รายงานโครงงานวิชาระบบปฏิบัติการ (Operating system)

เรื่อง CPU Scheduling

จัดทำโดย

นาย ศิวกร วงศ์พลกานันท์ 600610781

เสนอ

รศ.ดร. นริศรา เอี่ยมคณิตชาติ

Concept & code of 3 algorithm

Source code : https://github.com/Thanatossan/OS

First come First serve

```python
def FCFS(FCFS_time, processList, numberProcess):
    temp = 0
    # time_array = []
    # process_array = []
    for i in range(len(processList)):
        FCFS_time += temp
        temp += processList[i]
        # process_array.append(i+1)
        # time_array.append(FCFS_time/numberProcess)

    # print(time_array)
    # print(process_array)
    FCFS_time = FCFS_time / numberProcess
    return FCFS_time
```

First come First serve จะทำการประมวลผลตัวแรกที่เข้ามาใน process ก่อนจากนั้นนำมาเรียงลำดับ โดย waiting time จะเพิ่มขึ้นตามจำนวนของ process ก่อนหน้าใช้ในการรอ และ average waiting time จะเท่ากับ waiting time ของ process ทั้งหมด หารด้วย จำนวน process โดย ฟังก์ชั้นนี้จะ return ค่า avg waiting time ของ First Come First Serve

Shortest-Job-First

```python
def SJF(SJF_time, processList, numberProcess):

    temp = 0
    # time_array = []
    # process_array = []
    for i in range(len(processList)):
        SJF_time += temp
        temp += min(processList)
        processList.remove(min(processList))
        # time_array.append(SJF_time/numberProcess)
        # process_array.append(i+1)
    # print(process_array)
    # print(time_array)
    SJF_time = SJF_time / numberProcess
    return SJF_time
```

Process ทุกตัวมีลำดับการประมวลผลที่เท่ากัน เพราะเนื่องจากไม่มีการคิด arrival time ดังนั้น SJF algorithm จะเอาตัว process ที่มีเวลาน้อยที่สุดมาประมวลผลก่อน เพื่อมาประมวลผล waiting time และ average waiting time จึงเท่ากับ waiting time ของ process ทั้งหมด หารด้วยจำนวน process ทั้งหมด ซึ่งฟังก์ชั่น SJF return ค่า average waiting time ของ shortest job first

Round Robin

```python
def RR(RR_Time, processList, numberProcess):
    countTime = 0

    end_process = [0] * numberProcess
    timeQuantum = 8
    point = 0
    # time_array = []
    # process_array = []
    i = 0
    while len(processList) > 0:
        point = point % (len(processList))
        RR_Time += (countTime - end_process[point])
        if processList[point] > timeQuantum:
            processList[point] -= timeQuantum
            countTime += timeQuantum
            end_process[point] = countTime
        elif processList[point] <= timeQuantum:
            countTime += processList[point]
            processList.remove(processList[point])
            end_process.remove(end_process[point])
            point -= 1
        point += 1
        i = i+1
        # time_array.append(RR_Time/numberProcess)
        # process_array.append(i+1)
    RR_Time = RR_Time/numberProcess

    return RR_Time
```

Round Robin จะจัด queue ให้แต่ละ process ประมวลผลได้ตามลำดับ แต่มีตัว Time Quantum ที่จะหนดเวลา time out ของการทำงานของ process โดยถ้า time out ตัว Time Quantum จะไปทำ process ต่อไป โดยค่าที่ return ออกจาก ฟังก์ชั่นคือ average waiting time ของ Round Robin

