Activity 5: Process Scheduling

	ชื่อ - นามสกุล	รหัสนิสิต
1	Keerati Setthasuk	6733021521
2	Peerawit Seemamahawong	6733183521
3	Supanut Udompataikul	6733278521

วัตถุประสงค์

1. เพื่อให้นิสิตเข้าใจหลักการของ process scheduling
2. เพื่อให้นิสิตสามารถเปรียบเทียบผลการทำงานของ scheduling algorithm แบบต่างๆ

สิ่งที่ต้องทำ

ใช้ simulator ในการจำลอง process scheduling ด้วย algorithm ต่างๆ ตามโจทย์ และใส่ผลลัพธ์หรือตอบคำถามในพื้นที่ที่เว้นไว้ให้ในเอกสารนี้

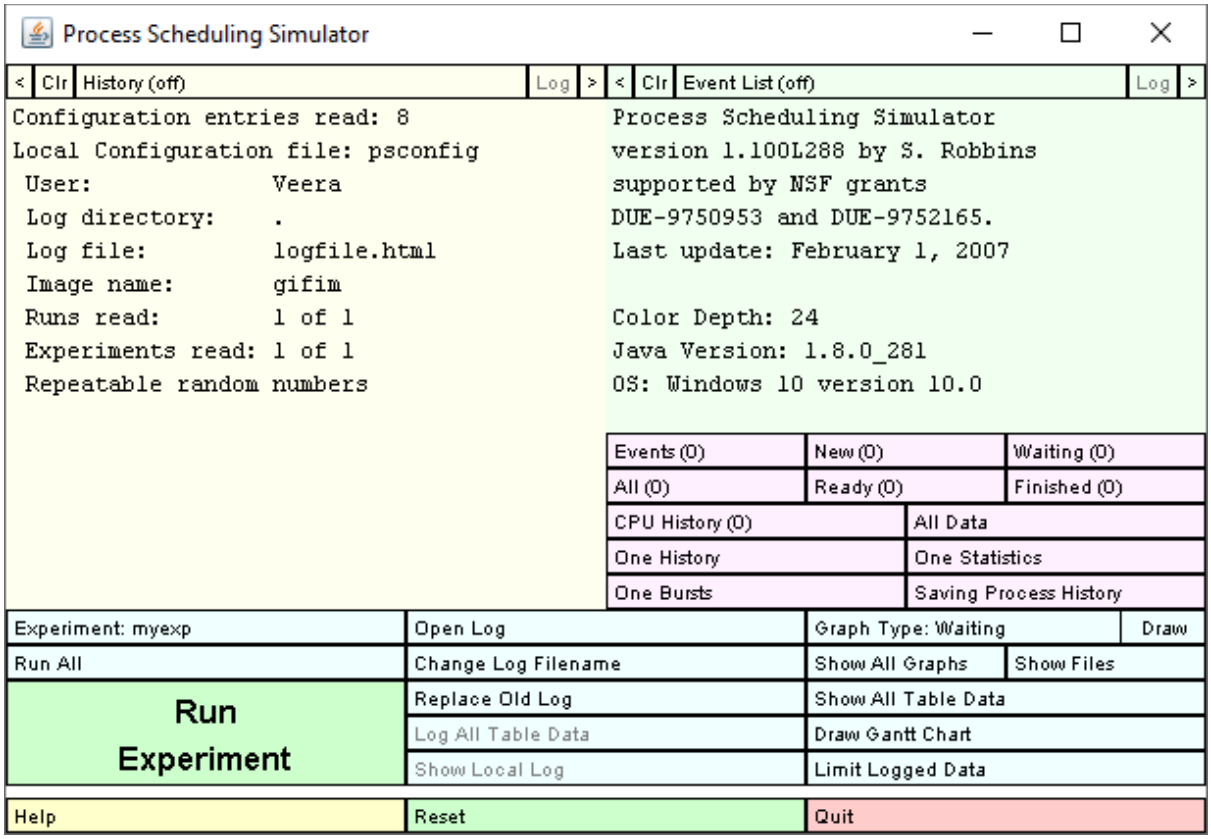
การส่งงาน

ส่งเป็นไฟล์ pdf ของเอกสารนี้ที่เติมผลลัพธ์และคำตอบแล้ว โดยให้ใส่รายชื่อและเลขประจำตัวของสมาชิกในกลุ่มทุกคนด้วย

