Activity 6: Process Scheduling

วัตถุประสงค์

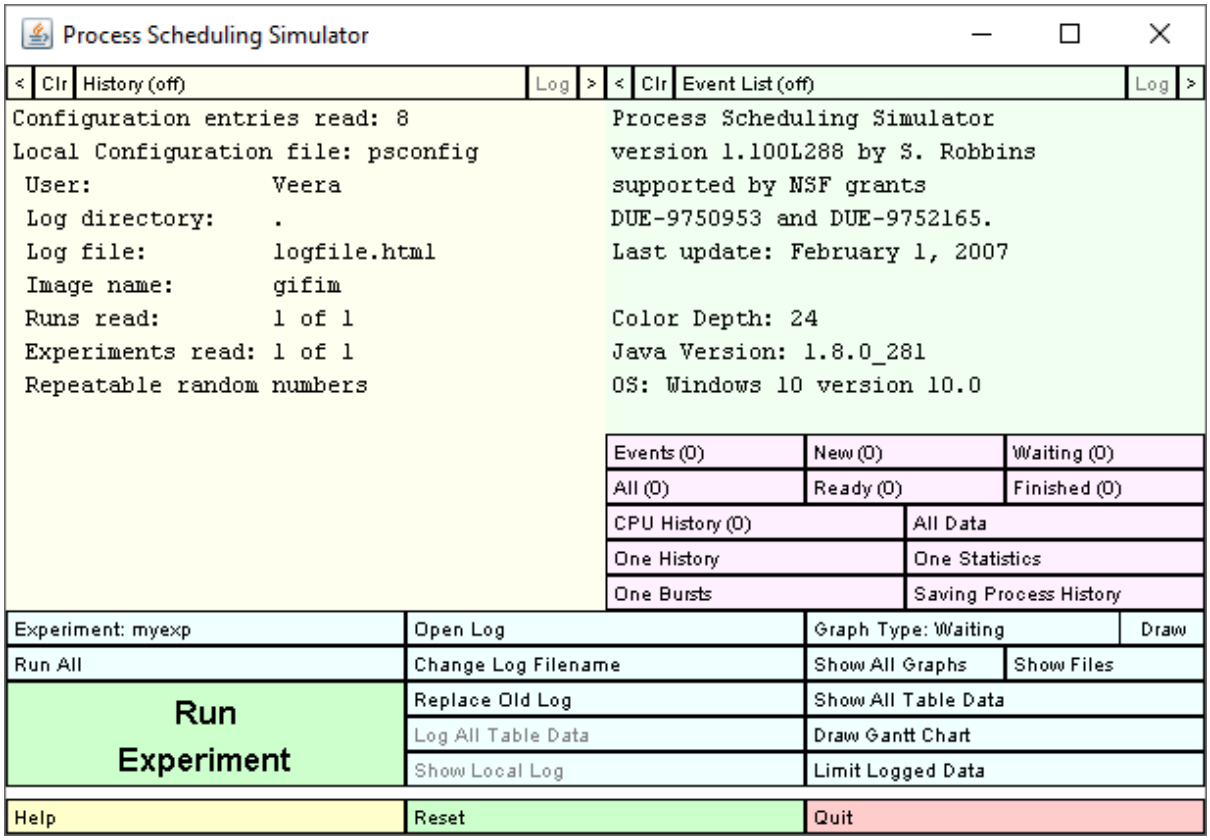
1. เพื่อให้นิสิตเข้าใจหลักการของ process scheduling
2. เพื่อให้นิสิตสามารถเปรียบเทียบผลการทำงานของ scheduling algorithm แบบต่างๆ

กิจกรรมในชั้นเรียน

ส่วนที่ 1 และ 2 ใช้ simulator ในการจำลอง process scheduling ด้วย algorithm ต่างๆ

ติดตั้ง simulator สำหรับส่วนที่ 1 และ 2

1. ติดตั้ง Javaรุ่น1.6 ขึ้นไปลงในเครื่อง Notebook ของสมาชิกในกลุ่มอย่างน้อย 1 เครื่อง
2. Download ไฟล์ ps.zip จาก course material ในส่วนของ Activity 6: Process Scheduling (ps.zip) แล้ว unzip
3. ใน folder ps จะมีไฟล์ psconfig ซึ่งจะเป็นไฟล์สำหรับการตั้งค่าต่างๆ ของsimulator ให้แก้ไข บรรทัดที่ 4 จากคำว่า user Local User เป็น user XXXXXXXXX ซึ่งเป็นรหัสนิสิตของสมาชิกในกลุ่ม 1 คน (คนใดก็ได้)
4. ทดลองว่าโปรแกรมสามารถใช้งานได้โดยเข้าไปที่ folder ps แล้วเรียกใช้คำสั่ง "runps.bat" (สำหรับ Windows) หรือ "runps.sh" (สำหรับ linux หรือ mac os x) จะได้ผลลัพธ์ดังนี้



