แผนบริหารการสอนประจำบทที่ 5

การสลับโปรเซส (Process Scheduling)

วัตถุประสงค์

1. เข้าใจหลักการพื้นฐานกระบวนการสลับงานของหน่วยประมวลผล
2. สามารถอธิบายขั้นตอนวิธีการสลับงานของหน่วยประมวลผลในรูปแบบต่าง ๆ ได้
3. สามารถอธิบายเงื่อนไขและวิเคราะห์ความเหมาะสมของการสลับงานของแต่ละวิธีได้

เนื้อหา

1. หลักการพื้นฐานของการสลับงาน (Basic Concepts of Scheduling)
2. สภาพการณ์และเกณฑ์คุณภาพของการสลับงาน (Scheduling Circumstances and Criteria)
3. ขั้นตอนวิธีการสลับงาน (Scheduling Algorithms)
4. การวัดประสิทธิภาพของวิธีสลับงาน (Algorithm Evaluation)
5. การสลับงานภายใต้ระบบหลายหน่วยประมวลผล (Multiple-Processor Scheduling)

กิจกรรมการเรียนการสอน

1. บรรยาย
2. ฝึกปฏิบัติการระบบลีนุกซ์
3. วิดีโออนิเมชั่นการทำงานคอมพิวเตอร์
4. ค้นคว้าด้วยตัวเอง
5. รายงานหน้าชั้น
6. ฝึกปฏิบัติ

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. สไลด์การสอน
3. โปรแกรม google class room
4. โปรแกรม facebook

5. โปรแกรม VMware
6. โปรแกรมระบบปฏิบัติการลีนุกซ์ Ubuntu

การวัดผลและการประเมินผล

1. แบบฝึกหัดท้ายบท
2. การถามตอบในชั้นเรียน
3. ใบงาน
4. สอบเก็บคะแนน

บทที่ 5

การสลับโปรเซส (Process Scheduling)

ในบทนี้กล่าวถึงการสลับโปรเซสให้แก่หน่วยประมวลผล เพื่อตอบสนองความต้องการใช้งานหลายโปรแกรมในคราวเดียวกัน ซึ่งนับว่าเป็นเป้าหลักในการออกแบบระบบปฏิบัติการในยุคปัจจุบัน เนื้อหาในบทเริ่มจากหลักการพื้นฐานของการสลับงาน การควบคุมบริบทของแต่ละโปรเซสให้สามารถทำงานได้ต่อเนื่องเมื่อจำเป็นต้องคืนหน่วยประมวลผลให้แก่โปรเซสอื่น รายละเอียดขั้นตอนวิธีการสลับโปรเซสซึ่งมีหลายวิธี และมีสิทธิภาพในด้านต่าง ๆ แตกต่างกัน การวัดประสิทธิภาพของขั้นตอนวิธีสลับโปรเซส และความเข้าใจการสลับโปรเซสภายใต้ระบบที่มีหน่วยประมวลหลายชุด

5.1 หลักการพื้นฐานของการสลับงาน (Basic Concepts of Scheduling)

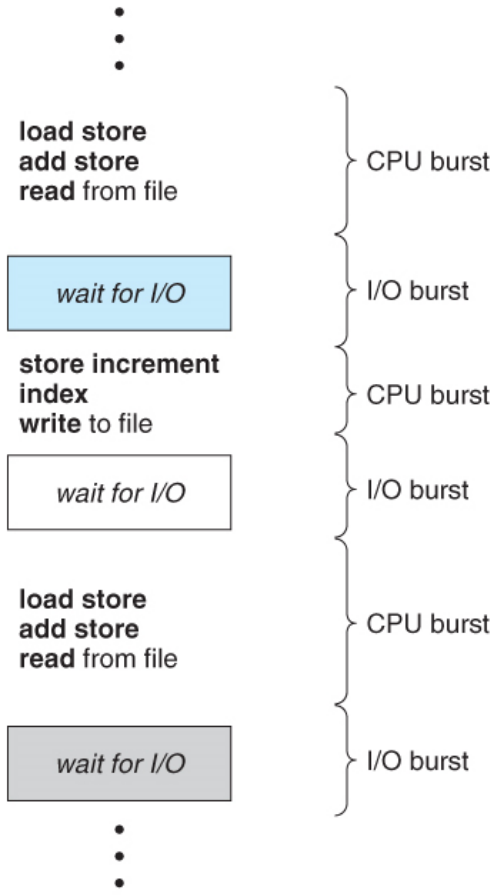
การจัดการการสลับการทำงานของการประมวลผลมีความสำคัญต่อการตอบสนองความต้องการในการทำงานระบบการทำงานหลายงานในเวลาเดียวกัน (multi-programmed operating system) ทรัพยากรของระบบคอมพิวเตอร์ที่มีอยู่อย่างจำกัด จำเป็นต้องถูกจัดสรรสลับกันใช้งานระหว่างโปรแกรมต่าง ๆ ที่ถูกสั่งงานให้ประมวลผล การสลับงานนั้นต้องคำนึงถึงสิ่งดังต่อไปนี้ ความเท่าเทียม ระดับความสำคัญของงาน ประสิทธิภาพการใช้งานหน่วยประมวลผล และภาระในการจัดการ วิธีการสลับงานมีหลายวิธี และมีความเหมาะสมในลักษณะงานที่แตกต่างกันซึ่งจะได้กล่าวในลำดับต่อไป

การประมวลผลภายใต้ระบบคอมพิวเตอร์ที่มีโครงสร้างทางฮาร์ดแวร์แบบหน่วยประมวลผลเดี่ยว (single-processor system หรือ single CPU) การทำงานของ ซีพียู สามารถประมวลผลได้เพียงโปรแกรมเดียวในช่วงเวลาหนึ่ง ๆ อย่างไรก็ตามมักพบว่า ซีพียู ตกอยู่ในสภาวะ “ว่าง” เพื่อรอให้ส่วนอื่น ๆ เช่น อุปกรณ์อินพุทเอาท์พุท (I/O device) ทำงานหลังจากมีการรับส่งข้อมูลกัน ดังนั้นเพื่อให้การใช้งานในช่วงเวลาว่างของซีพียู ให้เป็นประโยชน์ จึงนำเอาโปรแกรมอื่นเข้ามาประมวลผลควบคู่กันไป ทำให้ทรัพยากรต่าง ๆ รวมถึง ซีพียู หรือ หน่วยควมจำ ของระบบคอมพิวเตอร์ถูกโปรแกรมจำนวนหนึ่งเข้าครอบครองและใช้งาน จึงเป็นการสลับกันทำงานระหว่างโปรแกรม ซึ่งจะกระทำสลับกันไปต่อเนื่อง และเป็นการเพิ่มประสิทธิภาพการใช้งานซีพียู (CPU utilization) ให้ดียิ่งขึ้น

ความต้องการสลับงานให้แก่ ซีพียู จำเป็นต้องพิจารณารายละเอียดลักษณะการประมวลผลที่เกิดขึ้นในวงรอบของการประมวลผลในหนึ่งโปรเซส ซึ่งมีรายละเอียดดังนี้

5.1.1 วงรอบการประมวลผลโปรเซส (Cycle of Process Execution)

ในวงรอบของการประมวลผลโปรเซส (cycle of process execution) ประกอบด้วยช่วงเวลาการทำงานของ (CPU burst) และช่วงเวลาในการคอยให้อุปกรณ์อินพุทเอาท์พุททำงาน (I/O burst) ซึ่งเป็นช่วงของการถ่ายโอนข้อมูลเข้าหรือออกไปยังอุปกรณ์อินพุทเอาท์พุท (R.Teodor, 2015) วงรอบการประมวลผลหนึ่งโปรเซส เริ่มด้วย CPU burst และตามด้วย I/O burst สลับต่อเนื่องกันไปจนกระทั่งโปรแกรมสิ้นสุด ดังแสดงในรูปที่ 5-1



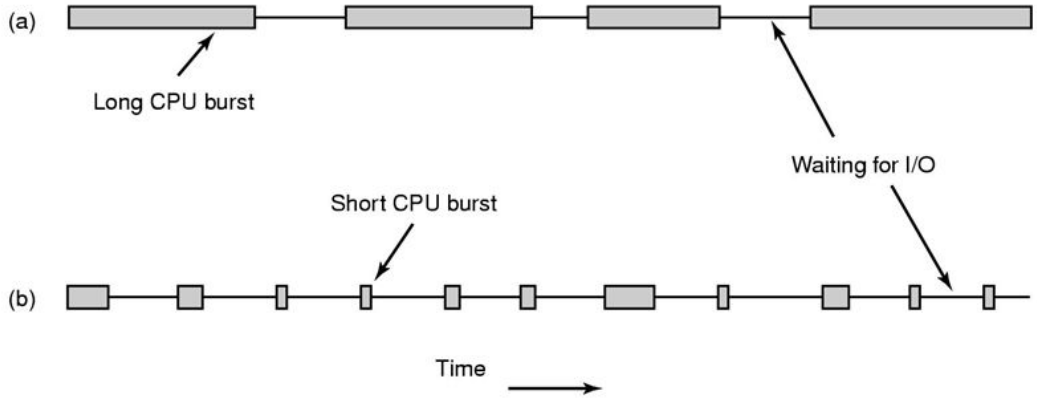
รูปที่ 5-1: ลำดับการการทำงานซีพียู และอุปกรณ์อินพุทเอาท์พุท(CPU and I/O burst)

ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

จะเห็นได้ว่าภาระของ ซีพียู ทำการนำเข้าคำสั่งจากหน่วยความจำหลัก มาประมวลผลทีละคำสั่ง ซึ่งถือเป็นของ CPU burst เมื่อใดมีคำสั่ง สั่งการให้อ่านหรือเขียนไฟล์ซึ่งถูกจัดเก็บไว้ในฮาร์ดดิสก์ โดยฮาร์ดดิสก์หรือแหล่งเก็บข้อมูลขนาดใหญ่ถือเป็นอุปกรณ์อินพุทเอาท์พุท ดังนั้นระบบปฏิบัติการจึงทำการติดต่อไปยังอุปกรณ์อินพุทเอาท์พุท เพื่อทำการถ่ายโอนข้อมูลซึ่งเป็นช่วงเวลาของการคอย I/O burst (Garrido, 2011)

5.1.2 ลักษณะโปรแกรมในการใช้ซีพียูและอุปกรณ์อินพุทเอาท์พุท (CPU and I/O bounds)

การพิจารณาลักษณะโปรแกรม การใช้งานซีพียูหรือการใช้งานอุปกรณ์อินพุทเอาท์พุท จะทำให้การเลือกวิธีในการสลับงานทำได้อย่างถูกต้องและมีประสิทธิภาพ การพิจารณาคือ โปรแกรมใดที่การทำงานส่วนใหญ่เป็นการประมวลผลของ ซีพียู โดยใช้เวลานาน (long CPU burst) มากกว่าช่วงเวลาของ I/O burst ดังแสดงในรูปที่ 5-2(a) จะสามารถเรียกโปรแกรมนี้ได้ว่าเป็นโปรแกรมที่มีลักษณะ CPU bound ในทางกลับกัน หากโปรแกรมใดมีช่วงเวลาส่วนใหญ่เป็นการคอยการถ่ายโอนข้อมูล (I/O burst) นานกว่าช่วงเวลาการประมวลผลของ ซีพียู (short CPU burst) แล้ว โปรแกรมนี้มีลักษณะ I/O bound ดังแสดงในรูปที่ 5-2(b)

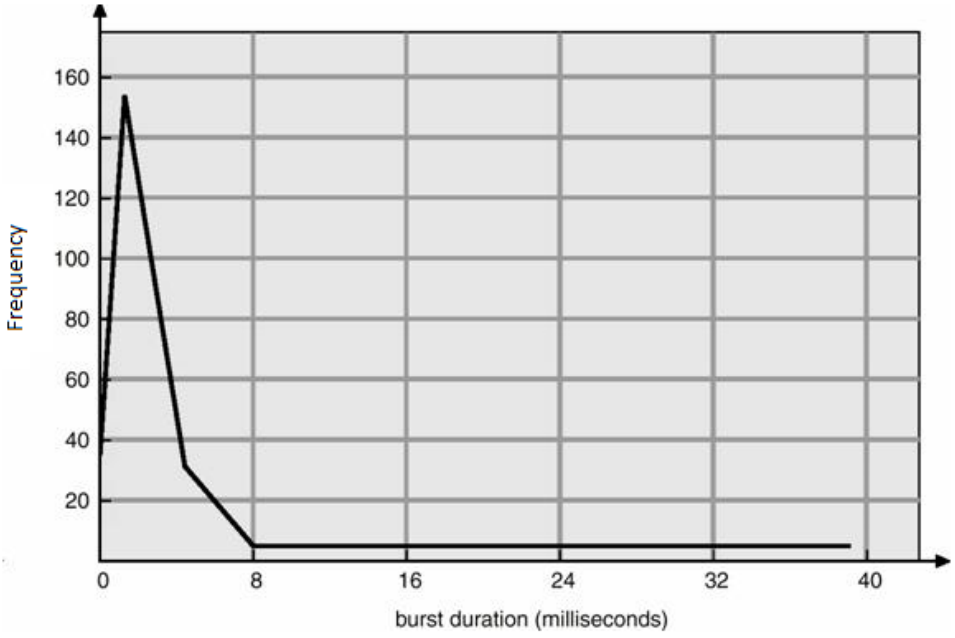


รูปที่ 5-2: CPU and I/O Bounds (a) CPU bound (b) I/O bound

ที่มา: (E.Baird, 2015)

จากการวัดและสังเกตุพฤติกรรมของการทำงานโปรเซสอย่างกว้างขวางไม่จะเป็นในระดับโปรเซสภายในระบบคอมพิวเตอร์เดียวกัน หรือโปรเซสระหว่างระบบคอมพิวเตอร์ พบว่าการกระจายของจำนวนโปรเซสที่มีช่วงเวลา CPU burst ส่วนใหญ่เป็นแบบ short CPU burst มากกว่าจำนวนโปรเซสที่มีช่วงเวลาการทำงานแบบ long CPU burst

จากรูปที่ 5-3 จะเห็นได้ว่าอัตราส่วนจำนวนโปรเซสที่มีพฤติกรรมการประมวลผลแบบ short CPU-burst มีแนวโน้มสูงในลักษณะเอ็กซ์โพเนนเชียล (exponential) ซึ่งช่วงเวลาที่ใช้ในการประมวลผลส่วนใหญ่อยู่ที่ประมาณ 2 milliseconds เท่านั้น



รูปที่ 5-3: การกระจายช่วงเวลาการประมวลผล (Histogram of CPU-burst Durations)

ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

5.1.3 ผู้จัดการสลับงานแก่หน่วยประมวลผล (CPU scheduler)

เมื่อใดที่ซีพียูอยู่ในสภาวะว่าง (waiting state) ระบบปฏิบัติการจะทำการสลับเอาโปรเซสจากโปรแกรมอื่นซึ่งอยู่ใน ready queue เข้ามาประมวลผลแทน ผู้จัดการสลับงานคือ short-term scheduler (ดู **Error! Reference source not found.** และ **Error! Reference source not found.** ประกอบ) การเลือกโปรเซสใน ready queue ไม่จำเป็นต้องเป็นลักษณะ “มาก่อนได้ก่อน” First-in-First-Out (FIFO) เสมอไป แต่ขึ้นอยู่ขั้นตอนวิธีที่เลือกใช้

5.2 สภาพการณ์และเกณฑ์คุณภาพของการสลับงาน (Scheduling Circumstances and Criteria)

5.2.1 สภาพการณ์ในการสลับงาน (Scheduling circumstances)

ระบบปฏิบัติการกระทำการสลับโปรเซสเมื่อเหตุการณ์หนึ่งต่อไปนี้เกิดขึ้น

1) เมื่อโปรเซสเปลี่ยนสถานะจาก running state เป็น waiting state ตัวอย่างเช่นเมื่อโปรเซส เรียกใช้อุปกรณ์อินพุทเอาท์พุทเป็นต้น

2) เมื่อโปรเซสเปลี่ยนสถานะจาก running state เป็น ready state ตัวอย่างเช่นเมื่อโปรเซสถูกขัดจังหวะ เป็นต้น

3) เมื่อโปรเซสเปลี่ยนสถานะจาก waiting state เป็น ready state ตัวอย่างเช่นเมื่ออุปกรณ์อินพุทเอาท์พุททำงานเสร็จสมบรูณ์และส่งสัญญาณให้แก่กลับมายังโปรเซส เป็นต้น และ

4) เมื่อโปรเซสทำงานเสร็จสิ้นสมบรูณ์ และยุติการทำงาน ด้วยการออกจากระบบ

5.2.2 เกณฑ์การสลับงาน (Scheduling criteria)

เกณฑ์การสลับงาน (scheduling criteria) เป็นการพิจารณาประสิทธิภาพของวิธีการสลับการทำงาน โดยมีข้อพิจารณาดังต่อไปนี้

1) Average waiting time: เวลาเฉลี่ยที่โปรเซสจะรอคอยภายใน ready queue ก่อนที่จะได้การประมวลผล เวลาดังกล่าวนี้ควรมีค่าน้อย ซึ่งบ่งบอกถึงประสิทธิของวิธีการสลับงานให้แก่ซีพียู

2) CPU utilization: ประสิทธิภาพการใช้ซีพียู ให้ประมวลได้มากที่สุด ในทางทฤษฎี ค่าประสิทธิภาพสามารถวัดเป็นร้อยละ 0 – 100 แต่ในทางปฏิบัติแล้วประสิทธิภาพการใช้งานอยู่ระหว่างร้อยละ 40 – 90

3) Throughput: ความสัมฤทธิ์ผลของงาน สามารถวัดได้จากจำนวนงานที่ทำสำเร็จภายในหนึ่งหน่วยเวลา เช่น 10 โปรเซส เสร็จภายใน 1 วินาที

4) Turnaround time: เวลาที่ใช้ประมวลผลในหนึ่งโปรเซส โดยเวลารวมตั้งแต่การนำเข้าชุดคำสั่ง การรอใน ready queue การประมวลผลใน ซีพียู และการรอการทำงานของอุปกรณ์อินพุทเอาท์พุท