ติดตั้ง simulator

1. ติดตั้ง Java ลงในเครื่อง Notebook ของสมาชิกในกลุ่มอย่างน้อย 1 เครื่อง
2. Download ไฟล์ ps.zip จาก course material ในส่วนของ Activity 5: Process Scheduling (ps.zip) แล้ว unzip

3. ทดลองว่าโปรแกรมสามารถใช้งานได้โดยเข้าไปที่ folder ps แล้วเรียกใช้คำสั่ง "runps.bat" (สำหรับ Windows) หรือ "runps.sh" (สำหรับ linux หรือ mac os x) จะได้ผลลัพธ์ดังนี้



4. ศึกษาการใช้งานเพิ่มเติมจากไฟล์ ps_doc.html ใน folder ps

ใน folder ps จะมีไฟล์สำหรับการตั้งค่าการจำลองอยู่ 2 ไฟล์คือ

- myrun.run เป็นไฟล์ที่กำหนดค่า parameter ต่างๆ ของการจำลองในแต่ละครั้งเช่น

  - algorithm = scheduling algorithm

  - numprocs = จำนวนโปรเซส

  - firstarrival = เวลาที่โปรเซสแรกมาถึง

  - interarrival = ระยะห่างระหว่างเวลาที่โปรเซสจะเข้ามาใช้ซีพียู โดยระบุเป็น probability distribution

  - duration = ระยะเวลาโดยรวมที่โปรเซสจะใช้งานซีพียู โดยระบุเป็น probability distribution

- cpuburst = ระยะเวลาการใช้งานซีพียูแต่ละครั้ง (cpu burst time) โดยระบุเป็น probability distribution
- ioburst = ระยะเวลาการใช้งาน I/O แต่ละครั้ง (I/O burst time) โดยระบุเป็น probability distribution

- probability distribution มีอยู่ 3 แบบ คือ constant, exponential และ uniform

ตัวอย่างไฟล์ myrun.run

```
name myrun
comment This contains two types of processes
algorithm SJF
seed 5000

numprocs 15
firstarrival 0.0
interarrival constant 0.0
duration uniform 10.0 15.0
cpuburst constant 10.0
ioburst uniform 10 20
basepriority 1.0

numprocs 15
firstarrival 0.0
interarrival constant 0.0
duration constant 4.0
cpuburst constant 1.0
ioburst uniform 10.0 20.0
basepriority 1.0
```

ไฟล์ตัวอย่างนี้กำหนดให้การจำลองแต่ละครั้ง จะมีการสร้างโปรเซส จำนวน 30 โปรเซส โดยแบ่งเป็นสองกลุ่ม กลุ่มละ 15 โปรเซส สิ่งที่แตกต่างกันระหว่างสองกลุ่มนี้คือขนาดของงานโปรเซสในกลุ่มแรกมีเวลาในการทำงานอยู่ในช่วงระหว่าง 10-15 time unit และมี cpu burst คงที่คือ 10 unit ส่วนกลุ่มที่สองมีเวลาทำงานเท่ากันทุกโปรเซสคือ 4 unit และมี cpu burst คงที่คือ 1 unit
โดยทุกโปรเซสจะเข้ามาใช้ซีพียู (firstarrival) ที่เวลาเดียวกันคือเวลา 0 และมี io burst ในช่วง 10-20 unit

- myexp.exp เป็นไฟล์ที่กำหนดภาพรวมการจำลองทั้งหมดว่าจะต้องทำการจำลองด้วยค่า parameter ตามที่กำหนดใน myrun.run เป็นจำนวนกี่ครั้ง และสามารถกำหนดค่า parameter จำเพาะสำหรับการ run ในแต่ละครั้งได้

  ตัวอย่างเช่น

```
name myexp
comment This experiment contains 2 runs
run myrun algorithm FCFS key "FCFS"
run myrun algorithm SJF key "SJF"
```

  ตัวอย่าง myexp.exp ข้างต้น จะเป็นการกำหนดให้ทำการจำลอง 2 ครั้ง โดยครั้ง แรกจะเป็นการใช้ FCFS ในการทำ process scheduling และในครั้งที่ 2 จะใช้ SJF

1. เริ่มใช้งาน simulator โดยเข้าไปที่ folder ps แล้วเรียกใช้คำสั่ง "runps.bat" (สำหรับ Windows) หรือ "runps.sh" (สำหรับ linux หรือ mac os x)
2. กดปุ่ม “Run Experiment” (ปุ่มสีเขียวใหญ่ๆที่อยู่ด้านล่างซ้าย) เพื่อเริ่มการจำลอง process scheduling สำหรับ 30 โปรเซส ทั้งในแบบ SJF (shortest-job-first) และ FCFS (first-come-first-served)