5. ศึกษาการใช้งานเพิ่มเติมจากไฟล์ ps_doc.html ใน folder ps

ส่วนที่ 1

ใน folder ps จะมีไฟล์สำหรับการตั้งค่าการจำลองอยู่ 2 ไฟล์คือ

- myrun.run เป็นไฟล์ที่กำหนดค่า parameter ต่างๆ ของการจำลองในแต่ละครั้งเช่น จำนวน process (numprocs) ระยะห่างระหว่างเวลาที่ process จะเข้ามาใช้ CPU (interarrival) ระยะเวลาที่ process จะใช้งาน cpu (duration) ลักษณะการใช้งาน CPU ของแต่ละ process (cpuburst)
  และลักษณะการใช้งาน I/O ของแต่ละ process (ioburst)

  ตัวอย่างเช่น

```
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
```

ไฟล์นี้กำหนดให้การจำลองแต่ละครั้ง จะมีการสร้าง process จำนวน 30 process โดยแบ่งเป็นสองกลุ่ม กลุ่มละ 15 โปรเซส สิ่งที่แตกต่างกันระหว่างสองกลุ่มนี้คือขนาดของงาน กลุ่มแรกมีเวลาในการทำงานอยู่ในช่วงระหว่าง 10-15 unit และมี cpu burst คงที่คือ 10 unit ส่วนกลุ่มที่สองมีเวลาทำงานคงที่คือ 4 unit และมี cpu burst คงที่คือ 1 unit

โดยทุก process จะเข้ามาใช้ cpu ที่เวลาเดียวกันคือเวลา 0 และมี io burst ในช่วง 10-20 unit

- myexp.exp เป็นไฟล์ที่กำหนดภาพรวมการจำลองทั้งหมดว่าจะต้องทำการจำลองด้วยค่า parameter ตามที่กำหนดใน myrun.run เป็นจานวนกี่ครั้ง 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 process 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) ของแต่ละ process ในช่วงเวลาของการจำลอง โดยสามารถเลือกได้ว่าจะดูกราฟของ FCFS หรือ SJF และสามารถเก็บภาพกราฟลงไฟล์ได้ โดยการกดปุ่ม “Save” ในบรรทัดล่างสุดของหน้าต่างนี้ แล้วป้อนชื่อไฟล์ เช่น fcfs.gif

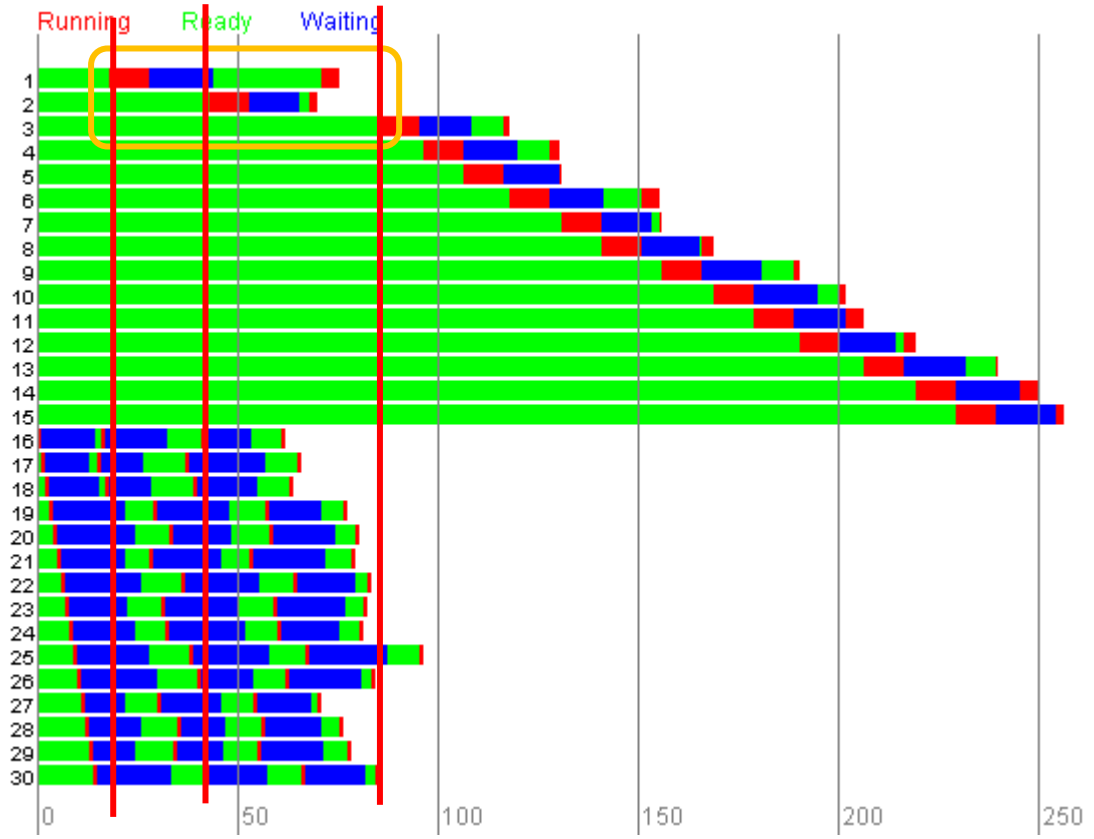
5. ออกจากโปรแกรมโดยการกดปุ่ม “Quit” (ปุ่มสีชมพูที่อยู่ด้านล่างขวา)

