แผนการบริหารการสอนประจำบทที่ 6

เนื้อหาประจำบท

บทที่ 6 กำหนดการใช้ซีพียู

1. แนวคิดในการจัดการโปรเซสของโปรแกรมที่กำลังดำเนินการและรูปแบบของการประมวลผล
2. การจัดตารางการดำเนินการของโพรเซส
3. การดำเนินการและการสื่อสารระหว่างโพรเซส

จุดประสงค์เชิงพฤติกรรม

เมื่อศึกษาบทที่ 6 แล้วนักศึกษาสามารถ

1. อธิบายการจัดการตารางการจัดการทำงานของหน่วยประมวลผลกลาง
2. อธิบายการแบ่งสรรและจัดลำดับการเข้าใช้งานของทรัพยากร
3. อธิบายการทำงานของอัลกอริทึมในการเลือกโปรเซสเข้าไปใช้หน่วยประมวลผลกลาง

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายหลักการทำงานของระบบปฏิบัติการ พร้อมยกตัวอย่างประกอบการบรรยาย
2. ให้ผู้เรียนศึกษาเอกสารประกอบการเรียนการสอน ศึกษาทำความเข้าใจและซักถาม
3. ให้ผู้เรียนทำแบบฝึกหัดและงานที่ได้รับมอบหมาย
4. ทดสอบย่อยหลังจบบทเรียน

สื่อการเรียนการสอน

1. สื่ออิเล็กทรอนิกส์ประกอบการสอนวิชาระบบปฏิบัติการ
2. เอกสารประกอบการสอนวิชาระบบปฏิบัติการ
3. หนังสืออ่านประกอบค้นคว้าเพิ่มเติม

การวัดผลและประเมินผล

1. สังเกตจากการซักถามในระหว่างการเรียน
2. สังเกตจากความสนใจและความตั้งใจ
3. ประเมินจากการอภิปรายกลุ่มย่อย และจากการทำแบบฝึกหัด
4. ประเมินจากการสอบระหว่างภาคและปลายภาค

บทที่ 6
กำหนดการใช้ซีพียู

ในการทำงานของโพรเซส จำเป็นต้องมีการเรียกใช้ทรัพยากรของระบบซึ่งมีอยู่อย่างจำกัด ระบบจำเป็นต้องมีการแบ่งสรรและจัดลำดับการเข้าใช้งานของทรัพยากร ซีพียูเป็นทรัพยากรที่จำเป็นต้องมีการจัดลำดับให้หลายโพรเซสเข้าไปใช้งาน ดังนั้นเมื่อซีพียูว่างและมีโพรเซสเข้าคิวรออยู่ที่คิวพร้อมต้องการเข้าไปใช้งานที่ซีพียู ในการกำหนดการใช้ซีพียู (CPU scheduler) จะทำหน้าที่เลือกโพรเซสที่อยู่ในคิวพร้อมออกมาทีละโพรเซส และมอบซีพียูให้โพรเซสที่ได้รับเลือก ทำให้โพรเซสนั้นเข้าไปทำงานที่ซีพียูได้ และเปลี่ยนสถานะจากสถานะพร้อมเป็นสถานะทำงาน

การจัดเวลาซีพียู (CPU scheduling) เป็นหลักการทำงานหนึ่งของระบบปฏิบัติการ ที่ทำให้คอมพิวเตอร์มีความสามารถในการรันโปรแกรมหลาย ๆ โปรแกรมในเวลาเดียวกัน ซึ่งการแบ่งเวลาการเข้าใช้ซีพียูให้กับโพรเซสที่อาจถูกส่งมาหลาย ๆ โพรเซสพร้อม ๆ กัน ในขณะที่ซีพียูอาจมีจำนวนน้อยกว่าโพรเซส หรืออาจมีซีพียูเพียงตัวเดียว จะทำให้คอมพิวเตอร์สามารถทำงานได้ปริมาณงานที่มากขึ้นกว่าการให้ซีพียูทำงานให้เสร็จทีละโพรเซส ในบทนี้เราจะมาพูดถึงอัลกอริทึมพื้นฐานของการจัดเวลาซีพียูนี้ โดยจะพูดถึงวิธีการหลัก ๆ ที่แต่ละอัลกอริทึมมีแตกต่างกัน ข้อดีข้อเสียและความเหมาะสมต่อระบบงานแบบต่าง ๆ เพื่อการเลือกใช้อย่างถูกต้อง

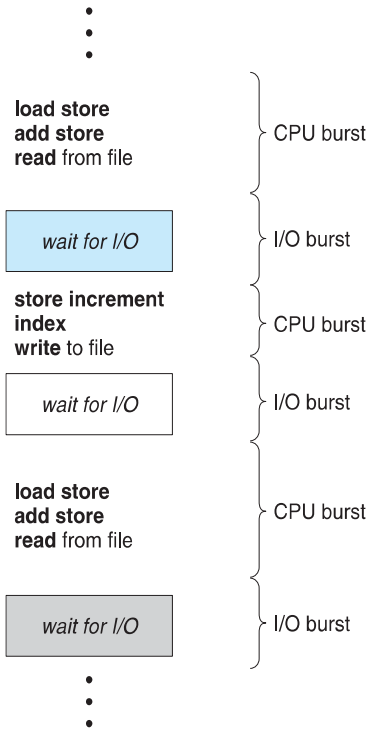
6.1 หลักความต้องการพื้นฐาน

จุดประสงค์ของการรันโปรแกรมหลายโปรแกรมคือ ความต้องการที่จะให้ซีพียูมีการทำงานตลอดเวลา เพื่อให้มีการใช้ซีพียูอย่างเต็มที่และเต็มประสิทธิภาพ ซึ่งระบบคอมพิวเตอร์มีซีพียูตัวเดียว ในเวลาใดเวลาหนึ่งซีพียูจะทำงานได้เพียงงานเดียวเท่านั้น ถ้ามีหลายโปรแกรมหรือหลายงาน งานที่เหลือก็ต้องคอยจนกว่าจะมีการจัดการให้เข้าไปใช้ซีพียู ความคิดและหลักการของการทำงานกับหลายโปรแกรมในขั้นพื้นฐานนั้นค่อนข้างที่จะไม่ซับซ้อน แต่ละโปรแกรมจะถูกรันจนกระทั่งถึงจุดที่มันจะต้องคอยอะไรซักอย่างเพื่อใช้สำหรับการทำงานช่วงต่อไป ส่วนมากการคอยเหล่านี้ก็คือการทำงานที่เกี่ยวข้องกับอินพุต/เอาต์พุตนั่นเอง ในระบบคอมพิวเตอร์ที่มีความสามารถรันโปรแกรมได้ทีละโปรแกรม การทำงานของระบบก็จะไม่ซับซ้อน ซีพียูจะหยุดการทำงานในระหว่างที่คอยอินพุต/เอาต์พุตนี้ ซึ่งการคอยเหล่านี้เป็นการเสียเวลาโดยเปล่าประโยชน์ เพราะซีพียูไม่ได้ทำงานเลย ส่วนหลักในการรันหลายโปรแกรม เราพยายามใช้เวลาที่ซีพียูต้องคอยนี้ทำงานอย่างอื่นต่อไป

ดังนั้นเมื่อใดก็ตามที่ซีพียูต้องคอย และยังมีโปรแกรมในหน่วยความจำหลายโปรแกรมที่คอยการใช้ซีพียูอยู่ เราจะจัดให้ซีพียูทำงานในโปรแกรมเหล่านั้นทันที ซึ่งระบบจะจัดการนำเอาโปรแกรมที่คอยอินพุต/เอาต์พุตออกไปก่อน เพื่อที่จะให้โปรแกรมอื่นที่คอยใช้ซีพียูนี้สามารถเข้ามาได้ ถ้าทำงานในแบบนี้ไปเรื่อย ๆ ซีพียูก็จะได้มีงานเกือบตลอดเวลากับโปรแกรมหลาย ๆ โปรแกรมที่อยู่ในระบบ การจัดเวลาให้กับซีพียูแบบนี้ เป็นหลักความต้องการพื้นฐานของระบบปฏิบัติการในคอมพิวเตอร์ ทรัพยากรที่คอมพิวเตอร์มีอยู่ในเครื่อง ๆ หนึ่ง จะถูกจัดสรรการใช้ก่อนการนำไปให้โปรแกรมใด ๆ ซีพียูเองก็ถือได้ว่าเป็นทรัพยากรของระบบคอมพิวเตอร์ชนิดหนึ่งที่มีความสำคัญมากที่สุด โดยซีพียูนี่เองที่จะเป็นศูนย์กลางของการสร้างระบบปฏิบัติการที่มีความสามารถในการรันหลายโปรแกรม