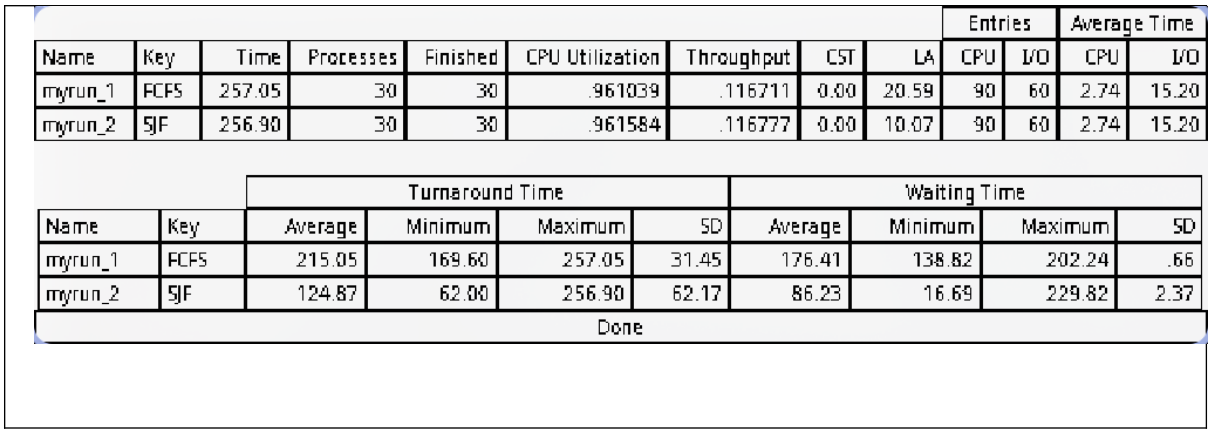
3. กดปุ่ม “Show All Table Data” (ปุ่มกลางของแถวขวาสุด) เพื่อเรียกดูค่าสถิติต่างๆ ของผลจากการจำลอง

4. กดปุ่ม “Draw Gantt Chart” (ปุ่มกลางของแถวขวาสุด) เพื่อเรียกดูกราฟแสดงสถานะ (Running, Ready, Waiting) ของแต่ละโปรเซสในช่วงเวลาของการจำลอง โดยสามารถเลือกได้ว่าจะดูกราฟของ FCFS หรือ SJF และสามารถเก็บภาพกราฟลงไฟล์ได้ โดยการกดปุ่ม “Save” ในบรรทัดล่างสุดของหน้าต่างนี้ แล้วป้อนชื่อไฟล์ เช่น fcfs.gif

5. ออกจากโปรแกรมโดยการกดปุ่ม “Quit” (ปุ่มสีชมพูที่อยู่ด้านล่างขวา)

ส่วนที่ 1

1. แสดงตารางที่ได้ในขั้นตอนที่ 3 "Show All Table Data"



									Entries		Average Time	
Name	Key	Time	Processes	Finished	CPU Utilization	Throughput	CST	LA	CPU	I/O	CPU	I/O
myrun_1	FCFS	257.05	30	30	.961039	.116711	0.00	20.59	90	60	2.74	15.20
myrun_2	SJF	256.90	30	30	.961584	.116777	0.00	10.07	90	60	2.74	15.20