ตอบคำถามต่อไปนี้

1. จากตารางที่ได้ในขั้นตอนที่ 3 "Show All Table Data" แสดงว่า scheduling algorithm อันไหนดีกว่า เมื่อใช้ตัวชี้วัดต่างๆ กัน (ทำเครื่องหมาย ✓ ในช่องของอันที่ดีกว่า)

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

2. พิจารณาจากกราฟที่ได้ในขั้นตอนที่ 4 "Draw Gannt Chart from SJF" 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



เพราะตอนที่ process 1 ได้ทำงานครั้งแรก ตอนนั้น process 16 – 30 ติดสถานะ waiting ทำให้ CPU จึงหยิบ process ที่มี duration ที่เหลือน้อยที่สุดมากทำต่อ ซึ่งก็คือ process แรก ในทำนองเดียวกัน process ที่ 2 กับ ที่ 3 ได้ทำงานแรกเพราะว่า process ที่ได้ทำงานไปแล้วติดสถานะ waiting หมด

ส่วนที่ 2

- แก้ไฟล์ myrun.run เป็นแบบนี้

```
name myrun
comment This run specifies one type of process
algorithm FCFS
seed 5000
numprocs 20
firstarrival 0.0
```

```
interarrival constant 0.0
duration constant 100
cpuburst uniform 10 100
ioburst constant 10
basepriority 1.0
```

ไฟล์นี้กำหนดให้การจำลองแต่ละครั้ง จะมีการสร้าง process จำนวน 20 process โดยทุก processจะเข้ามาใช้ cpu ที่เวลาเดียวกันคือเวลา 0 โดยแต่ละ process จะ ใช้เวลา cpu ทั้งหมด 100 unit และมีลักษณะการใช้งาน cpu แบบต่อเนื่องครั้งละเป็นช่วงเวลาระหว่าง 10-100 unit ก่อนที่จะสลับไปใช้ I/O เป็นช่วงเวลาคงที่คือ 10 unit

- ให้รันโปรแกรม simulation ใหม่อีกครั้ง พิจารณาตารางผลลัพธ์และ Gannt chart
- จากผลการทดลอง scheduling algorithm ใดให้ผลลัพธ์ที่ดีกว่าอย่างชัดเจนในด้านใดบ้าง เพราะเหตุใด

แบบ SJF จะทำงานที่ต้องลงแรงน้อยเสร็จก่อน ทำให้ average waiting time และ average turnaround time ต่ำ แต่แบบ SJF จะมี CPU utilization ไม่เท่ากับ 1 นั่นคือจะมีบางจังหวะที่ CPU ว่างงาน

แบบ FCFS มี average waiting time และ average turnaround time สูงกว่า เนื่องจาก schedule process ที่มาก่อนเสมอ ทำให้ process ที่ใช้เวลาทำงานน้อยต้องรอนาน และ FCFS มี CPU utilization เท่ากับ 1 นั่นคือ FCFS สามารถจัดงานให้ CPU มีงานทำตลอด

ส่วนที่ 3

- 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
- scheduling algorithm ใดเป็นผลดีกับ CPU bound process ดังกล่าวมากกว่า เพราะอะไร

แบบ FCFS เพราะมันจะได้เริ่มทำงานตามลำดับที่มันเข้าไป ซึ่งทำให้ waiting time และ turnaround time ของ CPU bound process น้อย ในขณะที่แบบ SJF กว่ามันจะได้ทำงานอาจจะต้องรอนานมาก ๆ ไม่ว่ามันจะเข้าไปเป็นลำดับที่เท่าไรก็ตาม