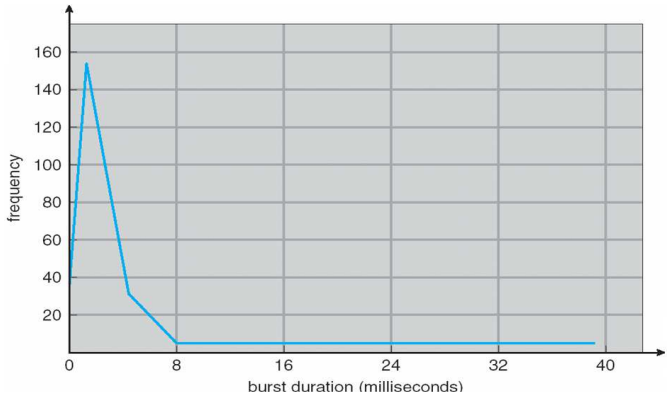
6.1.1 ช่วงเวลาอินพุต/เอาต์พุต และช่วงเวลาใช้ซีพียู (I/O and CPU Burst Cycle)

ความสำคัญของการจัดเวลาของซีพียูนั้น ขึ้นอยู่กับคุณลักษณะการทำงานของโพรเซส โดยทั่วไปการทำงานของโพรเซสจะประกอบด้วยเวลาที่ใช้ซีพียู (CPU burst cycle) และเวลาที่คอยอุปกรณ์อินพุต/เอาต์พุต (Input/Output and CPU burst cycle) ในขณะที่มีการทำงานโพรเซส จะมี การสลับการทำงานระหว่าง 2 ช่วงเวลานี้เท่านั้น และจะเกิดไม่พร้อมกัน และการทำงานมักจะเริ่มจาก การใช้ซีพียู แล้วก็ตามด้วยรออินพุต/เอาต์พุต เมื่อจบการรอคอยก็จะตามมาด้วยเวลาของซีพียู สลับกันไปเรื่อย ๆ จนกว่าจะจบการทำงาน ซึ่งการทำงานนี้มักเป็นการใช้เวลาซีพียูเพื่อทำการจบหรือสิ้นสุดโพรเซสมากกว่าการรอคอยอินพุต/เอาต์พุต (ดูภาพที่ 6.1)



ภาพที่ 6.1 การทำงานที่หลากหลายที่ใช้เวลาซีพียู และเวลาในการคอยอินพุต/เอาต์พุต
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.262)

จากการค้นคว้าที่ผ่านมาได้มีการศึกษาถึงลักษณะช่วงเวลาของการใช้ซีพียูไว้ สามารถมองเห็นคร่าว ๆ ว่า ลักษณะของช่วงเวลาที่ใช้ซีพียูในแต่ละโพรเซสจะมีรูปแบบอย่างไร ถึงแม้ว่ารูปแบบของเวลาอาจจะมีความแตกต่างกันอยู่บ้าง แต่ก็มีแนวโน้มที่แต่ละโพรเซสมักจะมีคาบเวลาการเข้าใช้ซีพียูคล้าย ๆ กันในภาพที่ 6.2



ภาพที่ 6.2 ฮีสโตแกรมของเวลาซีพียู
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.263)

จากกราฟอธิบายได้ว่า การใช้ซีพียูของโพรเซสใด ๆ นั้น มักจะมีความถี่มากสำหรับเวลาซีพียูช่วงเวลาสั้น ๆ ส่วนเวลาซีพียูระยะเวลานาน ๆ นั้นจะมีความถี่น้อยกว่า นอกจากนี้ยังมีแนวโน้มว่า โพรเซสที่มีการติดต่อแลกเปลี่ยนข้อมูลกับอุปกรณ์ภายนอกมาก ๆ ก็มักจะมีช่วงเวลาของการใช้ซีพียูค่อนข้างสั้นแต่บ่อยครั้ง และมีช่วงเวลาการใช้ซีพียูนาน ๆ เพียงเล็กน้อย และการศึกษาคุณสมบัติของโพรเซสต่าง ๆ เหล่านี้อย่างละเอียด ทำให้เราสามารถนำมาใช้เป็นข้อมูลในการเลือกลักษณะของอัลกอริทึมที่เหมาะสมสำหรับการจัดเวลาให้กับซีพียูให้ดีที่สุดในระบบคอมพิวเตอร์แต่ละระบบได้ต่อไป

6.2 ตัวจัดการเวลาซีพียู

เมื่อใดก็ตามที่ซีพียูว่าง ระบบปฏิบัติการจะต้องเข้ามาเลือกโพรเซสตัวใดตัวหนึ่งที่คอยอยู่ในคิวเข้ามาใช้งานซีพียู การเลือกโพรเซสเพื่อเข้ามาใช้ซีพียูนี้ จะถูกจัดการด้วยส่วนที่เรียกว่า ตัวจัดการช่วงสั้น (Short – term scheduler) หรือตัวจัดการเวลาซีพียู (CPU scheduler) ตัวจัดการเวลานี้จะเลือกโพรเซสที่อยู่ในหน่วยความจำที่พร้อมในการทำงานที่สุด เพื่อให้ครอบครองเวลาซีพียูและทรัพยากรที่เกี่ยวข้องกับโพรเซสนั้น ตัวของโพรเซสในหน่วยความจำนั้นไม่จำเป็นต้องเป็นแบบมาก่อนได้ก่อน (FIFO : First in First out) เสมอไป อาจจะเป็นไปตามลำดับความสำคัญ โครงร่างต้นไม้ (Tree) หรืออาจจะเป็น ลิ้งลิสต์ก็ได้ อย่างไรก็ตามโพรเซสทุกโพรเซสที่พร้อมใช้ซีพียู จะต้องมีโอกาสได้เข้าครอบครองเวลาซีพียูไม่เวลาใดก็เวลาหนึ่ง ซึ่งการเข้าและออกจากการครอบครองเวลาซีพียูแต่ละครั้ง จำเป็นต้องมีการเก็บข้อมูลไว้เสมอว่า เข้ามาแล้วได้ทำอะไรไปบ้าง ช่วงต่อไปจะทำอะไร เป็นต้น โดยใช้พื้นที่หน่วยความจำส่วนหนึ่งเก็บข้อมูลของแต่ละโพรเซสหลังเสร็จสิ้นการใช้ซีพียู พื้นที่หน่วยความจำที่มีชื่อว่าบล็อกควบคุมโพรเซส (Process Control Block : PCB)

6.2.1 การให้สิทธิการจัดเวลา (Preemptive Scheduling)

การตัดสินใจของซีพียูในการเลือกให้โพรเซสใด ๆ ทำงาน ขึ้นอยู่กับสถานการณ์ดังนี้

1) เมื่อมีการเปลี่ยนสถานะของโพรเซสจากสถานะรัน ไปเป็นสถานะคอย เช่น ในสภาวะที่คอยอินพุต/เอาต์พุต หรือการคอยให้โพรเซสลูกเสร็จสิ้นไปก่อน เป็นต้น

2) เมื่อมีการเปลี่ยนสถานะของโพรเซสจากรัน เป็นสถานะพร้อม เช่น เมื่อมีอินเทอร์รัพเกิดขึ้น

3) เมื่อมีการเปลี่ยนสถานะของโพรเซสจากสถานะคอย เป็นสถานะพร้อม เช่น เมื่อ อินพุต/เอาต์พุต เสร็จสิ้นไปแล้ว เป็นต้น

4) เมื่อโพรเซสเสร็จสิ้นหรือสิ้นสุดการดำเนินการ

ทั้ง 4 สถานการณ์ดังกล่าวข้างต้น ในสถานการณ์ที่ 1 และ 4 นั้นเป็นสถานการณ์ที่จะต้องมีการตัดสินใจทำอะไรอย่างใดอย่างหนึ่ง โดยไม่สามารถหลีกเลี่ยงได้ เช่น ต้องไปเลือกโพรเซสใหม่เข้ามาทำงานต่อไป เนื่องจากโพรเซสเดิมไม่ใช้ซีพียูอีกแล้ว แต่สำหรับสถานการณ์ที่ 2 และ 3 นั้น การตัดสินใจต้องอยู่บนพื้นฐานหรือกฎเกณฑ์ของแต่ละอัลกอริทึมที่ใช้ ซึ่งอาจทำให้มีการทำงานที่แตกต่างกันไป การตัดสินใจที่เกิดขึ้นเนื่องจากสถานการณ์ 1 และ 4 การจัดเวลาซีพียูจะเป็นแบบไม่ให้สิทธิ์ก่อน (Non preemptive) นอกนั้นจะเรียกว่าให้สิทธิ์ก่อน (Preemptive) ภายใต้การทำงานแบบไม่ให้สิทธิ์ก่อนนี้ โพรเซสจะครอบครองเวลาซีพียูไปจนกว่าจะเสร็จ หรือเปลี่ยนสถานะตัวเองเป็นสถานะคอย ซึ่งเป็นเพียงวิธีการที่ใช้ได้เฉพาะกับระบบของฮาร์ดแวร์เฉพาะแบบเท่านั้น ถ้าระบบนี้ต้องการทำเป็นระบบที่ให้ผู้ใช้หลายคนเข้ามาใช้งานพร้อมกัน (Multi-user) การจัดเวลาแบบให้สิทธิ์ก่อนจะเหมาะสมกว่า

6.2.2 ตัวส่งต่อ (Dispatcher)