5.3 ขั้นตอนวิธีการสลับงาน (Scheduling Algorithms)

ขั้นตอนวิธีในการสลับงานมีหลายหลายวิธี แต่มีเป้าหมายเดียวกันคือต้องการเลือกโปรเซสที่อยู่ใน ready queue ให้ได้อย่างมีประสิทธิภาพตามเกณฑ์คุณภาพ ในที่นี้อธิบาย 6 ขั้นตอนวิธีดังต่อไปนี้

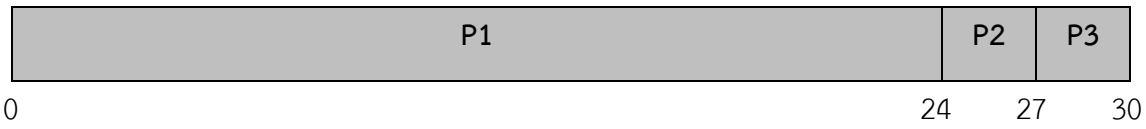
5.3.1 การสลับงานแบบ “มาก่อน ได้รับบริการก่อน” (First-Come, First-Served Scheduling : FCFS)

FCFS เป็นวิธีการที่เรียบง่ายที่สุด คือ โปรเซสใดที่ถูกบรรจุลงใน ready queue เป็นลำดับต้นก็จะถูกนำเข้าสู่ซีพียู เพื่อทำการประมวลผลก่อนโปรเซสที่มาทีหลัง (B.L.Stuart, 2008) โดยการต่อแถวเพื่อคอยรับการประมวลผลกระทำโดย บริเวณหัวขบวนของคิว (header) ของ ready queue ชี้ไปยังตำแหน่ง Process Control Block (PCB) ของโปรเซสแรกสุดของคิว จากนั้น PCB ของโปรเซสแรกนี้ ก็ชี้ไปยังตำแหน่งของ PCB ของโปรเซสอื่น เป็นเช่นนี้ตามลำดับกระทั่งถึง PCB ของโปรเซสสุดท้ายภายในคิว นอกจากนี้บริเวณตัวบ่งชี้ท้ายขบวนของคิว (tail) ของ reader queue ทำการชี้ไปยังตำแหน่ง PCB ของโปรเซสอันสุดท้ายในคิว

อีกด้วย ดังแสดงไว้ใน**Error! Reference source not found.** ซึ่งเกิดเป็นคิวแบบ link-list ในลำดับแบบ First – in – First – Out (FIFO) การพิจารณาประสิทธิภาพด้านเวลารอคอยเฉลี่ย (average waiting time : AWT) สามารถคำนวนได้ ตามตัวอย่างดังนี้

ตัวอย่างที่ 5-1: การสลับโปรเซสด้วยวิธี FCFS

Process	CPU Burst Time
P1	24
P2	3
P3	3



เวลารอคอยเฉลี่ย (AWT) = (0 + 24 + 27)/3

= 17 milliseconds

จากตัวอย่างที่ 5-1 โปรเซสใน ready queue มีจำนวน 3 โปรเซส โดยเรียงกันตามลำดับคือ P1, P2 และ P3 ซึ่งใช้เวลาในการประมวลผล (CPU burst) เป็น 24, 3 และ 3 milliseconds ตามลำดับ เมื่อนำโปรเซสทั้งหมดมาวาดเป็นแผนผังการทำงาน (Gantt chart) ตามวิธีการสลับโปรเซสแบบ FCFS จะได้ว่า P1 เริ่มประมวลผลที่เวลา 0 millisecond, P2 เริ่มทำการประมวลผลหลังจาก P1 ทำงานเสร็จที่เวลา 24 millisecond และ P3 เริ่มทำการประมวลผลที่เวลา 27 millisecond ดังนั้นเวลาเฉลี่ยของการรอคอยการประมวลผล (AWT) ของทั้งสามโปรเซส คือ 17 milliseconds

อย่างไรก็ตามหากพิจารณาลำดับการเข้าสู่ ready queue ของโปรเซสใหม่โดยให้เป็น P2, P3 และ P1 จะสามารถเขียนแผนผังการทำงานใหม่ดังนี้

และสามารถคำนวนเวลารอคอยเฉลี่ยเป็น

AWT = (6 + 0 + 3)/3 = 3 milliseconds

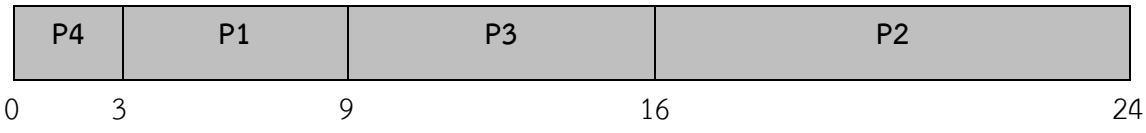
จะเห็นได้ว่าเมื่อเรียงลำดับโปรเซสใหม่ โดยให้โปรเซสที่มีเวลาการประมวลผลสั้น ได้รับการประมวลผลก่อน จะส่งผลให้เวลารอคอยเฉลี่ยของโปรเซสทั้งหมด น้อยลงอย่างมาก

5.3.2 การสลับงานแบบ “งานน้อยสุด ทำก่อน” (Shortest-Job-First Scheduling : SJF)

จากตัวอย่างที่แล้ว เมื่อโปรเซสจัดเรียงใหม่ โดยให้โปรเซสที่มีเวลาประมวลผลน้อยได้อยู่ในลำดับต้นของคิว (Reddy, 2009) อย่างไรก็ตามเวลาที่ใช้ในการประมวลผล (CPU burst) ที่แท้จริงนั้น ทราบจากการวัดค่าซึ่งโปรเซสนั้น ๆ ต้องผ่านการทำงานแล้วเสียก่อน ดังนั้น SJF จึงเป็นวิธีการใหม่ที่เลือกโปรเซสจาก ready queue โดยพิจารณาจากค่าประมาณการของ CPU burst ซึ่งเป็นค่าประมาณการเวลาประมวลผลล่วงหน้า (predicted next CPU burst) ของโปรเซสนั้น ๆ โดยวิธีการนำเอาเวลาการประมวลผลในรอบก่อน ๆ ที่ผ่านมา มาพิจารณาประกอบ วิธีการแบบ SJF ทำการจัดเรียงโปรเซสจากค่าเวลา CPU burst ที่สั้นที่สุดได้ทำการประมวลผลก่อน แต่หากโปรเซสใดมีค่าประมาณการเท่ากันจะใช้วิธีการเลือกแบบ FCFS แทน พิจารณาตัวอย่างต่อไปนี้

ตัวอย่างที่ 5-2: การเลือกโปรเซสด้วยวิธี SJF

Process	Predicted CPU Burst Time
P1	6
P2	8
P3	7
P4	3



AWT = (3 + 16 + 9 + 0)/4 = 7 milliseconds

จากตัวอย่างที่ 5-2 โปรเซสใน ready queue มีจำนวน 4 โปรเซส พร้อมกับเวลา CPU burst ซึ่งได้จากการประมาณการด้วยการมองย้อนกลับไปในการใช้เวลาเพื่อประมวลผลของแต่ละโปรเซสในรอบที่ผ่านมา เมื่อ SJF ทำการคำนวนเวลาประมวลผลเสร็จสิ้น ก็จะสามารถเลือกโปรเซสซึ่งเป็นไปตามลำดับ

ของเวลาที่สั้นที่สุด ดังปรากฏตามแผนผังงานคือ P4, P1, P3 และ P2 เมื่อคำนวนค่าเฉลี่ยเวลารอคอย (AWT) ได้เป็น 7 milliseconds

ถึงแม้วิธีการเลือกโปรเซสจาก ready queue ด้วยการเลือกเวลาประมวลผลของโปรเซสที่สั้นที่สุดก่อน และไล่เลียงไปตามลำดับนั้น ไม่มีความซับซ้อนแต่อย่างใด แต่ความซับซ้อนมาอยู่ที่การประมาณการค่าเวลาประมวลผลในรอบถัดไปของแต่ละโปรเซส ซึ่งเป็นภารกิจที่ผู้จัดการสลับงาน (short-term scheduler) ต้องคำนวน โดยวิธีการคำนวนเวลาประมวลผลของหนึ่งโปรเซสมีรายละเอียดดังนี้

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$$

โดยที่

τ_{n+1} คือ เวลาประมาณการของการประมวลผลในรอบถัดไป (predicted next CPU burst)

t_n คือ เวลาการประมวลผลที่เกิดขึ้นจริงในรอบล่าสุด (recent actual CPU burst)

τ_n คือ เวลาประมาณการของการประมวลผลในอดีต (history of predicted CPU burst)

ทั้งนี้สามารถกำหนดน้ำหนักให้กับส่วนเวลาการประมวลผลที่เกิดขึ้นจริงในรอบล่าสุด และส่วนเวลาประมาณการของการประมวลผลในอดีตได้ด้วยพารามิเตอร์ α โดยที่ $0 \leq \alpha \leq 1$

พิจารณาการคำนวนค่าประมาณการของเวลาประมวลผลของโปรเซสดังรายละเอียดดังนี้

ตารางที่ 5-1: ค่าประมาณการเวลาประมวลผล (predicted CPU burst)

n	0	1	2	3	4	5	6	7	8
Predicted CPU burst	10	8	6	6	5	9	11	12	...
Actual CPU burst	6	4	6	4	13	13	13	13	...

