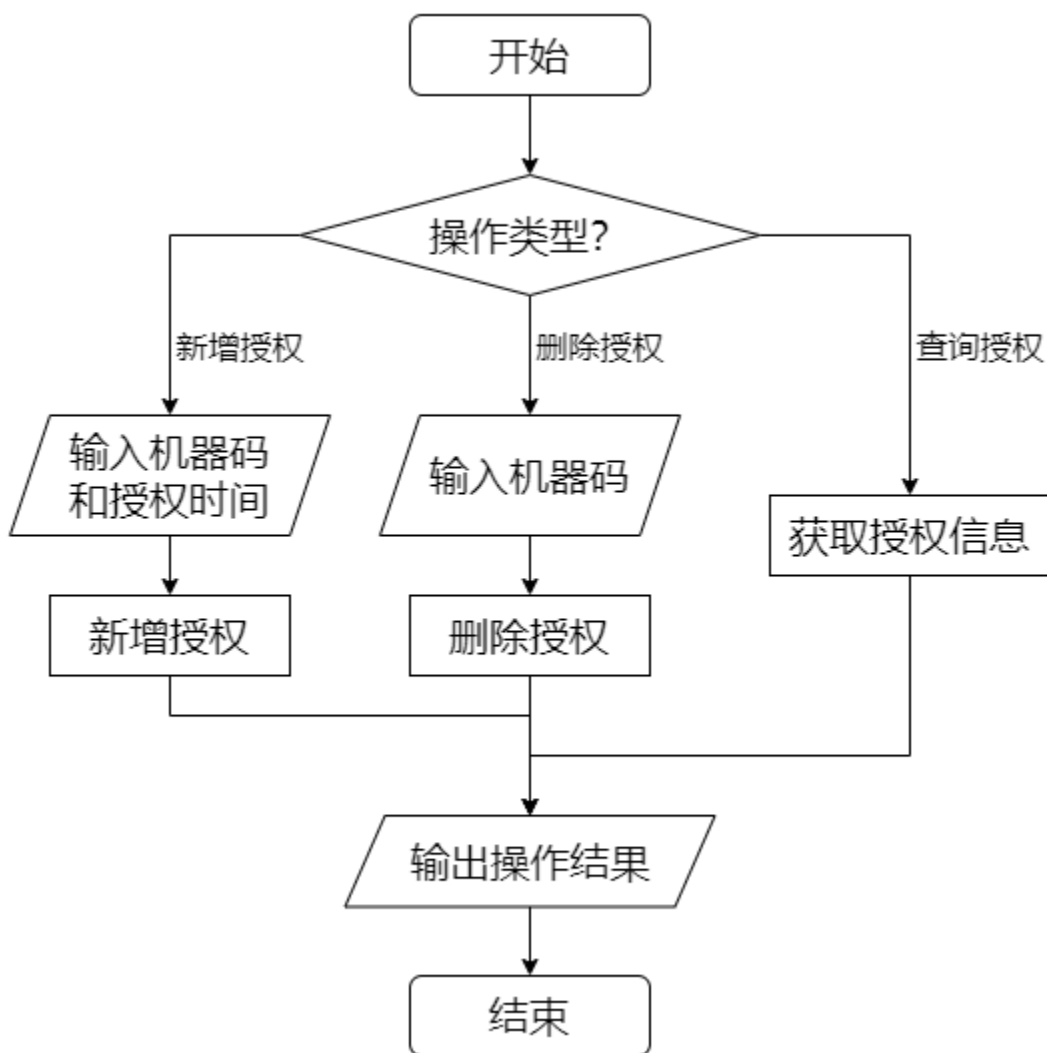
如果授权验证通过，则直接进入程序；否则将提示联系管理员进行授权，并退出程序。

用户将客户端程序生成的机器码告知管理员，申请使用授权。

等待管理员完成授权操作以后，重启客户端程序，授权验证通过后，即可正常使用软件。

1.2 注册机的逻辑

注册机程序的授权管理逻辑，如图所示。



注册机程序用来对程序进行授权管理，包括 [添加授权](#)，[删除授权](#)，[授权查询](#) 等功能。

添加授权时，输入机器码和授权时间，然后向服务器请求注册，服务器收到请求后会在授权表中新增一条记录。

删除授权时，输入机器码，然后向服务器请求删除授权，服务器收到请求后会在授权表中清空相应的记录。

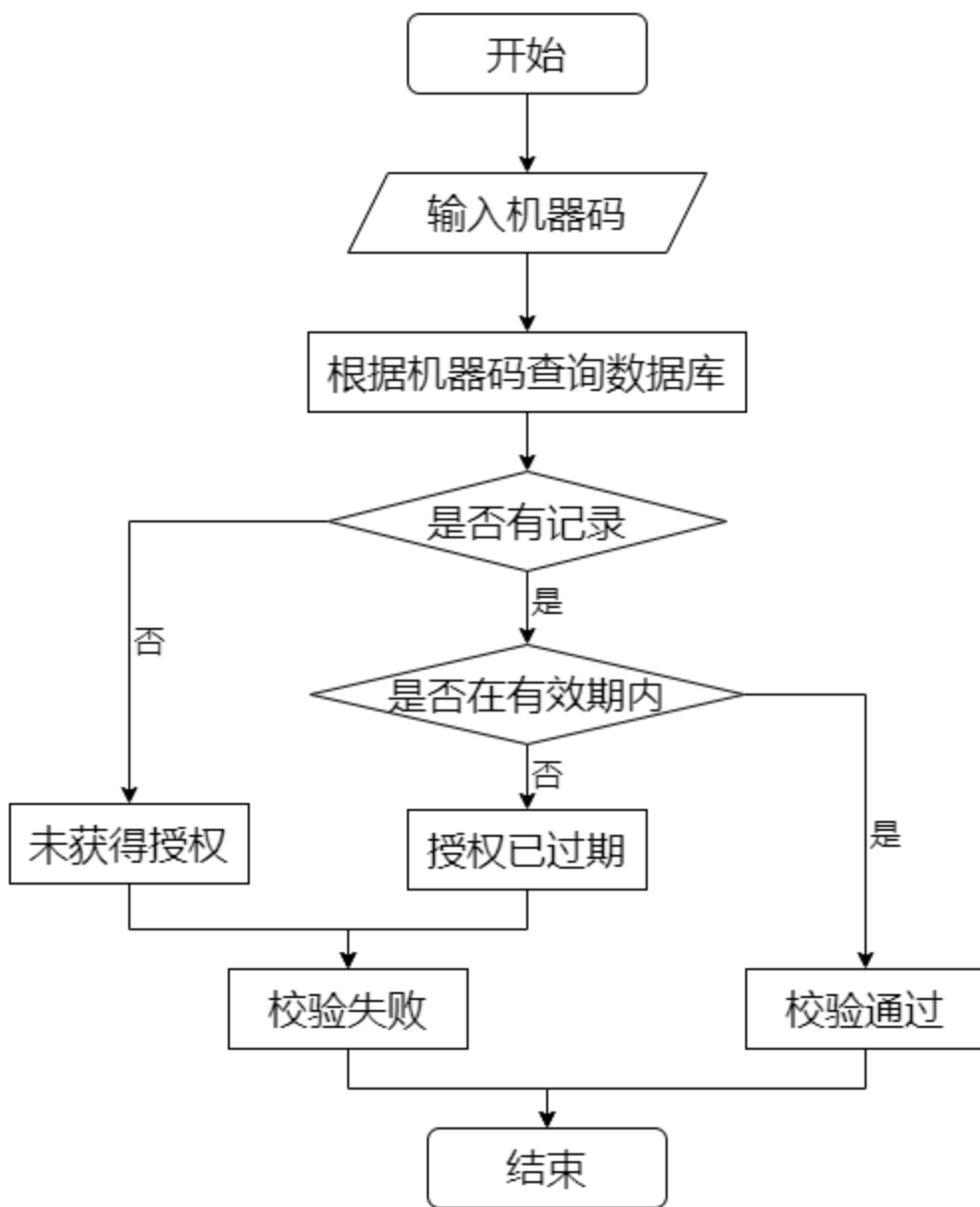
授权查询时，向服务器发送请求，服务器会查询现有的授权信息，并将查询结果返回。

1.3 服务器的逻辑

服务器的功能分为两部分，[授权校验](#) 和 [授权管理](#)。

授权校验时，请求主要来自于客户端。服务器根据请求参数中的机器码查询数据，若未查到记录，则说明该机器未授权，验证失败；若查到记录，则检查其授权时间是否过期，若过期说明授权失效，验证失败，否则验证通过。

服务器端授权校验逻辑，如图所示。



授权管理大致分为 [新增授权](#)、[撤销授权](#)、[查询授权](#)

功能。授权管理的请求主要来自注册机。

新增授权时，向数据库中插入一条记录，包括机器码和授权时间。若数据库中已有记录，则更新数据。

撤销授权时，根据机器码查找记录，并将该条记录中的授权时间清零。

查询授权时，从数据库中根据条件查询记录，并将结果数据按照一定规则组织并返回。

2. 实现步骤

总体思路捋清楚后，接下来我们一步步把它完成。

2.1 客户端

将授权验证过程放到服务器以后，客户端的逻辑可以大大简化，不再需要复杂的加密解密算法，只需要请求服务器，然后根据响应结果做相应的处理即可。

2.1.1 封装机器码生成算法