องค์ประกอบที่สำคัญอีกตัวหนึ่งที่เกี่ยวข้องกับฟังก์ชันในการจัดเวลาซีพียูก็คือ สิ่งที่เรียกว่า Dispatcher ซึ่งเป็นโมดูลที่ทำหน้าที่ควบคุมการครอบครองซีพียูของโพรเซส โมดูลนี้ประกอบด้วยฟังก์ชัน

1) การย้าย Context

2) การย้ายไป User mode

3) กระโดดไปยังตำแหน่งที่เหมาะสมของโปรแกรม เพื่อที่จะเริ่มรันโปรแกรมนั้นใหม่อีกครั้ง

ลักษณะของ Dispatcher คือการทำ Context switching ดังนั้นควรมีการทำงานที่เร็วที่สุดเท่าที่จะทำได้ เพราะว่ามันจะต้องทำงานทุกครั้งที่มีการย้ายโพรเซส ซึ่งเวลาที่ถูกใช้ไปกับการย้ายโพรเซสที่กำลังใช้ซีพียูให้ออกจากซีพียู และนำโพรเซสอื่นเข้าใช้ซีพียูแทนเช่นนี้เรียกว่า Dispatch latency

6.3 เกณฑ์การวิเคราะห์ประสิทธิภาพ

อัลกอริทึมในการเลือกโพรเซสเข้าไปใช้ซีพียูมีหลายอัลกอริทึม และแต่ละอัลกอริทึมของการจัดเวลาซีพียูมีคุณสมบัติแตกต่างกันไป ดังนั้นการเลือกอัลกอริทึมสำหรับใช้ในสถานการณ์ต่าง ๆ นั้น จำเป็นที่จะต้องพิจารณาถึงคุณสมบัติและเป้าหมายการทำงานเหล่านี้ให้ดีว่ามีข้อดีข้อเสียอย่างไร ซึ่งลักษณะข้อพิจารณาแต่ละชนิดนั้น สามารถสร้างความแตกต่างให้เห็นได้อย่างชัดเจนในการตัดสินว่าอัลกอริทึมใดดีที่สุด ข้อพิจารณาดังกล่าวมีดังนี้

1) มีการใช้งานหน่วยประมวลผลกลาง (CPU utilization) โดยจำเป็นให้หน่วยประมวลผลกลางถูกใช้งานให้มากที่สุด โดยปกติควรจะมีการใช้งานร้อยละ 40 ถึงร้อยละ 90 ทั้งนี้ขึ้นอยู่กับปริมาณงานในระบบ

2) มีปริมาณงานมากที่สุด (Throughput) คือปริมาณงานต่อหน่วยเวลา เกณฑ์นี้เกี่ยวพันกับการใช้งานหน่วยประมวลผลกลาง ทั้งนี้ปริมาณงานที่ผ่านเข้ามาในระบบมาก หน่วยประมวลผลกลางจะถูกใช้งานมากตามปริมาณงาน

3) มีเวลาครบวงงานน้อยที่สุด (Turnaround time) เป็นเวลาต่อรอบงาน คือเวลาที่ใช้ในระบบทั้งหมดในการทำงานของโพรเซสใด ๆ หรือกล่าวได้ว่าเป็นเวลาที่ผู้ใช้ต้องรอ โดยเริ่มตั้งแต่นำโพรเซสเข้าสู่ระบบจนได้รับผลลัพธ์หรือเอาท์พุตที่ต้องการกลับมา สิ่งที่ควรให้ความสำคัญคือแม้ว่าการใช้งานหน่วยประมวลผลกลางจะสูง แต่ถ้าผู้ใช้ระบบต้องรอนาน อาจทำให้เกิดความล่าช้าต่อความต้องการของผู้ใช้งาน

4) มีเวลารอน้อยที่สุด (Waiting time) คือเวลาของโพรเซสที่ถูกรอใน Ready queue

5) มีเวลาตอบสนองน้อยที่สุด (Response time) หมายถึงเวลาที่มีการร้องขอข้อมูลหรือการส่งงานเข้าไปในระบบและได้การตอบสนองกลับมาในครั้งแรก (ไม่ใช่ผลลัพธ์) เช่น ระบบที่มี การโต้ตอบข้อมูล (interactive system) ซึ่งต้องมีการตอบข้อมูลกลับมาหลังป้อนคำสั่งเข้าไป

6.4 อัลกอริทึมของการจัดเวลา

อัลกอริทึมสำหรับการจัดการเวลาในโพรเซสนั้น มีความสำคัญอยู่ที่การตัดสินว่าจะให้โพรเซสใดครอบครองเวลาซีพียูก่อน ซึ่งต่อไปนี้เราจะมาศึกษากันว่ามีวิธีการใดบ้างที่ใช้ในการตัดสินใจคัดเลือกโพรเซส

6.4.1 First-Come, First-Served (FCFS) Scheduling

เป็นอัลกอริทึมที่ง่ายที่สุด โดยจะกำหนดให้โพรเซสที่ร้องขอซีพียูก่อน เป็นโพรเซสที่ได้รับซีพียูก่อนเมื่อมีโพรเซสที่อยู่ในสถานะพร้อมที่จะทำงาน โพรเซสนั้นจะถูกนำเข้าไปต่อท้ายคิวพร้อม เมื่อซีพียูว่าง ระบบปฏิบัติการจะเรียกกำหนดการซีพียู เพื่อให้พิจารณามอบซีพียูให้แก่โพรเซสที่อยู่ต้นคิวของคิวพร้อม สูตรในการคำนวณหาเวลาครบวงงาน สามารถคำนวณได้ดังนี้

$T = \sum_{i=1}^{n} Ti \times 1/n$ เมื่อ $Ti = Fi - Ai$

Ti หมายถึง เวลาครบวงงานของแต่ละโพรเซส (Turnaround time)
Fi หมายถึง เวลาที่แต่ละโพรเซสทำงานเสร็จสิ้น (Finish time)
Ai หมายถึง เวลาที่แต่ละโพรเซสเข้ามาในระบบ (Arrival time)
n หมายถึง จำนวนของโพรเซสที่เข้ามาในระบบ
T หมายถึง เวลาครบวงงานเฉลี่ย (Average Turnaround time)

<u>ตัวอย่างที่ 1</u> ระบบคอมพิวเตอร์มี 3 โพรเซสที่ต้องการเข้าใช้งานซีพียู คือ P1, P2 และ P3 เมื่อ

1) โพรเซส P1 เข้าระบบเมื่อเวลา 8.00 และต้องการใช้ซีพียู 2 หน่วยเวลา
2) โพรเซส P2 เข้าระบบเมื่อเวลา 8.10 และต้องการใช้ซีพียู 1 หน่วยเวลา
3) โพรเซส P3 เข้าระบบเมื่อเวลา 8.25 และต้องการใช้ซีพียู 0.25 หน่วยเวลา

ให้แสดงวิธีทำเพื่อหาเวลาครบวงงานเฉลี่ย กรณีใช้อัลกอริทึมจัดลำดับใช้ซีพียูแบบมาก่อนบริการก่อน

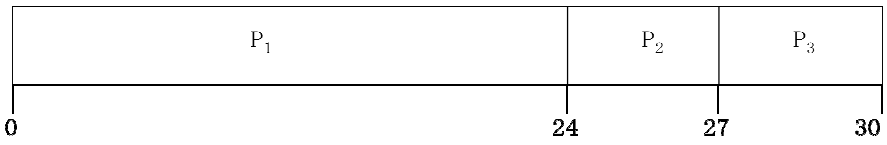
Process	Arrival Time	Run Time	Start Time	Finish Time	Turnaround Time
P1	8.00	2.00	8.00	10.00	2.00
P2	8.10	1.00	10.00	11.00	2.90
P3	8.25	0.25	11.00	11.25	3.00
				รวม	7.90
					เวลาครบวงงานเฉลี่ย = 7.90/3 = 2.63 หน่วยเวลา

จากการทำงานด้วยอัลกอริทึมนี้ สามารถคำนวณค่าเฉลี่ยของเวลาครบวงงานได้ เท่ากับ 2.63 หน่วยเวลา

ตัวอย่างที่ 2 ให้พิจารณาระบบที่ประกอบไปด้วย 3 โพรเซสที่ถูกรับเข้ามาในระบบ เรียงตามลำดับ คือ P1, P2 และ P3 โดยที่แต่ละโพรเซสต้องการใช้ซีพียูเป็นเวลาตามที่กำหนด ให้หาค่าเฉลี่ยของเวลารอ เมื่อกำหนดให้ใช้อัลกอริทึมแบบมาก่อนบริการก่อน

Process	Burst Time
P_1	24
P_2	3
P_3	3

การรอของแต่ละโพรเซส สามารถแสดงได้ด้วย Gantt Chart ดังนี้:



1) P1 เข้ามาในระบบและได้รับการจัดสรรให้ใช้ซีพียูทันที จึงไม่ต้องเสียเวลาในการรอซีพียู ดังนั้นเวลาในการรอซีพียู = 0 หน่วยเวลา

2) P2 ต้องรอจนกว่า P1 ทำงานเสร็จเรียบร้อยและคืนซีพียูให้กับระบบ ดังนั้นเวลาในการรอซีพียู = 24 หน่วยเวลา