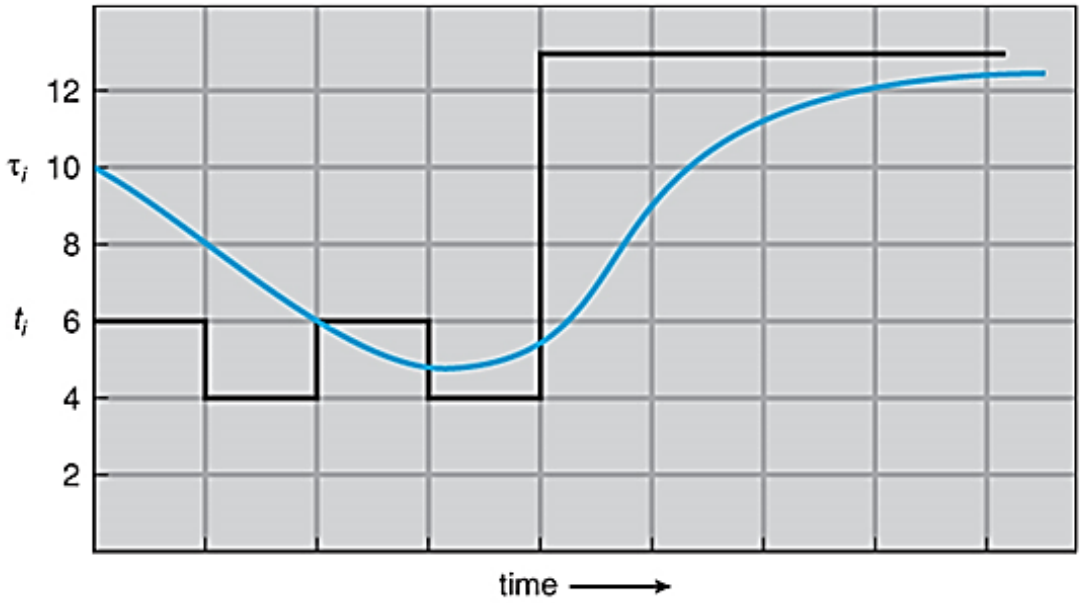
Time ⟶

ตารางที่ 5-1 เป็นตัวอย่างการคำนวนเวลาประมาณการการประมวลผลของโปรเซสหนึ่ง โดยกำหนดให้ค่าประมาณการเริ่มต้นของเวลาการประมวลผลเป็น 10 milliseconds ($\tau_0 = 10$) แต่เวลาการประมวลผลที่เกิดขึ้นจริงเป็น 6 milliseconds ($t_0 = 6$) ฉนั้นเราสามารถคำนวนหาค่าประมาณการเวลาในการประมวลผลของโปรเซสนี้ในรอบต่อไป (t_1) ได้ดังนี้

เมื่อ n = 1 และให้น้ำหนักเท่ากัน $\alpha = 0.5$

$$\begin{aligned}\tau_1 &= \alpha t_0 + (1-\alpha)\tau_0 \\ &= (0.5)(6) + (1-0.5)(10) \\ &= 8 \text{ milliseconds}\end{aligned}$$

อย่างไรก็ตามเมื่อโปรเซสได้รับการประมวลผล เวลาที่ใช้จริงเป็น 4 milliseconds เท่านั้น การประมาณการในรอบต่อไป (t_2) ก็กระทำเช่นนี้ต่อไปเรื่อย ๆ กระทั่งยุติโปรเซสดังกล่าว เมื่อนำผลในตารางมาวาดกราฟจะได้ตามรูปที่ 5-4



รูปที่ 5-4: เปรียบเทียบการ predicted CPU burst และ actual CPU burst

นอกจากนี้ SJF เป็นขั้นตอนวิธีที่มีคุณสมบัติสำคัญคือ preemptive หรือ nonpreemptive

1) Preemptive คือ SJF เป็นลักษณะที่ยอมให้โปรเซสที่กำลังประมวลผลอยู่ในปัจจุบันถูกขัดจังหวะโดยโปรเซสอื่น และเข้าแทรกการประมวลผลได้ โดยโปรเซสปัจจุบันจะรอจนกระทั่งการขัดจังหวะเสร็จสิ้นจึงกลับมาประมวลผลต่อ

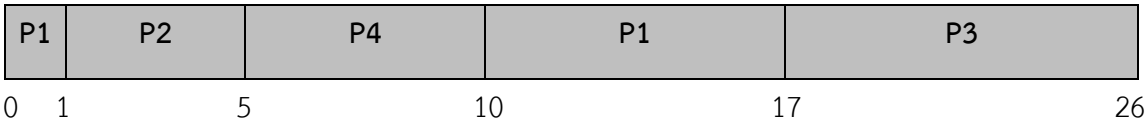
2) Nonpreemptive คือ SJF เป็นลักษณะที่โปรเซสที่กำลังประมวลผลอยู่ในปัจจุบันต้องทำงานจนเสร็จสิ้นรอบของตัวเอง โดยไม่ยินยอมให้โปรเซสอื่นเข้าขัดจังหวะ และแทรกการทำงานได้

พิจารณากรณีวิธี SJF แบบมีคุณลักษณะ preemptive จาก

ตัวอย่างที่ 5-3 ส่วนในกรณีที่เป็นรูปแบบ nonpreemptive นั้น เป็นดังตัวอย่างที่ 5-2

ตัวอย่างที่ 5-3: SJF แบบ Preemptive

Process	Arrival Time	Predicted CPU Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5



AWT = [(10-1) + (1-1) + (17-2) + (5-3)]/4 = 6.5 milliseconds

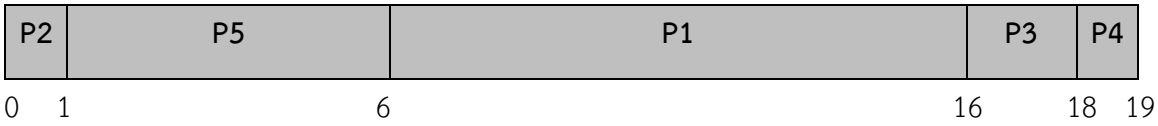
ในตัวอย่างนี้นำเวลาที่โปรเซสเข้าสู่ ready queue (arrival time) มาประกอบในการพิจารณา พบว่าในช่วงเวลาเริ่มต้น (เวลา t = 0) มีเพียง P1 ที่อยู่ในคิว ดังนั้น P1 จึงถูกประมวลผลก่อน ในขณะที่ซีพียู กำลังประมวลผลอยู่นั้น P2 เข้าสู่คิว (เวลา t = 1) ผู้จัดการสลับงานในวิธี SJF พิจารณาเวลาประมาณการประมวลผล (predicted CPU burst time) ที่จะใช้ในการประมวลผลสั้นกว่า P1 จึงทำการขัดจังหวะ P1 เพื่อแทรกให้ P2 เข้าสู่การประมวลผล ในขณะที่ P3 และP4 เข้าสู่คิวในเวลา t = 2 และ t = 3 ตามลำดับ อย่างไรก็ตามทั้งสอง โปรเซสมีช่วงเวลาประมาณการประมวลมากกว่า P2 จึงไม่เกิดการขัดจังหวะ ผู้จัดการสลับงานจึงบรรจุทั้งสองโปรเซสไว้ในคิว เมื่อ P2 ทำงานเสร็จสิ้น (เวลา t = 5) โปรเซส P4 จึงได้รับการประมวลผลถัดมา จากนั้นเป็นโปรเซส P1 และสุดท้ายคือโปรเซส P3 ซึ่งเรียงลำดับตามเวลาประมาณการประมวลผล เนื่องจากวิธีการสลับงานนี้อนุญาตให้มีการขัดจังหวะโปรเซสที่กำลังทำงาน จึงเป็นไปตามคุณสมบัติ preemptive

5.3.3 การสลับงานแบบ “ลำดับตามความสำคัญ” (Priority Scheduling : PS)

PS เป็นวิธีการเลือกโปรเซสโดยพิจารณาความสำคัญ (priority) ของโปรเซสแต่ละตัว (Das, et al., 2010) ค่าความสำคัญมีความสัมพันธ์ไปตามลักษณะประเภทของโปรเซสนั้น ๆ โดยโปรเซสที่มีค่าความสำคัญสูงสุดใน ready queue จะถูกเลือกให้ได้รับการประมวลผลก่อน ถัดจากนั้นจะเป็นโปรเซสที่มีระดับความสำคัญรองลงมา ในกรณีที่มีโปรเซสมากกว่าหนึ่งตัวมีระดับความสำคัญเท่ากัน จะใช้วิธีการเลือกแบบ FCFS ระดับความสำคัญมักจะแสดงเป็นค่าตัวเลขระหว่าง 0 – 4,095 โดยหมายเลข 0 ถือเป็นระดับความสำคัญสูงสุด และหมายเลข 4,095 เป็นระดับความสำคัญต่ำสุด พิจารณาตัวอย่างด้านล่าง