为了使代码更加规范整洁，符合面向对象的思想，便于后续优化拓展等。这里对 机器码生成算法 进行了封装。使用时可以通过 `MachineCodeGenerator` 的实例化对象，调用 `getMachineCode()` 方法来生成机器码。

```
import wmi
import hashlib

class MachineCodeGenerator:
    def __init__(self):
        self.pre_str = "HJDKAH" # 前缀
        self.suf_str = "SDFDTY" # 后缀

    # 获取机器码，机器码由以下四部分拼接组成
    # 1、CPU序列号 2、MAC地址 3、硬盘序列号 4、主板序列号
    self.m_wmi = wmi.WMI()

    # cpu序列号 16位
    def get_cpu_serial(self):
        cpu_info = self.m_wmi.Win32_Processor()
        if len(cpu_info) > 0:
            serial_number = cpu_info[0].ProcessorId
            return serial_number
```

```

        else:
            return "ABCDEFGHijklmnop"

# 硬盘序列号 15 位
    def get_disk_serial(self):
        disk_info = self.m_wmi.Win32_PhysicalMedia()
        disk_info.sort()
        if len(disk_info) > 0:
            serial_number = disk_info[0].SerialNumber.strip()
            return serial_number
        else:
            return "WD-ABCDEFGHijkl"

# mac地址 12 位
    def get_mac_address(self):
        for network in self.m_wmi.Win32_NetworkAdapterConfiguration():
            mac_address = network.MacAddress
            if mac_address != None:
                return mac_address.replace(":", "")
        return "ABCDEF123456"

# 主板序列号 14 位
    def get_board_serial(self):
        board_info = self.m_wmi.Win32_BaseBoard()
        if len(board_info) > 0:
            board_id = board_info[0].SerialNumber.strip().strip('.')
            return board_id
        else:
            return "ABCDEFGHijklmn"

# 拼接生成机器码
    def getMachineCode(self):
        mac_address = self.get_mac_address()
        cpu_serial = self.get_cpu_serial()
        disk_serial = self.get_disk_serial()
        board_serial = self.get_board_serial()

        combine_str = self.pre_str + mac_address + cpu_serial + \
            disk_serial + board_serial + self.suf_str
        combine_byte = combine_str.encode("utf-8")

