# 利用管程完成哲学家进餐问题

# 用三种不同的状态表示哲学家的活动：进餐、饥饿、思考

# (thinking,hungry,eating) state[5]

# 为第i个(i=0,1,...,4)哲学家设置条件变量self[i]，当哲学家

# 饥饿又不能获得筷子时，用self[i].lock来阻塞自己。

# 管程设置三个函数：take\_forks取筷子, put\_forks放筷子, test测试是否具备进餐条件。

# !/usr/bin/env python

# coding:utf-8

import threading

mutex = threading.RLock()

state = [0, 0, 0, 0, 0]

r\_lock0 = threading.RLock()

r\_lock1 = threading.RLock()

r\_lock2 = threading.RLock()

r\_lock3 = threading.RLock()

r\_lock4 = threading.RLock()

class Phd():

def \_\_init\_\_(self, key, left, right, lock):

self.key = key

self.left = left

self.right = right

self.lock = lock

z1 = Phd(0, 1, 4, r\_lock0)

z2 = Phd(1, 0, 2, r\_lock1)

z3 = Phd(2, 1, 3, r\_lock2)

z4 = Phd(3, 2, 4, r\_lock3)

z5 = Phd(4, 3, 0, r\_lock4)

obj\_Phd\_list = [z1, z2, z3, z4, z5]

inter = 0

def take\_forks(name):

global inter

while 1:

inter += 1

key = name.key

mutex.acquire()

state[key] = 1

res = test(name)

mutex.release()

if res == 1:

print("----", name.key, "have eating----")

print("----", name.key, "put forks")

put\_forks(name)

else:

print("----", name.key, "no forks")

name.lock.acquire()

if inter >= 30:

break

def test(i):

print(i.key, "--in the test")

if state[i.key] == 1 & state[i.left] != 2 & state[i.right] != 2:

state[i.key] = 2

try:

i.lock.release()

except:

pass

return 1

return 0

def put\_forks(i):

mutex.acquire()

state[i.key] = 0

test(obj\_Phd\_list[i.right])

test(obj\_Phd\_list[i.left])

mutex.release()

for i in range(5):

s = threading.Thread(target=take\_forks, args=(obj\_Phd\_list[i],))

s.start()