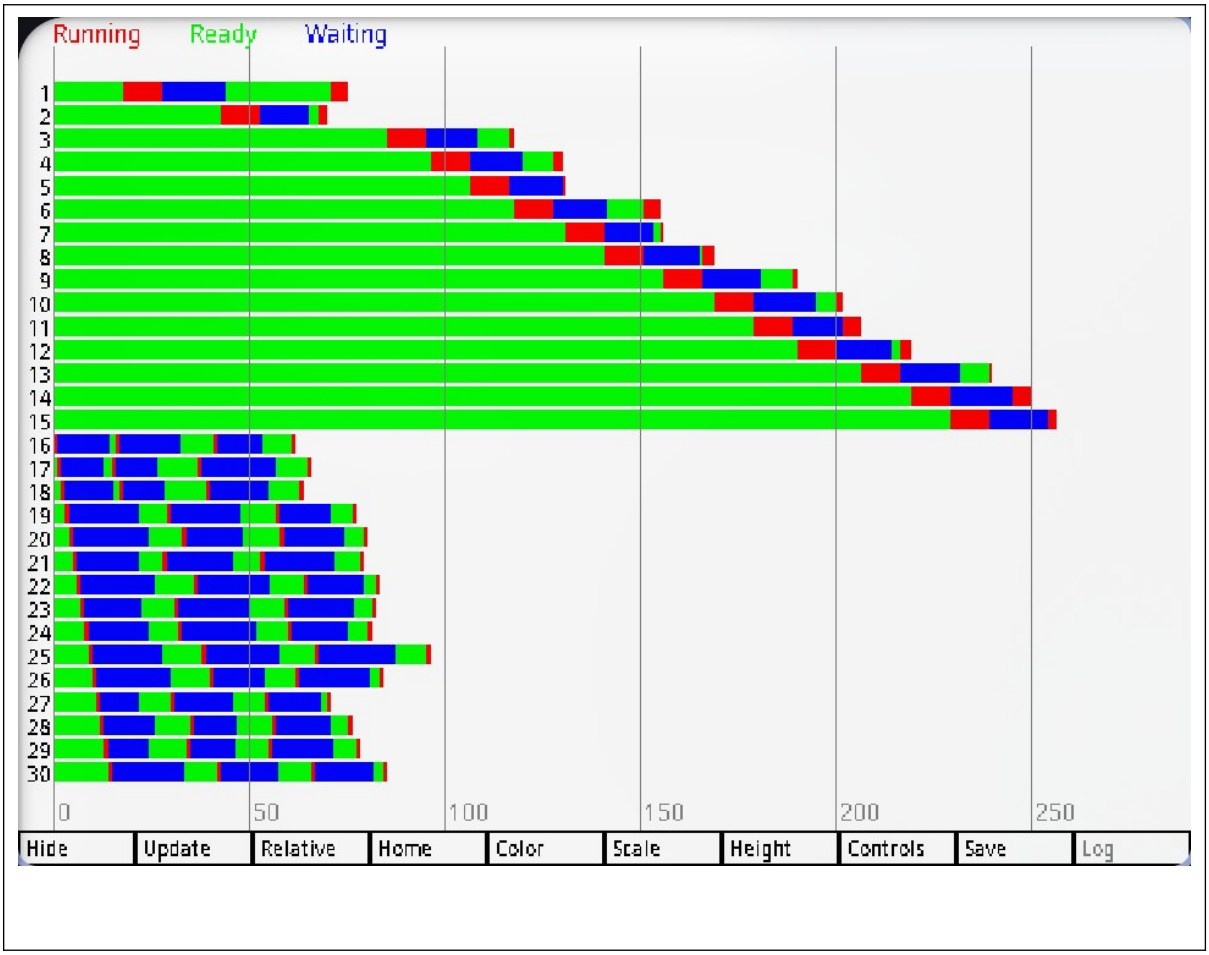
		Turnaround Time				Waiting Time			
Name	Key	Average	Minimum	Maximum	SD	Average	Minimum	Maximum	SD
myrun_1	FCFS	215.05	169.60	257.05	31.45	176.41	138.82	202.24	.66
myrun_2	SJF	124.87	62.00	256.90	62.17	86.23	16.69	229.82	2.37

Done

2. พิจารณาจากตารางในข้อ 1 พบว่า scheduling algorithm อันไหนดีกว่า เมื่อใช้ตัวชี้วัดต่างๆ กัน (ทำเครื่องหมาย P ในช่องของอันที่ดีกว่า)

	FCFS	SJF
Average Waiting Time สั้นกว่า		P
Throughput มากกว่า		P
Average Turnaround Time สั้นกว่า		P
CPU Utilization มากกว่า		P
Maximum Waiting Time สั้นกว่า	P	

3. แสดงกราฟของ SJF ที่ได้ในขั้นตอนที่ 4 "Draw Gannt Chart"



4. พิจารณาจากกราฟที่ได้ในข้อ 3 จะเห็นได้ว่ามีโปรเซสหมายเลข 16 ถึง 30 ซึ่งมี CPU Burst เล็กกว่า ได้ทำงานจนเสร็จก่อนโปรเซสหมายเลข 1 ถึง 15 อย่างไรก็ตาม โปรเซสหมายเลข 1, 2, 3 ได้เริ่มรันครั้งแรกก่อนที่โปรเซส 16-30 จะรันเสร็จทั้งหมด ในขณะที่โปรเซส 4-15 ได้เริ่มรันเมื่อโปรเซส 16-30 รันเสร็จหมดแล้ว เพราะเหตุใด