ตัวอย่างที่ 5-4: การเลือกโปรเซสด้วยวิธี PS

Process	Priority	CPU Burst Time
P1	3	10
P2	1	1
P3	4	2
P4	5	1
P5	2	5



AWT = (6 + 0 + 16 + 18 + 1)/5 = 8.2 milliseconds

จากตัวอย่างที่ 5-4 วิธี PS นำระดับความสำคัญของโปรเซสมาพิจารณา โดยจัดให้โปรเซส P2 ซึ่งมีระดับความสำคัญสูงสุดในคิว ได้รับการประมวลผลก่อน ตามด้วยโปรเซสที่มีความสำคัญในลำดับถัดไป โดยไม่สนใจเวลาที่จะใช้ในการประมวลผล การเลือกโปรเซสในลักษณะนี้เหมาะสำหรับการตอบสนองต่อประเภทโปรแกรมที่ต้องตอบโต้กับผู้ใช้ เช่น VDO streaming เป็นต้น

การกำหนดค่าระดับความสำคัญ (priority defining) ให้แก่โปรเซสเป็นสิ่งสำคัญที่จะลำดับความสำคัญให้แต่ละโปรเซสได้อย่างถูกต้อง เพื่อผู้จัดการสลับงานทำงานได้อย่างเหมาะสมและการตอบสนองของโปรแกรมเป็นไปอย่างสมจริงทันเหตุการณ์ การกำหนดระดับความสำคัญสามารถทำได้ใน 2 รูปแบบคือ Internally และ Externally

1) Internally defined priority: เป็นการกำหนดระดับความสำคัญโดยคำนวนจากปัจจัยภายในเช่น ข้อจำกัดของเวลา (time limit), ความต้องการหน่วยความจำ (memory requirement), จำนวนของไฟล์ที่ใช้ (number of opened files) และ อัตราส่วนของค่าเฉลี่ย I/O burst ต่อ CPU burst

2) Externally defined priority: เป็นการกำหนดระดับความสำคัญจากปัจจัยภายนอก เช่น ความสำคัญของโปรเซส ซึ่งมักจะเป็นไปตามชนิดของโปรแกรม เป็นต้น

วิธีการเลือกโปรเซสแบบ PS ที่มีคุณสมบัติ preemptive และ nonpreemptive เช่นเดียวกัน กล่าวคือ ในกรณี preemptive หากมีโปรเซสใดที่มีระดับความสำคัญสูงกว่าโปรเซสที่กำลังประมวลผลอยู่ ระบบปฏิบัติการโดยผู้จัดการสลับงานจะขัดจังหวะและขอแทรกการทำงาน เพื่อให้โปรเซสใหม่ที่มีระดับความสำคัญสูงกว่าเข้าประมวลผลก่อน ในกรณีตรงข้าม nonpreemptive จะไม่ยินยอมให้มีการขัดจังหวะเพื่อแทรกการทำงาน แต่จะปล่อยให้โปรเซสที่กำลังประมวลผล ทำจนกระทั่งสิ้นสุดงาน

อย่างไรก็ตาม อาจมีปัญหาที่เกิดขึ้นกับวิธีสลับงานแบบ PS กล่าวคือ โปรเซสใดที่มีระดับความสำคัญต่ำมาก อาจค้างอยู่ในคิว (ready queue) โดยไม่ได้รับโอกาสในการประมวลผลเลย เนื่องจากมีโปรเซสอื่น ๆ ที่มีระดับความสำคัญสูงกว่าเข้ามาในคิวเสมอ ปัญหานี้สามารถแก้ไขได้โดยการเพิ่มค่าความสำคัญให้มากขึ้นตามลำดับ เมื่อเวลาผ่านไปในระยะหนึ่ง ๆ (เช่นทุก ๆ หนึ่งชั่วโมง) จนกระทั่งโปรเซสดังกล่าวมีค่าความสำคัญอยู่ในระดับสูงพอที่จะถูกประมวล

5.3.4 การสลับงานแบบ “สลับกันเป็นวงรอบ” (Round-Robin Scheduling : RR)

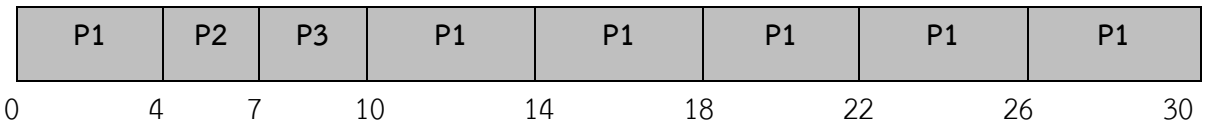
RR เป็นวิธีการสลับงานที่มีความคล้ายคลึงกับวิธี FCFS คือ โปรเซสใดที่เข้าสู่ ready queue ก่อน จะได้รับการประมวลผลก่อน ส่วนโปรเซสมาทีหลังจะได้รับการประมวลผลถัดไป อย่างไรก็ตาม วิธีการแบบ RR กำหนดช่วงเวลาการประมวลผลของแต่ละโปรเซสไว้ เรียกช่วงเวลานี้ว่า time quantum หรือ time slice โดยโปรเซสจะถูกประมวลผลนานไม่เกินช่วงเวลา time quantum ที่กำหนด หากงานของโปรเซสยังไม่เสร็จสิ้นสมบูรณ์ ผู้จัดการคิวจะทำการขัดจังหวะโปรเซส จากนั้นทำการสลับบริบท (context switch) ไปเก็บไว้ยัง PCB และนำโปรเซสดังกล่าวกลับมายัง ready queue อีกครั้ง (Saltzer & Kaashoek, 2009) เพื่อรอการประมวลผลในรอบต่อไป ทั้งนี้เพื่อเปิดโอกาสให้โปรเซสอื่นซึ่งรออยู่ในคิวได้รับการประมวลผลสลับกันไป ดังตัวอย่างที่ 5-5

ตัวอย่างที่ 5-5: การเลือกโปรเซสด้วยวิธี RR

Process	CPU Burst Time

P1	24
P2	3
P3	3

กำหนด Time quantum = 4 milliseconds



AWT = [(10 - 4) + 4 + 7]/3 = 5.67 milliseconds

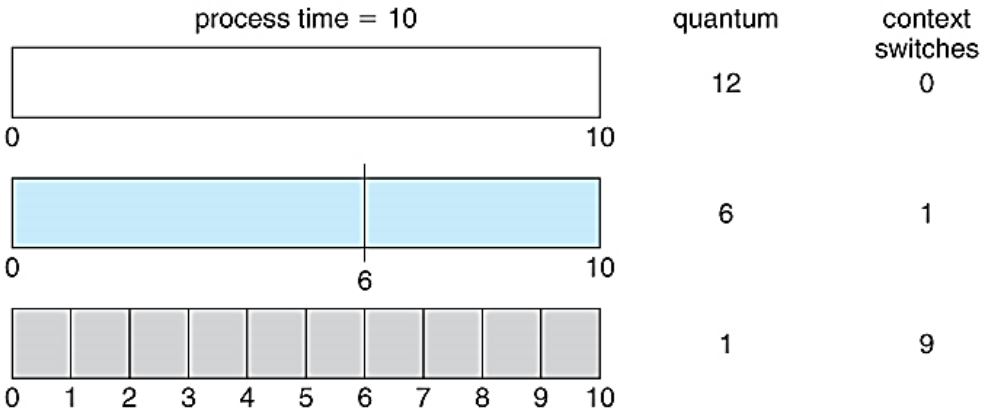
วิธีการเลือกโปรเซสแบบ RR ตามตัวอย่างที่ 5-5 จะเลือกโปรเซสลำดับแรกที่เข้าสู่คิว ในกรณีนี้คือ P1 เนื่องจาก time quantum ถูกกำหนดไว้ 4 milliseconds ดังนั้น P1 จึงได้รับการประมวลผลรอบแรกตามเวลาทำกำหนดโดยไม่สามารถทำงานให้เสร็จสมบูรณ์ได้ ผู้จัดการสลับงานทำการขัดจังหวะ P1 และนำ P2 เข้าสู่การประมวลผลต่อไป โดยนำ P1 ไปต่อท้ายคิว (ต่อท้าย P3) อย่างไรก็ตาม P2 และ P3 ใช้เวลาในการประมวลผลเพียง 3 milliseconds จึงทำงานเสร็จสมบรูณและยุติการทำงาน ในส่วนที่เหลือจึงเป็นการทำงานของ P1 เท่านั้น

วิธีการสลับงานแบบ RR จำเป็นต้องมีคุณลักษณะ preemptive เนื่องจากการควบคุมการสลับงาน จำเป็นต้องขัดจังหวะการทำงานของโปรเซสที่มี CPU burst time มากกว่า time quantum โดยเวลาที่ต้องรอคอย (waiting time) ในแต่ละรอบสามารถคำนวนได้จาก

$$(\mathrm{n}-1)\times \mathrm{q}$$

โดยที่ n คือ จำนวนโปรเซสใน ready queue และ q คือ quantum time (millisecond)

ดังนั้นคุณภาพการทำงานสลับงาน (performance) ของวิธีแบบ RR ขึ้นอยู่กับความสัมพันธ์ขององค์ประกอบ CPU burst time, quantum time และ context switch time พิจารณารูปที่ 5-5



รูปที่ 5-5: ความสัมพันธ์องค์ประกอบ CPU burst time, quantum time และ context switch time