3) P3 ต้องรอจนกว่า P2 ทำงานเสร็จเรียบร้อยและคืนซีพียูให้กับระบบ ดังนั้นเวลาในการรอซีพียู = 27 หน่วยเวลา

ดังนั้นค่าเฉลี่ยของเวลาที่ใช้ในการรอซีพียู คือ (0+24+27)/3 = 17 หน่วยเวลา

การทำงานของอัลกอริทึมนี้ดูเหมือนเป็นการยุติธรรมที่ให้สิทธิการเข้าใช้ซีพียูแก่โพรเซสที่เข้ามาอยู่ในคิวพร้อมก่อน แต่ในกรณีที่ในคิวพร้อมของระบบมีทั้งโพรเซสที่เน้นซีพียู และโพรเซสที่เน้น I/O จะพบว่า โพรเซสที่เน้น I/O จะต้องเสียเวลารอนานมาก เพื่อเข้าใช้งานซีพียูในระยะเวลาที่ไม่นานมากนัก

ซึ่งจะทำให้เกิดปัญหา Convoy effect คือ เหตุการณ์ที่โพรเซสขนาดเล็กในระบบ จะต้องเสียเวลารอโพรเซสขนาดใหญ่ที่ครอบครองซีพียูเป็นเวลานาน การทำงานของอัลกอริทึมนี้ เป็นการทำงานที่ไม่สามารถขัดจังหวะ หรือแทรกกลางได้ (Non-preemptive process) ซึ่งจะไม่เหมาะกับระบบที่ต้องมีการแบ่งส่วนการทำงานให้งานแต่ละงานได้ใช้ซีพียูอย่างทั่วถึง

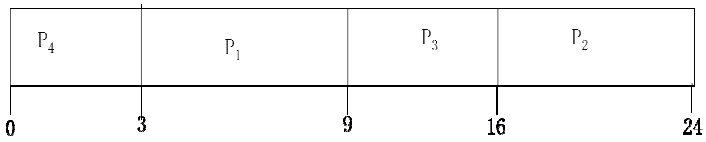
6.4.2 Shortest-Job-First (SJF) Scheduling

จากอัลกอริทึมมาก่อนบริการก่อนนั้น พบว่าค่าเฉลี่ยของเวลาครบวงงาน และค่าเฉลี่ยของเวลารอมีค่าสูง โดยเฉพาะกรณีที่ในคิวพร้อมมีโพรเซสที่ต้องการใช้ซีพียูเป็นเวลาที่แตกต่างกัน อัลกอริทึมของงานสั้นทำก่อน จะพยายามลดค่าเฉลี่ยของเวลาครบวงงาน และค่าเฉลี่ยของเวลารอ โดยกำหนดให้โพรเซสที่ต้องการใช้ซีพียูเป็นระยะเวลาน้อยได้เข้าใช้ซีพียูก่อนโพรเซสที่ต้องการใช้ซีพียูเป็นระยะเวลานาน

ตัวอย่างที่ 3 พิจารณาระบบที่ประกอบด้วยโพรเซส P1, P2, P3 และ P4 โดยที่ทุกโพรเซสถูกรับเข้ามาในระบบพร้อมกัน

Process	เวลาที่ต้องการใช้ซีพียู (burst time)
P1	6
P2	8
P3	7
P4	3

การรอของแต่ละโพรเซสจากอัลกอริทึม Shortest-Job-First (SJF) Scheduling สามารถแสดงได้ด้วย Gantt Chart ดังนี้



จาก Gantt Chart จะเห็นว่า

1) โพรเซส P1 ต้องรอเป็นเวลา 3 หน่วยเวลา
2) โพรเซส P2 ต้องรอเป็นเวลา 16 หน่วยเวลา
3) โพรเซส P3 ต้องรอเป็นเวลา 9 หน่วยเวลา
4) โพรเซส P4 ต้องรอเป็นเวลา 0 หน่วยเวลา
5) ค่าเฉลี่ยของเวลาที่ใช้ในการรอ คือ (3+16+9+0)/4 = 7 หน่วยเวลา

ตัวอย่างที่ 4 จากโจทย์ในตัวอย่างที่ 1 ให้แสดงวิธีทำเพื่อหาเวลาครบวงงานเฉลี่ย เมื่อกำหนดให้ใช้อัลกอริทึมจัดลำดับใช้ซีพียูแบบงานสั้นทำก่อน

Process	Arrival Time	Run Time	Start Time	Finish Time	Turnaround Time
P1	8.00	2.00	8.00	10.00	2.00
P2	8.10	1.00	10.25	11.25	3.15
P3	8.25	0.25	10.00	10.25	2.00
				รวม	7.15
					เวลาครบวงงานเฉลี่ย = 7.15/3 = 2.38 หน่วยเวลา

อัลกอริทึมนี้สามารถทำงานได้ทั้งแบบ Preemptive process และ Non-Preemptive process กรณีที่เป็นการทำงานแบบ Preemptive นั้น ขณะที่โพรเซสหนึ่งกำลังทำงาน และมีโพรเซสใหม่เข้ามาในคิวพร้อม และโพรเซสใหม่ต้องการใช้ซีพียูเป็นเวลาน้อยกว่าโพรเซสที่กำลังทำงาน โพรเซสเดิมที่กำลังทำงานจะถูกขัดจังหวะให้หยุดการทำงาน และคืนซีพียูให้แก่ระบบ เพื่อที่ระบบจะได้มอบซีพียูให้โพรเซสใหม่ที่ต้องการใช้ซีพียูเป็นเวลาน้อยกว่าได้เข้าไปทำงานที่ซีพียูก่อน

กรณีที่เป็นการทำงานแบบ Non-Preemptive ระบบจะยังคงให้โพรเซสเดิมทำงานต่อไปจนกว่าจะเสร็จสิ้นการทำงานของโพรเซสนั้น

ตัวอย่างที่ 5 ระบบคอมพิวเตอร์มี 3 โพรเซสที่ต้องการเข้าไปใช้งานซีพียู คือ

1) โพรเซส P1 เข้าระบบเมื่อเวลา 0.0 และต้องการใช้ซีพียู 8 หน่วยเวลา

2) โพรเซส P2 เข้าระบบเมื่อเวลา 0.4 และต้องการใช้ซีพียู 4 หน่วยเวลา

3) โพรเซส P3 เข้าระบบเมื่อเวลา 1.0 และต้องการใช้ซีพียู 1 หน่วยเวลา

ให้แสดงวิธีทำเพื่อหาเวลาครบรอบวงงานเฉลี่ย กรณีที่ใช้อัลกอริทึมจัดลำดับใช้ซีพียูแบบงานสั้นทำก่อน ในแบบ Preemptive และแบบ Non-Preemptive

1. จัดลำดับใช้ซีพียูแบบงานสั้นได้ก่อน ในแบบ Non-Preemptive

Process	Arrival Time	Run Time	Start Time	Finish Time	Turnaround Time
P1	0.0	8	0.0	8.0	8.0
P2	0.4	4	9.0	13.0	12.6
P3	1.0	1	8.0	9.0	8.0
				รวม	28.6
					เวลาครบวงงานเฉลี่ย = 28.6/3 = 9.53 หน่วยเวลา

2. จัดลำดับใช้ซีพียูแบบงานสั้นได้ก่อน ในแบบ Preemptive

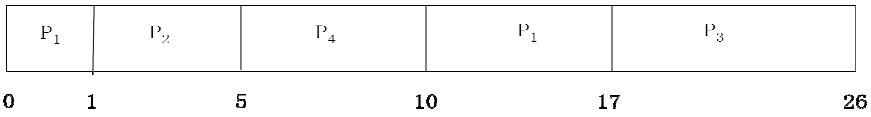
Process	Arrival Time	Run Time	Start Time	Finish Time	Left Time	Turnaround Time
P1	0.0	8	0.0	0.4	7.6	0.4
P2	0.4	4	0.4	1.0	3.4	0.6
P3	1.0	1	1.0	2.0	0	1.0
P2	1.0	3.4	2.0	5.4	0	4.4
P1	0.4	7.6	5.4	13.0	0	12.6
					รวม	19.0
เวลาครบวงงานเฉลี่ย = 19/3 = 6.33 หน่วยเวลา						

6.4.3 Shortest-remaining-time-first

เราเพิ่มแนวคิดของเวลาที่มาถึงของโพรเซสที่แตกต่างกัน และวิเคราะห์การจัดลำดับใช้ซีพียูแบบงานสั้นได้ก่อน ในแบบ Preemptive

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

การจัดลำดับใช้ซีพียูแบบงานสั้นได้ก่อน ในแบบ Preemptive สามารถแสดงได้ด้วย Gantt Chart ดังนี้:



ค่าเฉลี่ยของเวลาที่ใช้ในการรอ คือ = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

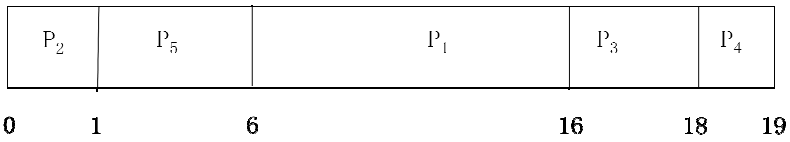
6.4.4 ลำดับความสำคัญ (Priority Scheduling)

