理解并发程序执行

Peterson 算法

一种抽象的解释

> A 和 B 争用厕所的包厢
>
> - 想进入包厢之前，A/B 都要先举起自己的旗子
>   - A 确认旗子举好以后，往厕所门上贴上 “B 正在使用” 的标签
>   - B 确认旗子举好以后，往厕所门上贴上 “A 正在使用” 的标签
> - 然后，
>
>   如果对方的旗子举起来，且门上的名字不是自己
>
>   ，等待
>   - 否则可以进入包厢
> - 出包厢后，放下自己的旗子

代码实现：

```
class Peterson:
    flag = '  '
    turn = ' '

    @thread
    def t1(self):
        while True:
            self.flag = '🏴' + self.flag[1]
            self.turn = '中'
            while self.flag[1] != ' ' and self.turn == '中':
                pass
            cs = True
            del cs
            self.flag = ' ' + self.flag[1]

    @thread
    def t2(self):
        while True:
            self.flag = self.flag[0] + '中'
            self.turn = '🏴'
            while self.flag[0] != ' ' and self.turn == '🏴':
                pass
            cs = True
            del cs
            self.flag = self.flag[0] + ' '
```

Model Checker.py

实现思路：

```
# python 创建状态机
def numbers(init=0, step=1):
    n = init
    while True:
        n += step
        yield n
```

```
>>> python3
>>> g = numbers()
>>> g
<generator object numbers at 0x107f873c0>
>>> g.__next__() # 状态机继续执行，直到执行到 yield 语句
1
>>> g.__next__()
2
```

使用方法：

```
# 自动生成程序状态机 (ubuntu下 /tmp)
# -t 参数: 生成树,
python3 model-checker.py mutex-bad.py | python3 visualize.py -t > a.html
```

Model Checking

> Model checking is a method for formally verifying finite-state systems——只要能为系统建立模型，就能用 prove by brute-force 证明正确/找到错误。

Model Chekcker.py: 程序的正确性 → 图论