```

```
machine_code = hashlib.md5(combine_byte).hexdigest()
return machine_code.upper()
```

由于每个人的机器情况不甚相同，有的机器获取CPU序列号、硬盘序列号、Mac地址等会出现获取失败，或者每次重启机器获取到的值不同等问题。

具体原因和解决办法我不得而知，但是就本程序而言，大家可以在生成机器码时，将有问题的序列号剔除，或者使用固定的随机字符串替代。

2.1.2 优化授权校验流程

在线版的授权校验方式逻辑非常简单，只需要将机器码发送给服务器，然后根据返回的校验结果进行相应的处理即可。

这里约定，返回结果 `== 0` 表示未授权，`== -1` 表示永久授权，`> 0` 表示授权通过，其值为有效期时间戳。

客户端授权校验流程的核心代码如下：

```
def checkAuthored():
    print("正在校验授权信息，请稍等...")
    try:
        url = "http://xxxxxxx.com" # 服务器地址
        machine_code = MachineCodeGenerator().getMachineCode()
        form = {
            "code": machine_code,
        }
        r = requests.post(url, data=form, timeout=5)
        endTime = int(r.text)
        if endTime == 0:
            print("您的设备没有授权，暂无法使用！")
            return False
        if endTime == -1:
            print("您的设备已获得【永久】使用授权！")
            return True
        if endTime > 0:
```

```

        time_local = time.localtime(endTime)
        dt = time.strftime("%Y-%m-%d %H:%M:%S", time_local)
        print("您的设备获得临时授权，授权有效期至【%s】" %dt)

        return True

    except:
        print("网络请求超时，请稍后再试！")

        return False

```

3.2 服务器

服务器使用 [Flask](#)

框架完成，两个路由，一个提供给客户端，用来做授权校验；另一个提供给注册机，用来授权管理。

```

from flask import Flask, request

app = Flask(__name__)

@app.route('/auth', methods=['POST'])
def authorize():
    # TODO: 授权校验

    return ""

@app.route('/regist', methods=['POST'])
def register():
    # TODO: 授权管理

    return ""

if __name__ == '__main__':
    app.run(debug=True)

```

3.2.1 授权管理类

我们将授权相关的操作封装到一个授权管理类 [AuthorizeManager](#) 中，提供了 [授权校验](#)，[新增授权](#)，[删除授权](#)，[授权查询](#) 等方法接口。核心代码如下：


```
import time
import json

class AuthorizeManager:

    def __init__(self):
        self.configPath = "config.json"
        self.authMap = {}

    # 新增授权
    def addAuth(self, code, ts):
        try:
            if code in self.authMap:
                endTime = self.authMap[code]
                if ts > endTime:
                    self.authMap[code] = ts
            else:
                self.authMap[code] = ts
            return "注册成功"
        except Exception as e:
            print(e)
            return "注册失败"

    # 删除授权
    def delAuth(self, code):
        try:
            if code in self.authMap:
                self.authMap[code] = 0
            return "删除成功"
        except Exception as e:
            print(e)
            return "删除失败"

    # 查询授权
    def queryAuth(self):
        retStr = "\n【授权结果如下】\n"
        authStr = ""
        for code, ts in self.authMap.items():
            if code and ts != 0:
```

```

        authStr += "{} {}{}\n".format(code, ts)
    if authStr:
        retStr += "机器码 {} 授权时间\n".format(code, ts)
        retStr += "-----\n"
        retStr += authStr
    else:
        retStr += "目前暂无授权"

    return retStr

# 授权校验
def checkAuth(self, code):
    # == 0 代表无授权,
    # == -1 代表永久授权
    # > 0 表示授权有效期的时间戳
    if code not in self.authMap:
        return 0

    endTime = self.authMap[code]
    if endTime <= 0:
        return endTime

    return endTime if (endTime > time.time()) else 0

authMgr = AuthorizeManager()

