Exp No:6 Title of the Exercise: Dining Philosophers Problem

Date:26/09/2022

Aim: --

To implement Dinning philosopher’s problem

Procedure: --

N philosophers, spend their time thinking and eating spaghetti. They eat at a round table with N individual seats. For eating each philosopher needs two forks (the resources). There are N forks on the table, one left and one right of each seat. When a philosopher cannot grab both forks it sits and waits. Eating takes random time, then the philosopher puts the forks down and leaves the dining room. After spending some random time thinking he again becomes hungry, and the circle repeats itself.

CODE: --

import threading

import random

import time

#inheriting threading class in Thread module

class Philosopher(threading.Thread):

running = True #used to check if everyone is finished eating

def \_\_init\_\_(self, index, forkOnLeft, forkOnRight):

threading.Thread.\_\_init\_\_(self)

self.index = index

self.forkOnLeft = forkOnLeft

self.forkOnRight = forkOnRight

def run(self):

while(self.running):

# Philosopher is thinking (but really is sleeping).

time.sleep(30)

print ('Philosopher %s is hungry.' % self.index)

self.dine()

def dine(self):

# if both forks are free, then philosopher will eat

fork1, fork2 = self.forkOnLeft, self.forkOnRight

while self.running:

fork1.acquire() # wait operation on left fork

locked = fork2.acquire(False)

if locked: break #if right fork is not available leave left fork

fork1.release()

print ('Philosopher %s swaps forks.' % self.index)

fork1, fork2 = fork2, fork1

else:

return

self.dining()

#release both the fork after dining

fork2.release()

fork1.release()

def dining(self):

print ('Philosopher %s starts eating. '% self.index)

time.sleep(30)

print ('Philosopher %s finishes eating and leaves to think.' % self.index)

def main():

forks = [threading.Semaphore() for n in range(5)] #initialising array of semaphore i.e forks

#here (i+1)%5 is used to get right and left forks circularly between 1-5

philosophers= [Philosopher(i, forks[i%5], forks[(i+1)%5])

for i in range(5)]

Philosopher.running = True

for p in philosophers: p.start()

time.sleep(100)

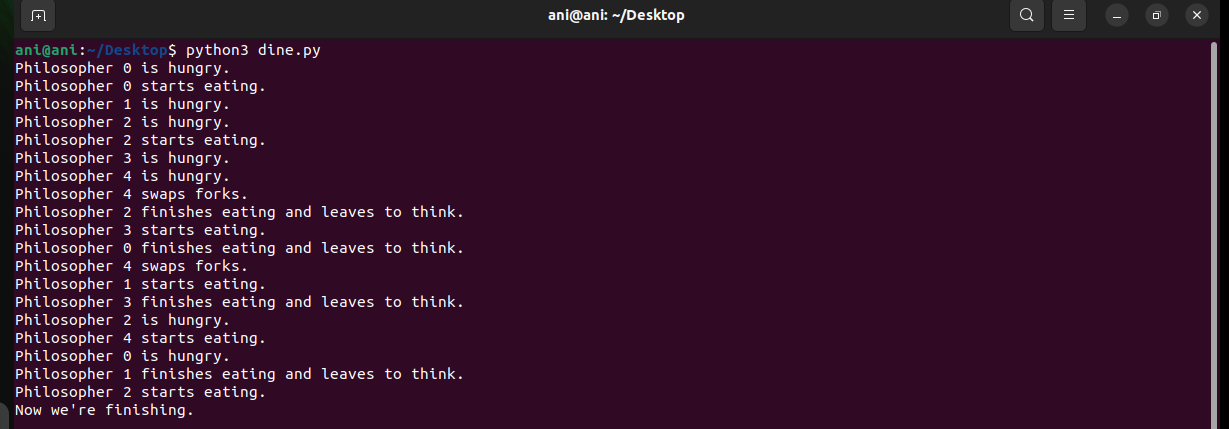
Philosopher.running = False

print ("Now we're finishing.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output: --



Result: --

We have successfully implemented dining philosopher’s problem in python