เป็นวิธีจัดลำดับการใช้ซีพียูโดยกำหนดลำดับความสำคัญให้แต่ละโพรเซส โดยระบบจะต้องกำหนดว่า ให้ตัวเลขที่มีค่าน้อยที่สุดแสดงถึงลำดับความสำคัญน้อยที่สุด ให้ตัวเลขที่มีค่ามากที่สุดแสดงถึงลำดับความสำคัญมากที่สุด หรือให้ตัวเลขที่มีค่าน้อยที่สุดแสดงถึงลำดับความสำคัญมากที่สุด ให้ตัวเลขที่มีค่ามากที่สุดแสดงถึงลำดับความสำคัญน้อยที่สุด

ตัวอย่างที่ 6 กำหนดให้โพรเซส P1 P2 P3 P4 P5 และ P6 มีระยะเวลาทำงานและค่าลำดับความสำคัญดังต่อไปนี้

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

สามารถแสดงได้ด้วย Gantt Chart ดังนี้:



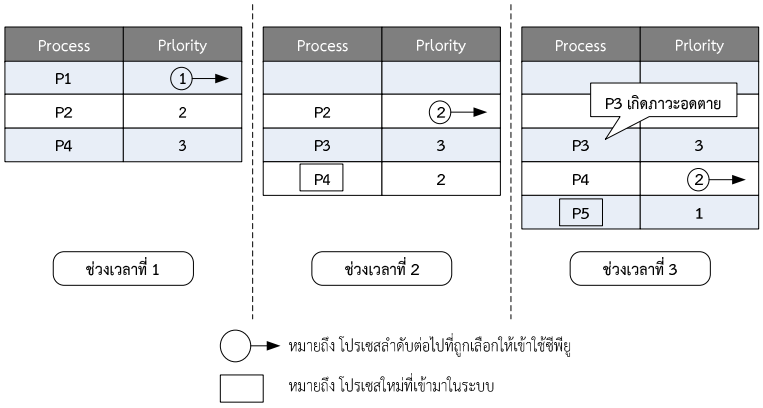
ค่าเฉลี่ยของเวลาที่ใช้ในการรอ คือ = 6+0+16+18+1/5 = 8.2 msec

การทำงานของอัลกอริทึมนี้สามารถทำงานได้ทั้งในกรณีแบบ Preemptive และแบบ Non-Preemptive โดยในกรณีที่อัลกอริทึมทำงานในแบบ Preemptive จะทำให้เกิดปัญหาสำคัญ คือ การอดตาย (Starvation) หมายความว่า โพรเซสที่มีลำดับความสำคัญต่ำกว่าถูกโพรเซสที่มีลำดับความสำคัญสูงกว่าแย่งชิงซีพียูไปใช้งาน ทำให้โพรเซสที่มีลำดับความสำคัญต่ำกว่าไม่มีโอกาสเข้าไปใช้ซีพียู วิธีการแก้ปัญหาการอดตาย สามารถทำได้โดยการทำ Aging คือ การกำหนดให้มีการเพิ่มค่าของลำดับความสำคัญของทุกโพรเซสในระบบเป็นระยะ

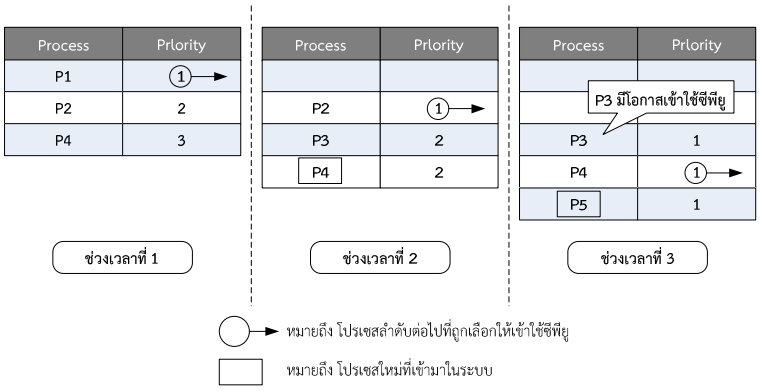
ตัวอย่างที่ 7 กรณีของการจัดลำดับใช้งานของซีพียูแบบจัดลำดับความสำคัญในแบบ Preemptive ที่มีการใช้ Aging และไม่มีการใช้ Aging

Process	ลำดับความสำคัญ
P1	1
P2	2
P3	3

เมื่อกำหนดให้ ตัวเลขที่มีค่าน้อยที่สุดมีลำดับความสำคัญสูงที่สุด



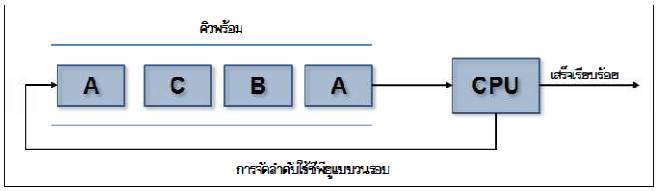
ภาพที่ 6.3 การจัดลำดับใช้งานซีพียูแบบจัดลำดับความสำคัญในแบบ Preemptive ที่ไม่มีการใช้ Aging



ภาพที่ 6.4 การจัดลำดับใช้งานซีพียูแบบจัดลำดับความสำคัญในแบบ Preemptive ที่มีการใช้ Aging

6.4.5 วิธีวนรอบ (Round-Robin Scheduling : RR)

อัลกอริทึมนี้ถูกออกแบบมาเพื่อใช้สำหรับระบบแบ่งเวลา โดยมีการทำงานเหมือนอัลกอริทึมแบบมาก่อนบริการก่อน แต่กำหนดให้โพรเซสใช้ซีพียูในเวลาที่จำกัด เรียกว่า เวลาควอนตัม (Quantum time) หรือ การแบ่งเวลา (time slice)



ภาพที่ 6.5 การแสดงการจัดลำดับซีพียูแบบวนรอบ

ในการทำงาน ตัวจัดลำดับการใช้ซีพียูจะเลือกโพรเซสจากต้นคิวพร้อมเข้าไปทำงานเป็นเวลา 1 เวลาควอนตัม ภายในระยะเวลาที่กำหนดถ้าโพรเซสสามารถทำงานเสร็จ โพรเซสจะคืนซีพียูให้ระบบ แต่ถ้าโพรเซสไม่สามารถทำงานเสร็จภายในเวลา 1 เวลาควอนตัม โพรเซสจะถูกขัดจังหวะและถูกนำไปต่อท้ายคิวพร้อม เพื่อเปลี่ยนให้โพรเซสอื่นเข้าไปทำงานในซีพียูต่อไป

ตัวอย่างที่ 8 ระบบคอมพิวเตอร์มีโพรเซสทั้งหมด 3 โพรเซส แต่ละโพรเซสมีเวลาเข้าระบบ และเวลาที่ต้องการใช้ซีพียู ดังนี้

1) โพรเซส P1 เข้าระบบเมื่อเวลา 0.0 และต้องการใช้ซีพียู 8 หน่วยเวลา
2) โพรเซส P2 เข้าระบบเมื่อเวลา 0.4 และต้องการใช้ซีพียู 4 หน่วยเวลา
3) โพรเซส P3 เข้าระบบเมื่อเวลา 1.0 และต้องการใช้ซีพียู 1 หน่วยเวลา

เมื่อกำหนดให้ใช้การจัดลำดับใช้งานซีพียูแบบวนรอบ ที่มีเวลาควอนตัมเท่ากับ 2.0 หน่วยเวลา ให้แสดงวิธีทำเพื่อคำนวณหาเวลาครบวงงานเฉลี่ย

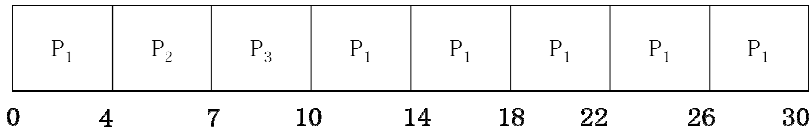
เวลา	โพรเซสที่ทำงาน
0.0-2.0	P1
2.0-4.0	P2
4.0-5.0	P3 ... งานเสร็จเรียบร้อย
5.0-7.0	P1
7.0-9.0	P2 ... งานเสร็จเรียบร้อย
9.0-11.0	P1
11.0-13.0	P1 ... งานเสร็จเรียบร้อย

Process	Arrival Time	Run Time	Start Time	Finish Time	Turnaround Time
P1	0.0	8	0.0	13.0	13.0
P2	0.4	4	2.0	9.0	8.6
P3	1.0	1	4.0	5.0	4.0
				รวม	25.6
เวลาครบวงงานเฉลี่ย = 25.6/3 = 8.53 หน่วยเวลา					