```

3.2.2 服务器代码

基于授权管理类中提供的接口，将服务器端代码完善一下。

```

from flask import Flask, request
from authorizeManger import authMgr

app = Flask(__name__)

@app.route('/auth', methods=['POST'])
def authorize():
    # 授权校验

```

```

    machine_code = request.form.get("code")
    result = authMgr.checkAuth(machine_code)

    return str(result)

@app.route('/regist', methods=['POST'])
def register():
    # 授权管理
    typeVal = int(request.form.get("type"))
    # type: 1 新增授权; 2 删除授权; 3 查询授权
    if typeVal == 1:
        machine_code = request.form.get("code")
        ts = request.form.get("ts")
        result = authMgr.addAuth(machine_code, int(ts))
    elif typeVal == 2:
        machine_code = request.form.get("code")
        result = authMgr.delAuth(machine_code)
    else:
        result = authMgr.queryAuth()

    return str(result)

if __name__ == '__main__':
    app.run(debug=True)

```

3.3 注册机

注册机的逻辑也非常简单，只需要根据服务器提供的接口，发送请求即可。

这里以 [新增授权](#)、[删除授权](#)、[查询授权](#) 为例，实现了授权管理的基本功能。核心代码如下：

```

import time
import requests

request_url = "服务器请求地址"

# 新增授权

```

```

def registCode(code, ts):
    url = "%s/regist" % request_url
    form = {
        "type": 1,
        "code": code,
        "ts" : ts,
    }
    r = requests.post(url, data=form)
    return r.text

# 删除授权
def deleteCode(code):
    url = "%s/regist" % request_url
    form = {
        "type": 2,
        "code": code,
    }
    r = requests.post(url, data=form)
    return r.text

# 查询授权
def queryInfo():
    url = "%s/regist" % request_url
    form = {
        "type": 3,
    }
    r = requests.post(url, data=form)
    return r.text

```

大家可以在此基础上进行拓展和完善，比如说添加可视化界面等等。

3. 效果演示

3.1 运行效果

由于客户端程序启动时要联网进行授权校验，所以需要保持网络畅通，并提前启动服务器。否则校验会失败。

```
C:\Windows\System32\cmd.exe - python client_main.py
正在校验授权信息,请稍等...
网络请求超时,请稍后再试!
请联系 13900000000 获取使用授权,授权后即可正常使用
机器码: B14526AA5B6DA89766A7E48D250E274C
Press Enter to exit...
```

首次启动程序时, 由于没有授权, 会提示 您的设备没有授权, 暂无法使用!。

```
C:\Windows\System32\cmd.exe - python client_main.py
正在校验授权信息,请稍等...
您的设备没有授权,暂无法使用!
请联系 13900000000 获取使用授权,授权后即可正常使用
机器码: B14526AA5B6DA89766A7E48D250E274C
Press Enter to exit...
```

此时我们启动注册机, 输入 1 选择新增授权, 根据提示复制 机器码 到注册机中, 并设置授权到期时间。

```
C:\Windows\System32\cmd.exe - python pojie.py
欢迎使用 xxx 注册机
您可以通过输入序号, 进行以下操作
【1】新增授权 【2】撤销授权 【3】查询授权信息

请输入操作序号:1
请输入机器码:B14526AA5B6DA89766A7E48D250E274C
请输入到期时间, 格式如: 2023-05-20 12:00:00
2023-06-01 15:00:00
注册成功
Press Enter to exit...
```

注册机提示注册成功, 此时重新启动客户端程序, 显示获得临时授权, 校验通过, 输出了 Hello World! 字样。

```
C:\Windows\System32\cmd.exe - python client_main.py
正在校验授权信息，请稍等...
您的设备获得临时授权，授权有效期至【2023-06-01 15:00:00】
Hello World!
Press Enter to exit...
```