ที่มา : (Silberschatz, Galvin, & Gagne, 2010)

จะเห็นโปรเซสหนึ่งต้องการเวลาในการประมวลผล (CPU burst time) 10 milliseconds หาก quantum time ถูกกำหนดไว้ที่ 12 milliseconds การขัดจังหวะและการสลับบริบทไม่เกิดขึ้น แต่ย่อมส่งผลให้ค่าเวลาที่ต้องคอยในคิวนานมาก หรือหากกำหนด quantum time ไว้น้อยเช่น 1 millisecond จะส่งผลให้มีการขัดจังหวะ และเสียเวลาในการสลับบริบทจำนวนมากเช่นกัน ในกรณีนี้สลับบริบทถึง 9 ครั้ง

5.3.5 การสลับงานแบบ “คิวหลายระดับ” (Multilevel Queue Scheduling : MQ)

MQ เป็นวิธีการสลับงานแบบพิจารณาระดับความสำคัญ (priority) เช่นเดียวกัน แต่ระดับความสำคัญถูกจัดเป็นกลุ่มความสำคัญไว้ใน ready queue ซึ่งแตกต่างจากวิธีของ PS ซึ่งระบุความสำคัญเป็นรายโปรเซส ทำให้การจัดการแบบ MQ ง่ายกว่า โดยโปรเซสจะถูกบรรจุให้อยู่ในกลุ่มคิวที่เหมาะสม เช่น กลุ่มโปรเซสทำงานเบื้องหน้า (foreground processes) และ กลุ่มโปรเซสทำงานเบื้องหลัง (background processes) (Dhotre, 2009)

กลุ่มโปรเซสทำงานเบื้องหน้า (Foreground processes) เป็นกลุ่มที่จำเป็นต้องได้รับการตอบสนองด้วยเวลารวดเร็ว เช่นเป็นประเภทที่ต้องตอบสนองการใช้งานของผู้ใช้ จึงมีความสำคัญสูงกว่ากลุ่มโปรเซสทำงานเบื้องหลัง (background process) ซึ่งอาจเป็นชุดคำสั่ง batch เช่นการสำรองข้อมูลที่สามารถประมวลผลได้เมื่อระบบคอมพิวเตอร์มีภารกิจน้อยลงในเวลากลางคืน เป็นต้น

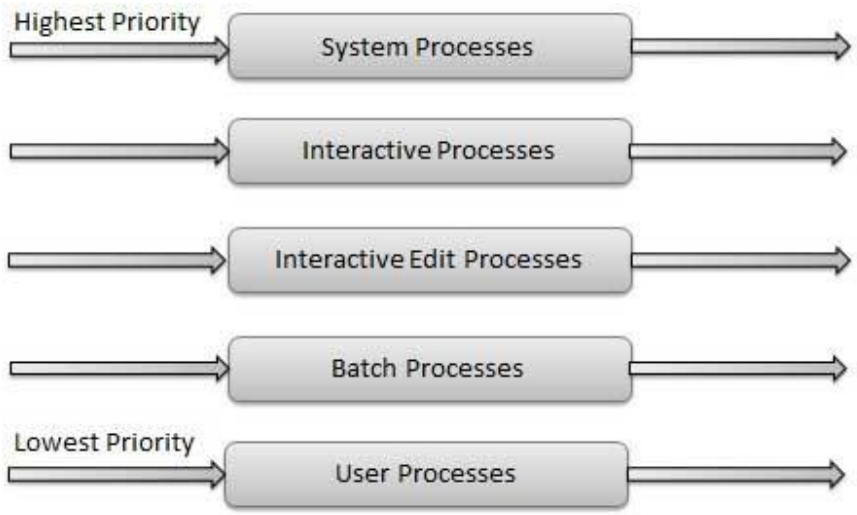
กลุ่มโปรเซสที่อยู่ในระดับเดียวกันนั้น ผู้จัดการสลับงานจะเลือกใช้วิธีการแบบ RR จนกระทั่งหมดในคิวกลุ่มเดียวกัน จึงเลื่อนมาสู่คิวที่มีระดับความสำคัญต่ำกว่าต่อไป ในการแบ่งกลุ่มอาจจัดให้มีหลายกลุ่ม ซึ่งระดับความสำคัญของแต่ละกลุ่มถูกกำหนดค่าไว้ชัดเจน และโปรเซสจะถูกจัดเข้ากลุ่มไว้อย่างถาวรไม่เปลี่ยนแปลง การจัดลำดับความสำคัญ สูง-ต่ำ ของกลุ่มโปรเซส ดังแสดงในรูปที่ 5-6 คือ

1) System processes: กลุ่มโปรเซสระบบ

2) Interactive processes: กลุ่มโปรเซสที่มีการตอบโต้กับผู้ใช้ เช่น เกมส์

3) Interactive editing processes: กลุ่มโปรเซสที่มีการตอบโต้กับผู้ใช้ในการทำงานประเภทเอกสาร เช่น MS-Word, MS-Excel

4) Batch processes: กลุ่มโปรเซสชุดคำสั่งงานประจำ เช่น การสำรองข้อมูล



รูปที่ 5-6: Multilevel Queue Scheduling (MQ)

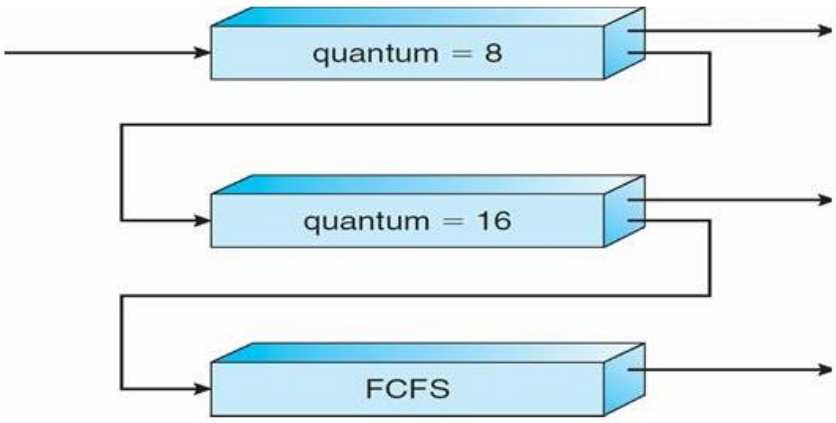
ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

นอกจากนี้วิธี MQ มีคุณลักษณะ preemptive เช่นกัน ซึ่งโปรเซสที่อยู่ในกลุ่มความสำคัญระดับสูงกว่าสามารถขัดจังหวะโปรเซสที่ระดับความสำคัญต่ำกว่า และแทรกเข้าประมวลผลได้

5.3.6 การสลับงานแบบ “คิวผลย้อนกลับหลายระดับ” (Multilevel Feedback Queue Scheduling : MFQ)

เนื่องจากวิธีการสลับงานแบบ MQ นั้น แต่ละกลุ่มคิวมีค่าระดับความสำคัญที่ตายตัว และโปรเซสถูกจัดเข้าประจำกลุ่มโดยไม่เปลี่ยนแปลง ดังนั้นโปรเซสไม่สามารถเปลี่ยนแปลงระดับความสำคัญข้ามกลุ่มได้ ซึ่งอาจไม่ยืดหยุ่นและไม่สอดรับกับความเป็นจริง ดังนั้นวิธี MFQ จึงปรับปรุงด้วยการอนุญาตให้โปรเซสสามารถเปลี่ยนแปลงระดับความสำคัญ โดยจัดให้เข้ากลุ่มที่แตกต่างกันในแต่ละรอบการประมวลผลได้ โดยพิจารณาจากเวลาการประมวลผล (CPU burst time) หากโปรเซสใดใช้เวลาของ ซีพียู มาก ก็จะถูกบรรจุเข้าสู่กลุ่มคิวที่มีระดับความสำคัญที่ต่ำลง จนกระทั่งกลุ่มต่ำสุดจะใช้วิธีแบบ FCFS กระทั่งสิ้นสุดงาน (Khurana, 2011)

ยกตัวอย่างเช่น กำหนดให้มีกลุ่มใน ready queue แบ่งระดับความสำคัญออกเป็น 3 กลุ่มคือ กลุ่ม 0, 1 และ 2 ผู้จัดการสลับงานจึงจัดให้โปรเซสในกลุ่มระดับความสำคัญสูงสุดในกลุ่ม 0 ทำการประมวลผลก่อน จนกระทั้งคิวของกลุ่มนี้ไม่มีโปรเซสเหลืออยู่ในคิว ผู้จัดการสลับงานจึงเลื่อนไปจัดการยังกลุ่มลำดับความสำคัญถัดไปดังรูปที่ 5-7



รูปที่ 5-7: Multilevel Feedback Queue Scheduling (MFQ)

ที่มา: (Silberschatz, Galvin, & Gagne, 2010, p. 198)