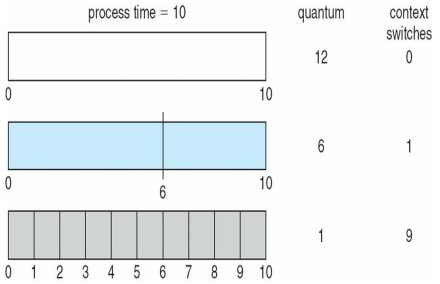
ตัวอย่างที่ 9 เมื่อกำหนดให้ใช้การจัดลำดับใช้งานซีพียูแบบวนรอบ ที่มีเวลาควอนตัมเท่ากับ 4.0 หน่วยเวลา

Process	Burst Time
P_1	24
P_2	3
P_3	3

สามารถแสดงได้ด้วย Gantt Chart ดังนี้:



เมื่อพิจารณาจะพบว่า โดยปกติแล้วเวลาครบวงงานเฉลี่ย (Turnaround time) จะมีค่าสูงกว่าอัลกอริทึมแบบ Shortest-Job-First (SJF) Scheduling แต่มีการตอบสนองที่ดีกว่าประสิทธิภาพของอัลกอริทึมแบบวนรอบขึ้นอยู่กับขนาดของเวลาควอนตัมที่กำหนด ถ้าเวลาควอนตัมมีขนาดใหญ่มาก พบว่าการทำงานของอัลกอริทึมแบบวนรอบจะเหมือนกับการทำงานของอัลกอริทึมแบบมาก่อนบริการก่อน ถ้าเวลาควอนตัมมีขนาดเล็ก เช่น 1 หน่วยเวลา จะทำให้แต่ละโพรเซสรู้สึกเหมือนว่ามีซีพียูเป็นของตัวเอง เนื่องจากโพรเซสมีโอกาสเข้าไปทำงานในซีพียูตลอดเวลา แต่สิ่งที่ตามมาคือ เวลาที่เสียไปในการสลับการทำงานจากโพรเซสหนึ่งไปยังอีกโพรเซสหนึ่ง ที่เรียกว่าการทำ Context switching ที่ต้องมีการเก็บข้อมูลต่าง ๆ ของโพรเซสที่กำลังทำงานในซีพียูและที่สลับออกไป

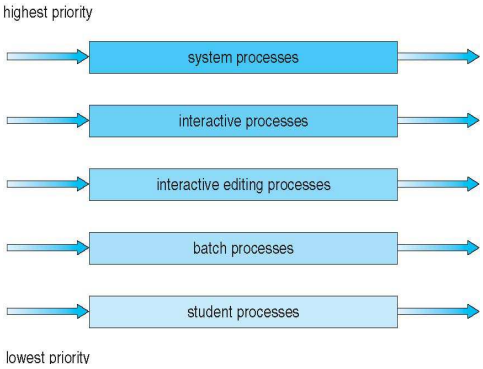


ภาพที่ 6.6 แสดงเวลาควอนตัมและเวลาในการสลับการทำงานโพรเซส
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.273)

6.5 คิวหลายระดับ

อัลกอริทึมของการจัดลำดับวิธีนี้ ถูกสร้างขึ้นจากแนวความคิดที่ว่า โพรเซสสามารถถูกแบ่งออกเป็นกลุ่มต่าง ๆ ได้หลายกลุ่ม เช่น โพรเซสของระบบ (System process) โพรเซสแบบกลุ่ม (Batch process) และโพรเซสแบบโต้ตอบ (Interactive process)

โพรเซสแต่ละกลุ่มจะมีเวลาการตอบสนอง (Response time) ที่แตกต่างกัน จึงต้องการการจัดลำดับที่แตกต่างกันด้วย เช่น โพรเซสแบบโต้ตอบต้องการได้รับการตอบสนองที่รวดเร็ว ควรได้ลำดับการทำงานก่อนโพรเซสแบบกลุ่ม ขั้นตอนวิธีในการจัดตารางการทำงานแบบแถวคอยหลายชั้นนี้ เริ่มจากการจัดแถวพร้อมของระบบออกเป็นหลาย ๆ แถวแยกจากกัน ดังแสดงในภาพที่ 6.7



ภาพที่ 6.7 แสดงการแบ่งระดับความสำคัญในการเข้าคิวหลายระดับขิงโพรเซส
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.273)

จากภาพที่ 6.7 แสดงการจัดลำดับแบบคิวหลายระดับ ที่แบ่งโพรเซสในคิวพร้อมออกเป็นคิวย่อย (Sub queue) 4 คิวย่อย ที่มีระดับความสำคัญแตกต่างกัน เมื่อมีโพรเซสใหม่เข้ามาในระบบ จะถูกนำไปเข้าคิวรอที่คิวใดคิวหนึ่ง โดยที่แต่ละคิวนี้สามารถมีการจัดลำดับที่แตกต่างกัน ในการทำงานนั้น โพรเซสที่อยู่ในคิวที่มีค่าลำดับความสำคัญสูงสุดจะถูกทำงานก่อน ส่วนโพรเซสที่อยู่ในกลุ่มที่มีค่าลำดับความสำคัญน้อยกว่าจะถูกทำงานได้ก็ต่อเมื่อ โพรเซสในคิวย่อยที่มีค่าลำดับความสำคัญสูงกว่าถูกทำงานเสร็จเรียบร้อยแล้วเท่านั้น

นอกจากนั้นก็จะต้องมีการจัดตารางการทำงานระหว่างแถวพร้อมเหล่านั้นด้วย โดยทั่วไปจะใช้การจัดตารางการทำงานแบบศักดิ์สูงได้ก่อน ชนิดที่ให้แทรกกลางได้ เช่น แถวสำหรับโพรเซสที่ทำงานแบบโต้ตอบ จะมีศักดิ์สูงกว่าแถวสำหรับโพรเซสที่ทำงานแบบกลุ่ม สมมติว่าในระบบมีแถวพร้อมอยู่ 5 แถว ดังนี้

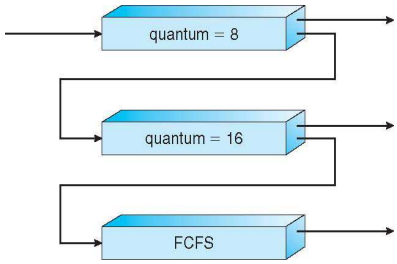
1) งานของระบบ (System processes)
2) งานแบบโต้ตอบ (Interactive processes)
3) งานแก้ไขข้อมูล (Interactive edition processes)
4) งานแบบกลุ่ม (Batch processes)
5) งานของนักศึกษา (Student processes)

โดยแถวบนจะมีศักดิ์สูงกว่าแถวล่าง ดังนั้นโพรเซสที่ทำงานแบบกลุ่มจะสามารถทำงานได้ต่อเมื่อโพรเซสที่อยู่ในแถวพร้อมของงานระบบ งานโต้ตอบ และงานแก้ไขข้อมูล ได้ทำงานจนเสร็จสมบูรณ์แล้ว และในกรณีที่มีโพรเซสใหม่เข้ามาสู่แถวพร้อมของงานแก้ไขข้อมูล ในขณะที่โพรเซสแบบกลุ่มกำลังทำงานอยู่ โพรเซสที่ทำงานแบบกลุ่มจะถูกแทรกกลางทันที หรืออาจจัดแบ่งเวลาของหน่วยประมวลผลให้แต่ละแถว โดยที่แต่ละแถวก็จะไปจัดแบ่งปันเวลาที่ได้รับกันเอง เช่น แถวพร้อมของการทำงานแบบโต้ตอบอาจจะได้รับช่วงเวลาในการใช้งานซีพียูถึง 80 เปอร์เซ็นต์ ในขณะที่แถวพร้อมของการทำงานแบบกลุ่มได้รับช่วงเวลาดังกล่าวเพียง 20 เปอร์เซ็นต์

6.5.1 การจัดตารางการทำงานแบบจัดลำดับหลายชั้นแบบเลื่อนชั้นได้ (Multilevel Feedback Queue Scheduling)

โดยปกติแล้วในวิธีการจัดตารางการทำงานแถวคอยหลายชั้น (Multilevel queue scheduling) โพรเซสที่เข้าสู่ระบบจะถูกกำหนดแถวที่แน่นอนตลอดการทำงาน โดยไม่อาจเปลี่ยนแถวได้อีกเลย ซึ่งวิธีดังกล่าวไม่ยืดหยุ่นนัก การจัดตารางการทำงานแบบจัดลำดับหลายชั้นแบบเลื่อนชั้นได้นี้ จะมีวิธีการในการทำงานที่แก้ข้อเสียดังกล่าว

โดยเมื่อโพรเซสใดใช้เวลาในการทำงานกับหน่วยประมวลผลกลางนานเกินไป โพรเซสนั้นจะถูกย้ายลงไปในแถวที่มีค่าศักดิ์ต่ำกว่าแถวเดิม วิธีนี้จะทำให้โพรเซสที่เน้นการรับส่งข้อมูล และโพรเซสโต้ตอบถูกจัดอยู่ในแถวพร้อมที่มีศักดิ์สูงขึ้น ในทำนองเดียวกันเมื่อโพรเซสใดโพรเซสหนึ่งรอคอยอยู่เป็นเวลานานมากแล้วในแถวที่มีศักดิ์ต่ำ โพรเซสนั้นก็สามารถที่จะย้ายไปต่อในแถวที่มีศักดิ์สูงขึ้นได้ ซึ่งวิธีการดังกล่าวเป็นรูปแบบหนึ่งของการเพิ่มศักดิ์ ตัวอย่างเช่น ระบบที่มีแถวพร้อม 3 แถว คือ แถวที่ 0, 1 และ 2 ตามลำดับ (จากภาพที่ 6.8)