Experimental result

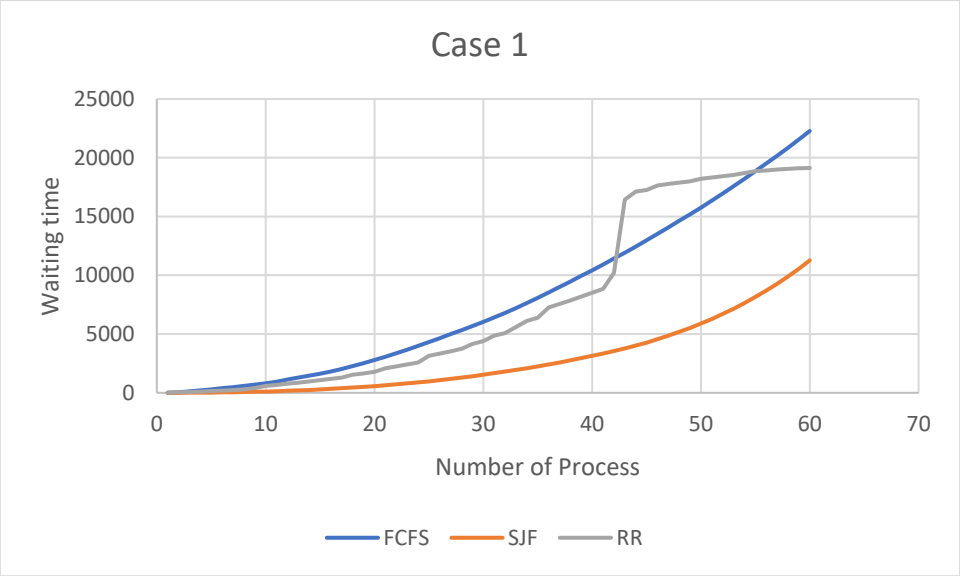
สมมติฐานที่ 1 ถ้ามีชุดโปรเซส ที่ต้องการใช้ CPU ความยาวไม่เท่ากันในการทำงาน เข้ามาเพื่อให้จัดคิวจำนวน 60 โปรเซส โดยทำการสุ่มโปรเซสที่ใช้เวลา (2 ถึง 8 milisec) จำนวน 70 %, โปรเซสที่ใช้เวลา (20 ถึง 30 milisec) จำนวน 20 % , โปรเซสที่ใช้เวลา (35 ถึง 40 milisec) จำนวน 10 % และเมื่อได้โปรเซสครบแล้วให้สุ่มลำดับที่เข้ามาในคิว

Dataset : [39, 40, 6, 4, 6, 7, 4, 2, 4, 29, 24, 4, 3, 3, 8, 40, 5, 28, 5, 30, 6, 6, 8, 7, 23, 3, 3, 3, 4, 6, 30, 35, 8, 7, 6, 20, 5, 2, 2, 21, 8, 4, 7, 24, 8, 3, 6, 7, 8, 7, 39, 4, 5, 20, 6, 4, 22, 37, 5, 21]

AVG First come first serve time = 371.35 milisecond

AVG Shortest-Job-First = 187.85 milisecond

AVG Round Robin = 309.65 milisecond



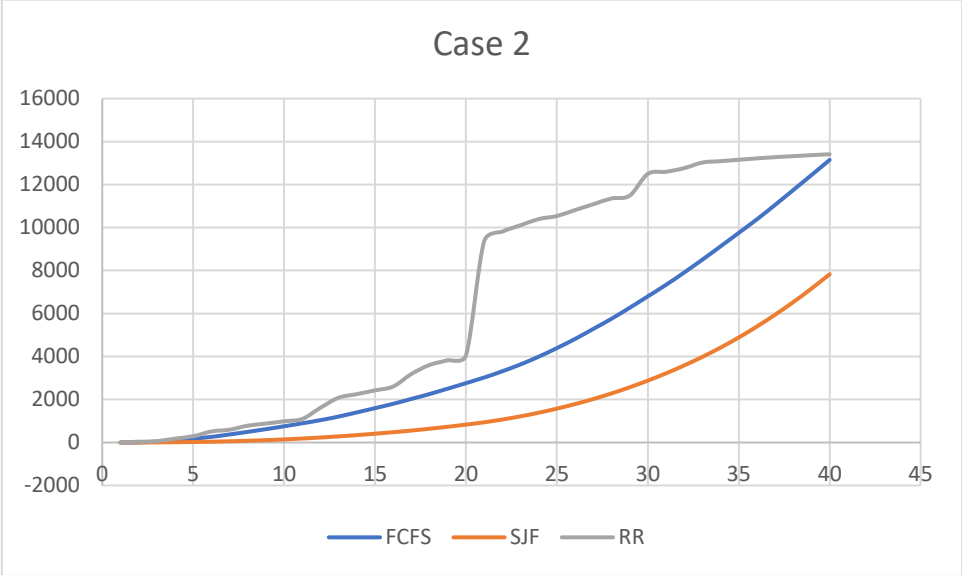
สมมติฐานที่ 2 ถ้ามีชุดโปรเซส ที่ต้องการใช้ CPU ความยาวไม่เท่ากันในการทำงาน เข้ามาเพื่อให้จัดคิวจำนวน 40 โปรเซส ดยทำการสุ่มโปรเซสที่ใช้เวลา (2 ถึง 8 milisec) จำนวน 50 %, โปรเซสที่ใช้เวลา (20 ถึง 30 milisec) จำนวน 30 % , โปรเซสที่ใช้เวลา (35 ถึง 40 milisec) จำนวน 20 % และเมื่อได้โปรเซสครบแล้วให้สุ่มลำดับที่เข้ามาในคิว