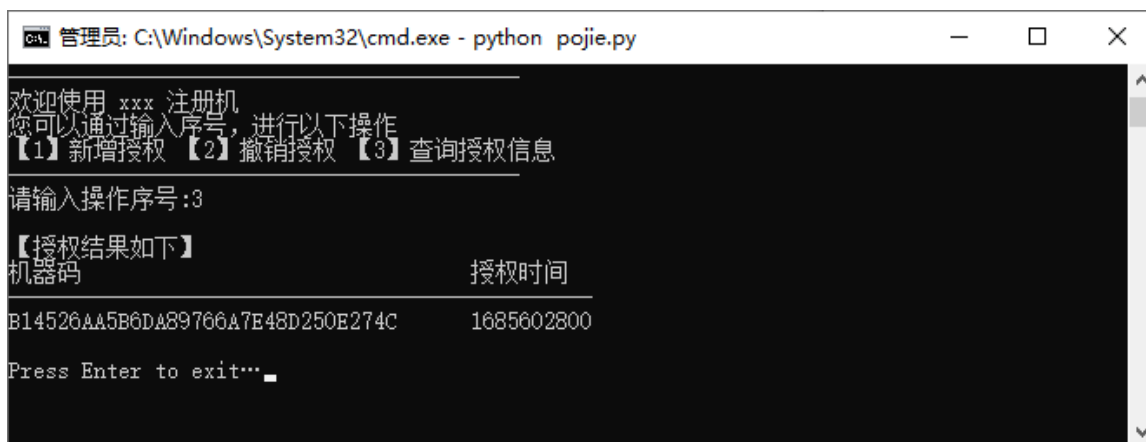
当你想要收回授权时，可以在注册机里选择 [撤销授权](#) 操作，输入要撤销授权的 [机器码](#)，即可完成。

```
C:\Windows\System32\cmd.exe - python pojie.py
欢迎使用 xxx 注册机
您可以通过输入序号，进行以下操作
【1】新增授权 【2】撤销授权 【3】查询授权信息
请输入操作序号:2
请输入机器码:B14526AA5B6DA89766A7E48D250E274C
删除成功
Press Enter to exit...
```

撤回授权以后，客户端程序再次启动时，便又会提示没有授权，暂无法使用。

```
C:\Windows\System32\cmd.exe - python client_main.py
正在校验授权信息，请稍等...
您的设备没有授权，暂无法使用!
请联系 13900000000 获取使用授权，授权后即可正常使用
机器码: B14526AA5B6DA89766A7E48D250E274C
Press Enter to exit...
```

此外，使用注册机可以查看当前已授权的所有设备，及其授权到期时间。



```
C:\Windows\System32\cmd.exe - python pojie.py
欢迎使用 xxx 注册机
您可以通过输入序号, 进行以下操作
【1】新增授权 【2】撤销授权 【3】查询授权信息
请输入操作序号:3
【授权结果如下】
机器码                                授权时间
B14526AA5B6DA89766A7E48D250E274C    1685602800
Press Enter to exit...
```

3.2 项目特点

相较于之前离线版的授权码机制，本项目有以下较为突出的优点：

1. 简单，不需要进行复杂的加密解密运算。
2. 安全，授权验证的逻辑全部放在了服务器端，用户无法通过修改本地时间等手段绕过授权验证。
3. 方便，管理者可以很方便的对自己的程序进行管理，包括：
 - 可以知道当前有多少人在用自己的软件
 - 可以随时调整任意客户端的授权时间，甚至撤销授权等等

4. 源码分享

以上便是授权码机制 V3.0

版的全部内容了，包括设计的总体思路，核心代码，以及运行效果展示。为了方便大家学习交流，我将项目源代码整理打包上传，有需要的朋友可以支持一下。

下载地址在下方 

精选留言

用户设置不下载评论