ภาพที่ 6.8 แถวคอยแบบป้อนกลับหลายระดับ
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.276)

โดยตัวจัดตารางการทำงานจะเริ่มจัดให้โพรเซสที่อยู่ในแถวที่ 0 ทำงานจนเสร็จหมดทั้งแถวเสียก่อน จากนั้นจึงจะเริ่มจัดให้โพรเซสที่อยู่ในแถวที่ 1 ได้เข้ารับการทำงานต่อไป ในทำนองเดียวกันการทำงานของโพรเซสในแถวที่ 2 เริ่มได้ก็ต่อเมื่อโพรเซสในแถวที่ 0 และที่ 1 ทำงานจนเสร็จหมดแล้ว และในขณะที่โพรเซสที่อยู่ในแถวที่ 2 กำลังทำงานอยู่ ถ้ามีโพรเซสใหม่เข้ามาในแถวที่ 1 โพรเซสที่กำลังทำงานอยู่ในแถวที่ 2 จะถูกแทรกกลางคัน หรือในขณะที่โพรเซสในแถวที่ 1 กำลังทำงานอยู่นั้น ได้มีโพรเซสใหม่เข้ามาในแถวที่ 0 โพรเซสที่กำลังทำงานอยู่ในแถวที่ 1 ก็จะถูกแทรกกลางคันเช่นเดียวกัน

ถ้ามีโพรเซสใหม่เข้ามาในระบบ โพรเซสนั้นจะถูกจัดให้เข้าทำงานในแถวที่ 0 แต่ถ้าไม่สามารถทำงานให้เสร็จสมบูรณ์ได้ภายใน 1 ส่วนแบ่งเวลาของแถวที่ 0 (ซึ่งมีค่าเท่ากับ 8 มิลลิวินาที) โพรเซสนั้นจะถูกเลื่อนลงไปต่อข้างท้ายของแถวที่ 1 และเมื่อโพรเซสในแถวที่ 0 ทำงานจนเสร็จหมดแล้ว ตัวจัดตารางการทำงานก็จะเริ่มมาจัดให้โพรเซสที่อยู่ในแถวที่ 1 ได้ทำงานต่อไป โดยแถวที่ 1 นี้ (ส่วนแบ่งเวลามีค่าเท่ากับ 16 มิลลิวินาที) ซึ่งถ้ามีโพรเซสใดไม่อาจทำงานให้เสร็จสมบูรณ์ได้ภายใน 1 ส่วนแบ่งเวลาที่กำหนด โพรเซสนั้นก็จะถูกนำไปต่อท้ายของแถวที่ 2 ซึ่งเป็นแถวสุดท้ายและมีการทำงานแบบมาก่อน- ได้ก่อน โดยโพรเซสที่อยู่ในแถวสุดท้ายนี้จะได้ทำงานก็ต่อเมื่อ แถวที่ 0 และ 1 ว่างแล้ว

จากขั้นตอนวิธีในการทำงานข้างต้น จะพบว่า โพรเซสที่ใช้ช่วงประมวลผลไม่เกิน 8 มิลลิวินาที เสมือนมีศักดิ์สูงที่สุด (เช่น โพรเซสที่ใช้เวลาส่วนใหญ่ทำงานด้านรับส่งข้อมูล) ส่วนโพรเซสที่ใช้ช่วง

ประมวลผลระหว่าง 8 – 24 มิลลิวินาที ก็เสมือนมีศักดิ์ต่ำลงมาอีกชั้นหนึ่ง และสำหรับโพรเซสที่ต้องใช้ช่วงเวลาประมวลผลนาน (มากกว่า 24 มิลลิวินาที) ก็จะเคลื่อนลงสู่แถวที่ 2 ซึ่งมีการจัดตารางแบบมาก่อน-ได้ก่อน เสมือนมีศักดิ์ต่ำที่สุด โดยทั่ว ๆ ไปแล้ว การทำงานโดยวิธีการจัดตารางการทำงานแบบจัดลำดับหลายชั้นแบบเลื่อนชั้นได้นี้ ถูกกำหนดโดยพารามิเตอร์ดังนี้

1) จำนวนแถวพร้อม
2) ขั้นตอนวิธีในการจัดตารางการทำงานของแต่ละแถว
3) วิธีที่จะใช้ในการพิจารณาเพื่อที่จะยกระดับให้โพรเซสที่มีค่าศักดิ์สูงขึ้น
4) วิธีที่จะใช้ในการพิจารณาเพื่อที่จะลดระดับให้โพรเซสที่มีค่าศักดิ์น้อยลง
5) วิธีที่จะใช้ในการพิจารณาว่า เมื่อโพรเซสเข้ามาในระบบ ควรจะให้อยู่ในแถวใด

จากนิยามของตัวจัดตารางการทำงานดังกล่าวข้างต้น จะพบว่า เป็นการรวบรวมเอาวิธีการในการจัดตารางหลายวิธีเข้าไว้ด้วยกัน ทำให้การทำงานวิธีจัดลำดับหลายชั้นแบบเลื่อนชั้นได้นี้ สามารถที่จะเลือกใช้ขั้นตอนวิธีที่เหมาะสมกับการทำงานในช่วงของการออกแบบ แต่อย่างไรก็ตามถึงแม้ว่าวิธีการจัดตารางการทำงานแบบนี้จะเป็นวิธีที่ใช้ได้ทั่วไปที่สุด แต่ต้องมีการพิจารณาเลือกค่าพารามิเตอร์ต่าง ๆ ให้เหมาะสมกับสภาวะของระบบต่าง ๆ ทั้งยังเป็นระบบที่ซับซ้อนที่สุดอีกด้วย

6.6 การจัดตารางการทำงานสำหรับหลายหน่วยประมวลผล

ถ้าหน่วยประมวลผลมีหลายแบบ (เรียกว่า Heterogeneous system) วิธีการจัดตารางก็จะถูกจำกัดมาก แต่ละหน่วยประมวลผลก็จะมีแถวคอยเป็นของตนเอง เพราะโพรเซสต่าง ๆ ย่อมมีลักษณะเฉพาะที่จะทำงานได้ในหน่วยประมวลผลแบบหนึ่งแบบเดียว เช่น โปรแกรมที่เขียนขึ้นด้วยภาษา Assembly ของ VAX จะไม่สามารถทำงานบนเครื่อง IBM ได้ เป็นต้น โพรเซสจึงต้องเข้าไปทำงานตามหน่วยประมวลผลที่เหมาะสม โดยแยกแถวคอยกันเอง

ถ้าหน่วยประมวลผลเป็นแบบเดียวกันหมด (เรียกว่า Homogenous system) สามารถเฉลี่ยแบ่งงานกันทำได้ดีขึ้น โดยอาจให้มีแถวคอยแยกแต่ละหน่วยประมวลผล แต่วิธีนี้อาจทำให้หน่วยประมวลผลบางตัวว่างงาน ในขณะที่บางตัวทำงานหนัก ดังนั้นจึงควรใช้แถวคอยร่วมแถวเดียวกันให้ทุก ๆ โพรเซสเข้าแถวคอยแถวเดียวกันหมด เมื่อมีหน่วยประมวลผลใดว่างก็จะให้รับงานไปจากแถวคอยนี้ การจัดแถวคอยร่วมนี้ อาจแบ่งได้เป็น 2 วิธี

1) ให้หน่วยประมวลผลแต่ละตัวจัดตารางการทำงานเอง โดยเลือกงานจากแถวคอยเดียวกัน ปัญหาหลักของวิธีนี้คือ การที่หน่วยประมวลผลใช้ข้อมูลร่วมกัน (แถวคอยร่วมกัน) ย่อมต้องการการประสานงานที่ดี เพื่อป้องกันปัญหาเขตวิกฤต จำเป็นต้องทำให้แน่ใจว่าจะไม่มีหน่วยประมวลผลใดเลือกโพรเซสซ้ำกัน

2) กำหนดให้หน่วยประมวลผลหนึ่งมีหน้าที่จัดตารางการทำงานโดยเฉพาะ คอยจัดตารางการทำงานให้ทุก ๆ หน่วยที่เหลือ วิธีนี้เรียกว่า วิธีเจ้านายและทาส (Master-slave) หรือเรียกว่าการทำงานแบบหลายหน่วยประมวลผลชนิดไม่สมมาตร (Asymmetric multiprocessing)

6.7 การจัดตารางการทำงานแบบตอบสนองฉับพลัน

การจัดตารางการทำงานแบบตอบสนองฉับพลัน แบ่งเป็น 2 ประเภท คือ

1) Hard Real-Time System ต้องการเวลาที่คงที่แน่นอนตายตัว โดยทั่วไปโพรเซสจะได้รับเวลาจำนวนหนึ่งเพื่อนำไปทำงานให้เสร็จ ตัวจัดตารางการทำงานจะให้สิทธิโพรเซส เพื่อรับประกันว่าโพรเซสจะทำงานเสร็จตามเวลา หรือไม่ก็ปฏิเสธการร้องขอของโพรเซสนั้น ถ้ามั่นใจว่าจะทำงานไม่เสร็จได้ตามเวลา การทำงานแบบนี้ถูกเรียกว่า การจองทรัพยากร (Resource reservation)