Dataset : [4, 36, 21, 5, 7, 28, 40, 4, 23, 5, 40, 20, 4, 3, 39, 4, 5, 5, 8, 21, 22, 27, 6, 20, 26, 3, 7, 7, 5, 24, 37, 4, 20, 5, 8, 5, 24, 38, 36, 37]

AVG First come first serve time = 319.65 milisecond

AVG Shortest-Job-First = 188.925 milisecond

AVG Round Robin = 335.25 milisecond



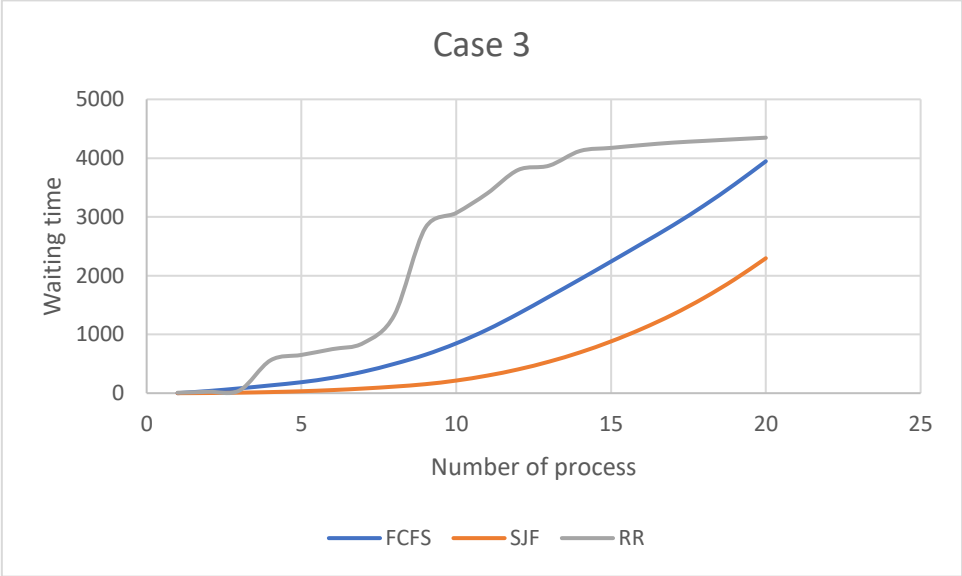
สมมติฐานที่3 ถ้ามีชุดโปรเซส ที่ต้องการใช้ CPU ความยาวไม่เท่ากันในการทำงาน เข้ามาเพื่อให้จัดคิวจำนวน 40 โปรเซส ดยทำการสุ่มโปรเซสที่ใช้เวลา (2 ถึง 8 milisec) จำนวน 50 %, โปรเซสที่ใช้เวลา (20 ถึง 30 milisec) จำนวน 30 % , โปรเซสที่ใช้เวลา (35 ถึง 40 milisec) จำนวน 20 % และเมื่อได้โปรเซสครบแล้วให้สุ่มลำดับที่เข้ามาในคิว

Dataset : [37, 8, 6, 3, 21, 29, 29, 22, 39, 39, 37, 22, 3, 7, 4, 4, 26, 28, 26, 6]

AVG First come first serve time = 197.3 milisecond

AVG Shortest-Job-First time = 114.8 milisecond

AVG Round Robin time = 217.4 milisecond



Conclusion

จากข้อมูล Data set ข้างต้นที่ได้นำมาทดลองรันผ่าน FCFS,SJF และ RR เวลาที่ได้เมื่อนำมาสร้างกราฟ waiting time ต่อ number of process แสดงให้เห็นว่า Algorithm ในการจัดการเวลาในแต่ละ process Shortest job first ดีที่สุด เพราะใช้เวลาประมวลผมน้อยที่สุด และรองลมาคือ Round robin ที่กำหนด time quantum = 8 ที่ทำให้การทำงานบางช่วงดีกว่า Shortest job first และ first come first serve ทำงานช้าที่สุดเพราะจัดการแค่ process ที่มาก่อนได้ทำงานก่อน