จากรูปที่ 5-7 คิวกลุ่ม 0 กำหนดเวลา quantum time = 8 milliseconds. หากโปรเซสใดประมวลผลไม่เสร็จภายในเวลาที่กำหนด ก็จะถูกย้ายข้ามกลุ่มไปต่อคิวที่มีระดับความสำคัญต่ำกว่า คือกลุ่ม 1 ซึ่งกำหนดให้มีเวลาประมวลของนานขึ้นเป็น quantum time = 16 milliseconds. เพื่อรอประมวลผลต่อไป หากโปรเซสดังกล่าวประมวลผลยังไม่สมบรูณ์ในชั้นนี้ ก็จะถูกย้ายไปต่อคิวกลุ่ม 2 อีกครั้งซึ่งกรณีนี้เป็นกลุ่มความสำคัญระดับสุดท้าย ผู้จัดการสลับงานจึงใช้วิธีเป็นแบบ FCFS ซึ่งโปรเซสจะได้รับการประมวลผลจนกระทั่งเสร็จงาน

5.4 การวัดประสิทธิภาพของวิธีสลับงาน (Algorithm Evaluation)

การวัดประสิทธิภาพของวิธีการสลับงาน เพื่อทราบถึงความเหมาะสมของวิธีการสลับงานนั้น ๆ เปรียบเทียบกับเงื่อนไข และบริบทภารกิจของระบบปฏิบัติการ การวัดประกอบด้วยด้านต่าง ๆ เช่น วัดประสิทธิภาพการใช้หน่วยประมวลผล (CPU utilization) ภายใต้ข้อกำหนดว่า เวลาตอบสอนงโปรเซส (response time) ต้องไม่เกิน 1 วินาที เป็นต้น

ในความเป็นจริงการวัดประสิทธิภาพทำได้หลายวิธี เช่น รูปแบบการคำนวน (Deterministic modelling), การวิเคราะห์ด้วยทฤษฎีคิว (Queueing modelling), การจำลองสถานการณ์ (Simulation) และ การพัฒนาจริง (Implementation)

5.4.1 รูปแบบการคำนวน (Deterministic modelling)

การวัดประสิทธิภาพวิธีการสลับงานโดยใช้การวิเคราะห์ผ่านการคำนวนเป็นที่นิยมแพร่หลาย การคำนวนใช้องค์ประกอบสองอย่างคือ กำหนดวิธีการสลับงานที่ต้องการวิเคราะห์ และตัวอย่างภารงาน การวิเคราะห์ประสิทธิภาพพิจารณาจากตัวเลข หรือผลการคำนวน ที่สามารถบ่งชี้ประสิทธิภาพด้านที่สนใจ ดังตัวอย่างการคำนวนต้านล่าง

ตัวอย่างที่ 5-6: การวิเคราะห์ประสิทธิภาพของวิธีการสลับงานด้วยวิธีคำนวน

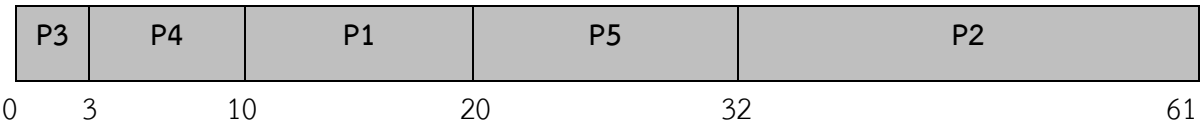
จงหา Average Waiting Time (AWT) ที่น้อยที่สุดจากการสลับงานแบบ FCFS, SJF และ RR โดยกำหนดให้ quantum time = 10 milliseconds

Process	CPU Burst Time
P1	10
P2	29
P3	3
P4	7
P5	12

1) วิธีแบบ FCFS

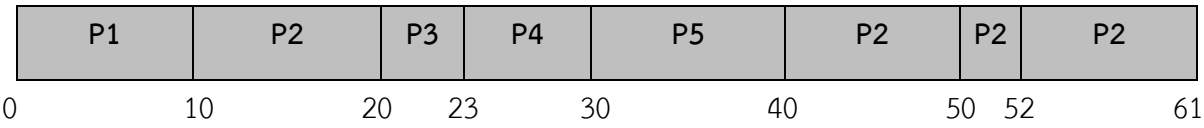
AWT = (0+10+39+42+49+61)/5 = 28 milliseconds

2) วิธีแบบ SJF



AWT = (10+32+0+3+20)/5 = 13 milliseconds

3) วิธีแบบ RR



AWT = (0+32+20+23+40)/5 = 23 milliseconds

ในสภาพแวดล้อมที่ระบบปฏิบัติการมีภาระงานตามที่กำหนด ผลการวิเคราะห์ผลลัพธ์จากการคำนวนของทั้งวิธี พบว่าวิธิการสลับงานแบบ SJF มีประสิทธิภาพด้านของเวลารอคอยเฉลี่ยต่ำที่สุด คือ 13 milliseconds ประสิทธิภาพลำดับที่สองคือ วิธีแบบ RR ซึ่งมีค่า AWT เป็น 23 milliseconds และวิธีที่ FCFS มีค่า AWT มากที่สุดคือ 28 milliseconds

5.4.2 การวิเคราะห์ด้วยทฤษฎีคิว (Queueing modelling)

เนื่องจากจำนวนโปรเซสที่ทำงานอยู่ในระบบปฏิบัติการ สามารถเปลี่ยนแปลงตลอดเวลา ไม่ว่าจะเป็นประเภท ระดับความสำคัญ CPU-burst จึงเป็นการยากที่ใช้การคำนวนเพื่อให้ได้ผลลัพธ์ที่ถูกต้องในทุก ๆ สถานการณ์ อย่างไรก็ตามเราสามารถประมาณการค่าเฉลี่ยของประสิทธิภาพในด้านต่าง ๆ ได้ เช่น ปริมาณงานทำสำเร็จในหน่วยเวลาที่กำหนด (throughput), คุณภาพการใช้งานหน่วยประมวลผล (CPU utilization) และ เวลาที่รอคอย (waiting time) การวัดเกิดจากการวิเคราะห์ความน่าจะเป็น โดยพิจารณาจากการกระจายของจังหวะเวลาโปรเซสที่เข้าสู่คิวในระบบ (process arrival-time distribution) ดังนั้นการใช้ทฤษฎีการวิเคระห์ความน่าจะเป็นนี้เรียกว่า queueing-network analysis โดย

$$\mathbf{n} = \boldsymbol{\lambda} \times \mathbf{W} \qquad (5 - 1)$$

โดยที่ $\mathbf{n}$ เป็นค่าเฉลี่ยความยาวของคิว (จำนวนโปรเซสในคิว)

$\boldsymbol{\lambda}$ เป็นค่าเฉลี่ยอัตราจำนวน processes ใหม่เข้าสู่ระบบ (#process/second)

$\mathbf{W}$ เป็นค่าเฉลี่ยของ waiting time

สมการดังกล่าวนี้สามารถใช้ประมาณการได้ทุกวิธีการสลับงาน ทั้งนี้ต้องอาศัยการวัดหรือสถิติของตัวแปรสองตัว เพื่อคำนวนหาค่าตัวแปรที่ต้องการทราบ ดังตัวอย่างที่ 5-7

ตัวอย่างที่ 5-7: การวิเคราะห์ประสิทธิภาพของวิธีการสลับงานด้วยทฤษฎีคิว

หากเราทราบจากสถิติว่าระบบปฏิบัติการมีโปรเซสใหม่เข้าสู่ระบบ 7 โปรเซสใน 1 วินาที และทราบด้วยว่า โดยปกติมีโปรเซสค้างอยู่ในคิว 14 โปรเซส อยากทราบเวลาเฉลี่ยของเวลาในการรอคอยของโปรเซสมีค่าเท่าไหร่

จากสมการ (5 – 1) $$n = \lambda \times W$$

ได้ $$n = 14,\ \lambda = 7$$

$$W = \frac{14}{7} = 2 \text{ Seconds}$$

5.4.3 การจำลองสถานการณ์ (Simulation)

การจำลองสถานการณ์ให้เสมือนสภาพแวดล้อมมีเกิดขึ้นจริงจะช่วยให้ผลการวิเคราะห์มีความแม่นยำมากยิ่งขึ้น แบบจำลองเป็นซอฟต์แวร์พร้อมโครงสร้างข้อมูลต่าง ๆ ซึ่งเลียนแบบสภาพจริงซึ่งถูกพัฒนาด้วยภาษาโปรแกรมคอมพิวเตอร์ ผู้วิเคราะห์จำเป็นต้องกำหนดสภาพให้ใกล้สภาพจริงมากที่สุด การทำงานของแบบจำลองจะอยู่บนพื้นฐานของสัญญาณนาฬิกา ซึ่งเหตุการณ์จะเกิดขึ้นตามลำดับพร้อมทั้งบันทึกเหตุการณ์เหล่านั้นเพื่อการวิเคระห์ แบบจำลองที่ให้ผลวิเคราะห์แม่นยำจำเป็นต้องจำลองสิ่งต่าง ๆ ที่เกิดขึ้นจริงให้สมจริง

5.4.4 การพัฒนาจริง (Implementation)

การพัฒนาขั้นตอนวิธีและนำไปทดสอบใช้จริงถือเป็นวิธีการที่ได้ผลแม่นยำที่สุด แต่มีความยุ่งยากให้ใช้เวลาในการพัฒนาค่อนข้างนาน การทดสอบอาจพบอุปสรรคในหลายประการ โดยเฉพาะในสภาพแวดล้อมจริงที่ไม่สามรถควบคุมได้ นอกจากนั้นยังมีค่าใช้ค่อนข้างสูงเมื่อเปรียบเทียบกับวิธีการอื่น ๆ