เนื่องจาก เมื่อ CPU ว่างจาก process 16-30 ที่ burst CPU time น้อยกว่า 1-15 แต่ burst CPU time ของ 1-15 เท่ากันหมดในช่วงแรกทำให้ให้ทำ process แรกก่อน

ส่วนที่ 2

- แก้ไฟล์ myrun.run เป็นแบบนี้

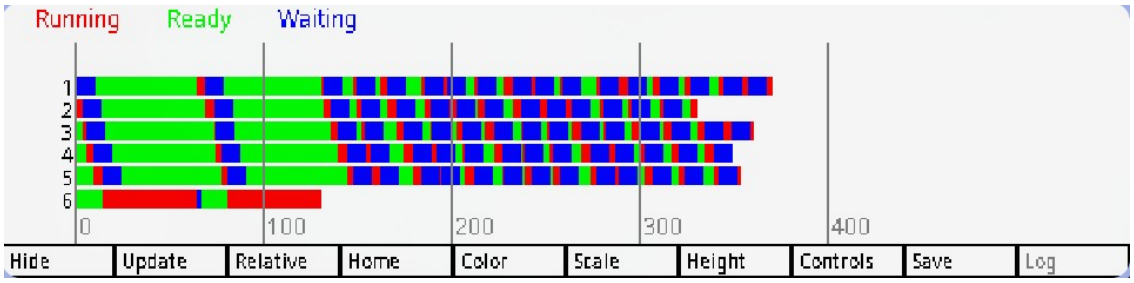
```
name myrun
comment two types of processes
algorithm FCFS
seed 5000
numprocs 5
firstarrival 0.0
interarrival constant 0.0
duration constant 50
cpuburst uniform 1 5
ioburst constant 10
basepriority 1.0

numprocs 1
firstarrival 0.0
interarrival constant 0.0
duration constant 100
cpuburst constant 50
ioburst uniform 1 5
basepriority 1.0
```

ไฟล์นี้ระบุรายละเอียดของโปรเซสสองแบบคือ แบบแรกเป็นแบบ I/O bound มี 5 โปรเซส แบบที่สองเป็นแบบ CPU bound มีหนึ่งโปรเซส

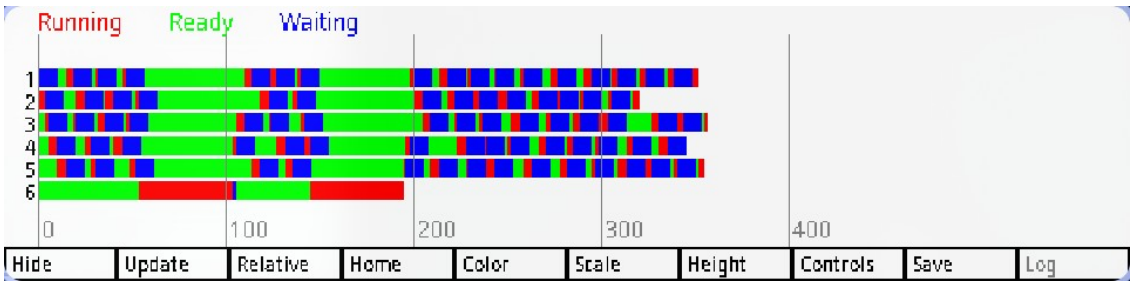
- ให้รันโปรแกรม simulation ใหม่อีกครั้ง พิจารณาตารางผลลัพธ์และ Gannt chart

5. แสดงตารางผลลัพธ์และ Gannt Chart ของทั้ง FCFS และ SJF

FCFS



SJF



6. พิจารณาจากตารางผลลัพธ์และ Gannt Chart ในข้อ 5 พบว่า scheduling algorithm ใดเป็นผลดีกับโปรเซสที่เป็น CPU bound มากกว่า เพราะอะไร

จาก Gannt Chart ในข้อ 5 Scheduling algorithm FCFS มีผลดีกับ CPU bound process มากกว่าเพราะได้เริ่มรันก่อน algo SJF และรันจบก่อน SJF เนื่องจาก CPU bound ใช้ CPU burst นานทำให้ใน SJF ได้ทำที่หลัง