5.5 การสลับงานภายใต้ระบบหลายหน่วยประมวลผล (Multiple-Processor Scheduling)

เนื้อหาที่ผ่านมาเป็นการอธิบายบนพื้นฐานของระบบคอมพิวเตอร์ที่มีหน่วยประมวลผลเดี่ยว หากพิจารณาในระบบที่มีหน่วยประมวลผลมากกว่าหนึ่ง (multiple CPUs) ความซับซ้อนในการสลับงานย่อมมีมากขึ้น นอกจากนี้ยังมีสิ่งที่ต้องคำนึงอีกหลายประการ ข้อสมมุติฐานในเบื้องต้นสำหรับระบบคอมพิวเตอร์ที่มีหน่วยประมวลหลายชุดในที่นี้ คือ หน่วยประมวลผลแต่ละชุดมีความเหมือนกันทุกประการ (homogeneous multiple-processors)

การทำงานของระบบที่มีหน่วยประมวลผลหลายชุดนั้น สามารถกำหนดให้ ซีพียู ตัวใดตัวหนึ่งเป็นผู้จัดการ (master) ทำการจัดการสลับงานของระบบทั้งหมด ส่วน ซีพียู ชุดอื่น ๆ ทำหน้าที่เพียงประมวลผลให้แก่ผู้ใช้เท่านั้น ดังนั้น master CPU เท่านั้นที่เข้าถึงข้อมูลได้ เรียกรูปแบบการทำงานลักษณะนี้ว่า asymmetric multiprocessing

ในทางตรงข้าม ซีพียู แต่ละตัวต่างจัดการงานของตัวเองอย่างอิสระ โดยอาจจะมีคิวร่วมกัน (a common ready queue) หรือมีคิวเป็นของตัวเอง (private ready queue) เรียกรูปแบบนี้ว่า symmetric multiprocessing (SMP) ซึ่งเป็นการใช้งานข้อมูลร่วมกัน ดังนั้นการจัดการประมวลผลจึงต้องคำนึงถึงความสอดคล้องกันเป็นสำคัญ

ประการหนึ่งที่ต้องคำนึงถึงคือการจัดสมดุลย์ในการประมวลผลของแต่หน่วยประมวลผล (Load balancing) ซึ่งเป็นการทำให้ภาระการประมวลผลของแต่ละ ซีพียู มีความสมดุลย์กัน ซึ่งเป็นการใช้หน่วยประมวลผลหลาย ๆ ชุดของระบบ อย่างเต็มประสิทธิภาพ มีวิธีการจัดการสมดุลย์ 2 แบบคือ Push migration และ Pull migration

1) Push migration เป็นกระบวนที่ตรวจสอบภาระงานของแต่ละ ซีพียู อย่างสม่ำเสมอ หากพบความไม่สมดุลย์เกิดขึ้น ภาระงานจะถูกผลักดันไปให้แก่หน่วยประมวลผลชุดอื่นที่ยังว่างหรือมีงานน้อย

2) Pull migration เป็นกระบวนตรงข้างคือดึงงานจากหน่วยประมวลผลชุดอื่นเมื่อพบว่าหน่วยประมวลผลตัวเองว่างลง

บทสรุป

ในบทนี้ได้กล่าวถึงหลักการพื้นฐานของการสลับงาน ซึ่งเป็นความสามารถที่สำคัญของระบบปฏิบัติการยุคใหม่ ความต้องการทำงานพร้อม ๆ กันหลายโปรแกรมในเวลาเดียวกัน ระบบปฏิบัติการจึงจำเป็นต้องใช้ชุดควบคุมโปรเซสเรียกว่า Process Control Block (PCB) มาช่วยเหลือในการจดจำบริบทให้กับโปรเซส ซึ่งถูกสลับซับเปลี่ยนกันประมวลผล วิธีการสลับงานมีด้วยกันหลายวิธี ในบทนี้ได้นำเสนอ 6 วิธีคือ 1) First – Come – First – Serve (FCFS) scheduling, 2) Shortest – Job – First scheduling (SJF), 3) Priority Scheduling (PS), 4) Round – Robin scheduling (RR), 5) Multilevel Queue scheduling (MQ), และ 6) Multilevel Feedback Queue scheduling (MFQ) นอกจากนั้นวิธีการวัดประสิทธิภาพของวิธีการสลับงานได้อธิบายพร้อมยกตัวอย่าง โดยวิธีการวัดประสิทธิภาพสามารถกระทำผ่านการคำนวน การวิเคราะห์ทฤษฎีคิว การจำลองเหตุการณ์เสมือนจริง และทดสอบด้วยการพัฒนาให้เห็นจริง

แบบฝึกหัดท้ายบท

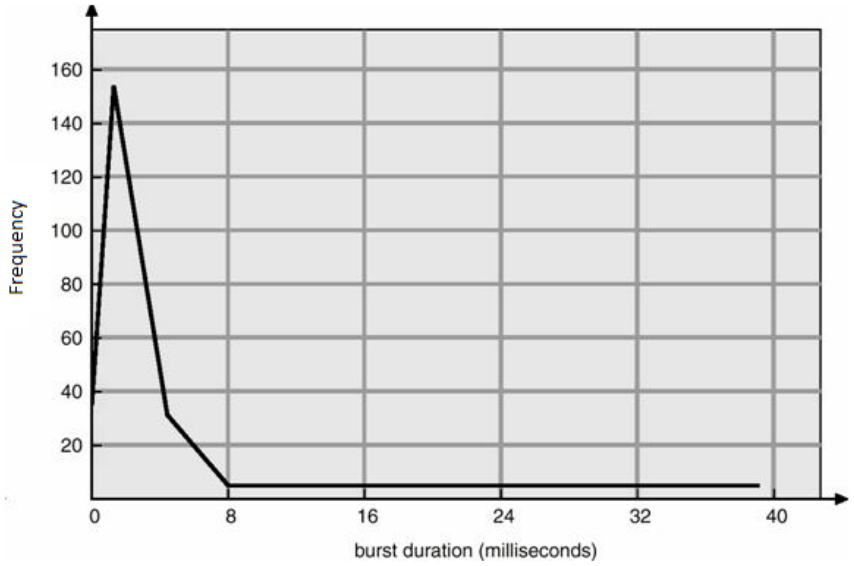
5.1. จงบอกเหตุผลความจำเป็นที่ต้องมีการสลับงานให้แก่ ซีพียู

5.2. จงอธิบายหลักการพื้นฐานของการสลับงาน

5.3. จงอธิบาย CPU Burst , I/O Burst และ CPU-I/O Burst Cycle

5.4. จงอธิบาย CPU Bound และ I/O Bound

5.5. จงอธิบายกราฟด้านล่างซึ่งแสดงช่วงเวลาของ CPU Burst



5.6. Scheduler ประเภทใดที่เป็นผู้จัดการสลับงานให้ ซีพียู

5.7. การสลับงานเกิดจากเหตุการณ์หรือสาเหตุใดได้บ้าง

5.8. จงอธิบายตัวบ่งชี้ประสิทธิภาพของการสลับงานต่อไปนี้

- CPU utilization:
- Throughput:
- Turnaround time:
- Waiting time:
- Response time:

5.9. จงเลือกอธิบายวิธีการสลับงานต่อไปนี้อย่างละเอียดมาอย่างน้อย 4 วิธี

- First-Come, First-Served Scheduling
- Shortest-Job-First Scheduling
- Priority Scheduling

- Round-Robin Scheduling
- Multilevel Queue Scheduling
- Multilevel Feedback Queue Scheduling

5.10. จากคำตอบข้อที่แล้วให้คำนวนหาค่า average waiting time (AWT) และเปรียบเทียบวิธีการที่เหมาะสมที่สุดพร้อมให้เหตุผล

Process	Arrival Time	Priority	Burst Time
P1	0	3	8
P2	1	1	4
P3	2	4	9
P4	3	5	5

บรรณานุกรม

B.L.Stuart. (2008). *Principles of Operating Systems: Design & Applications.* Cengage Learning EMEA.

Das, V. V., Vijayakumar, R., Debnath, N. C., Stephen, J., Meghanathan, N., Sankaranarayanan, S., . . . Thankachan, N. (2010). *Information Processing and Management.* Springer Science & Business Media.

Dhotre, I. A. (2009). *Operating Systems.* Technical Publication Pune.

E.Baird. (2015). *Uniprocessor Scheduling*. Retrieved September 30, 2017, from SlidePlayer: http://slideplayer.com/slide/777817/

Garrido. (2011). *Principles of Modern Operating Systems.* Jones & Bartlett Publishers.

Khurana, R. (2011). *Operating System.* Vikas Publishing House.

R.Teodor. (2015). *Computer-based Problem Solving Process.* World Scientific.

Reddy, C. M. (2009). *Operating Systems Made Easy.* Laxmi Publications.

Saltzer, J. H., & Kaashoek, M. F. (2009). *Principles of Computer System Design.* Morgan Kaufmann.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2010). *Operating System Concepts* (8th ed.). Asia: John Wiley & Sons.