2) Soft Real-Time System เป็นระบบที่ทำงานโดยไม่ต้องมีเวลามาจำกัด นั่นหมายถึงจะมีโพรเซสอยู่โพรเซสหนึ่งมีลำดับความสำคัญสูงกว่าโพรเซสอื่น การทำงานแบบ Soft Real-Time system อาจจะไม่เหมาะกับ Time sharing เพราะเกิดการล่าช้าหรือปัญหาการอดตาย แต่เหมาะกับระบบทั่ว ๆ ไป ที่สนับสนุนด้านมัลติมีเดีย กราฟฟิก เป็นต้น

เพื่อที่จะลดเวลาในการหยุดโพรเซสหนึ่งเพื่อให้อีกโพรเซสหนึ่งทำงาน (Dispatch latency) เราต้องอนุญาตให้โปรแกรมเรียกระบบที่สามารถแทรกกลางคันได้ วิธีหนึ่งที่ใช้ได้คือ การแทรกโปรแกรมโดยทำการเรียกระบบที่ทำงานในระยะยาวและเรียกว่าจุดแทรกกลางคัน (Preemption point) เพื่อตรวจสอบว่าโพรเซสที่มีลำดับความสำคัญสูงต้องได้ทำงานก่อน เมื่อโพรเซสดังกล่าวเสร็จงาน โพรเซสที่ถูกขัดจังหวะจึงจะได้ทำงานต่อ

อะไรจะเกิดขึ้นถ้าโพรเซสที่มีลำดับความสำคัญสูงกว่าต้องการอ่านหรือปรับปรุงข้อมูลใน Kernel ในขณะที่โพรเซสที่มีลำดับความสำคัญต่ำกว่ากำลังทำงาน โพรเซสที่มีลำดับความสำคัญสูงกว่าควรจะรอให้โพรเซสที่มีลำดับความสำคัญต่ำกว่าทำงานเสร็จก่อน สถานการณ์นี้ถูกเรียกว่า การกลับสิทธิ (Priority inversion) ปัญหานี้แก้ได้โดยสนธิสัญญาการสืบทอดศักดิ์ (Priority-inheritance protocol) ซึ่งโพรเซสเหล่านี้ (โพรเซสซึ่งกำลังเข้าถึงทรัพยากรที่โพรเซสที่มีลำดับความสำคัญสูงต้องการ) สืบทอดลำดับความสำคัญที่สูงจนกระทั่งทำงานเสร็จลำดับความสำคัญจะถูกกลับ (Revert) ไปเป็นดังเดิม โดยระยะที่เกิดการขัดแย้งกัน (Conflict phase) ของ Dispatch latency มี 2 องค์ประกอบดังนี้

1) เกิดการแทรกกลางคันของโพรเซสที่กำลังทำงานใน Kernel
2) โพรเซสที่มีลำดับความสำคัญต่ำปลดปล่อยทรัพยากรให้โพรเซสที่มีลำดับความสำคัญสูง

6.8 การประเมินอัลกอริทึม

จะเห็นว่ามีวิธีการจัดตารางได้หลายวิธี แต่ละวิธีมีตัวแปรและลักษณะเฉพาะ ทำให้การเลือกวิธีที่เหมาะสมหรือดีที่สุดทำได้ยาก ขั้นแรกจำเป็นต้องกำหนดคุณสมบัติก่อน ว่าต้องการคุณสมบัติใดมีค่าเท่าใด เช่น ให้ได้ประสิทธิผลการใช้ซีพียูสูงสุดที่เวลาตอบสนองนานที่สุดไม่เกิน 1 วินาที และให้มีอัตรางาน (Throughput) เสร็จสูงที่สุด โดยที่วงรอบการทำงาน (Turnaround time) โดยเฉลี่ยเป็นสัดส่วนโดยตรงกับการประมวลผลจริง

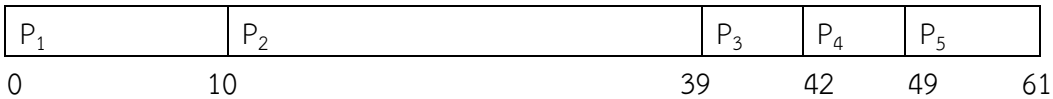
6.8.1 การกำหนดโมเดล (Deterministic Modeling)

วิธีนี้ทำโดยการกำหนดกลุ่มงานทดสอบขึ้นมาหนึ่งกลุ่ม แล้วคำนวณค่าคุณสมบัติของวิธีการจัดตารางแต่ละแบบ ตัวอย่างเช่น กำหนดกลุ่มงานดังนี้

โพรเซส (Process)	เวลาทำงาน (Burst Time)
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

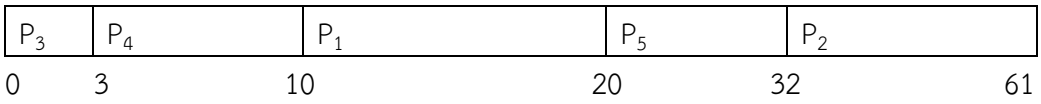
ให้ทั้ง 5 โพรเซสมาถึงระบบในเวลาเดียวกัน ที่เวลา 0 ตามลำดับ พิจารณาใช้วิธีการจัดตารางการทำงาน 3 แบบ คือ FCFS, SJF, RR (โดยมีส่วนแบ่งเวลา 10 มิลลิวินาที) แล้วคำนวณหาเวลารอคอยเฉลี่ยต่ำที่สุด

<u>วิธีมาก่อน-ได้ก่อน (FCFS)</u>



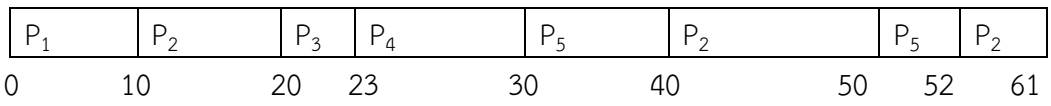
เวลารอคอยเฉลี่ย = (0 + 10 + 39 + 42 + 49) / 5 = 28 มิลลิวินาที

<u>วิธีสั้นที่สุดได้ก่อน (SJF)</u>



เวลารอคอยเฉลี่ย = (10 + 32 + 0 + 3 + 20) / 5 = 13 มิลลิวินาที

<u>วิธีเวียนเทียน (RR) (ส่วนแบ่งเวลา = 10 มิลลิวินาที)</u>



เวลารอคอยเฉลี่ย = (0 + 32 + 20 + 23 + 40) / 5 = 23 มิลลิวินาที

จากตัวอย่างนี้จะเห็นว่า วิธีงานที่สั้นที่สุดได้ก่อน มีค่าต่ำที่สุด (ดีที่สุด) คือ เพียงครึ่งเดียวของวิธีมาก่อน-ได้ก่อนและวิธีเวียนเทียนมีค่าปานกลาง

วิธีการกำหนดโมเดลนี้ เป็นวิธีที่ง่ายและสะดวก ผลลัพธ์ที่ได้ก็สามารถเปรียบเทียบเป็นตัวเลขได้โดยตรง แต่มีข้อเสียคือ ผลลัพธ์ที่ได้อาจเป็นเฉพาะกรณีตามกลุ่มงานที่กำหนดขึ้นเท่านั้น วิธีนี้เหมาะเพียงการอธิบายการจัดตารางแบบต่าง ๆ เพื่อเป็นโมเดลการใช้งานที่เหมาะสม เช่น จากโมเดลต่าง ๆ ที่กล่าวมาพอจะสรุปได้จากตัวอย่างว่า วิธีสั้นที่สุดได้ทำงานก่อน ให้เวลารอคอยเฉลี่ยสั้นที่สุด

6.8.2 การวิเคราะห์แถว (Queuing Models)

ระบบคอมพิวเตอร์อาจถือได้ว่าเป็นเครือข่ายของผู้ให้บริการ (Network of servers) ผู้ให้บริการแต่ละตัวมีแถวคอยของตนเอง หน่วยประมวลผลกลาง (ผู้ให้บริการประมวลผล) มีแถวพร้อมของตนเอง อุปกรณ์รับส่งข้อมูลแต่ละตัว (บริการรับส่งข้อมูล) ก็มีแถวคอยอุปกรณ์ของตน โดยใช้ค่าอัตราการมาถึงระบบ (Arrival rate) กับเวลาในการประมวลผล (Service rate) เราสามารถคำนวณหาค่าเฉลี่ยของประสิทธิผล ความยาวแถวคอย เวลารอคอย และอื่น ๆ ได้ เราเรียกกรรมวิธีนี้ว่า การวิเคราะห์เครือข่ายของแถวคอย (Queuing-network analysis)

ตัวอย่าง ให้ n เป็น ขนาดของแถวคอยเฉลี่ย (ไม่รวมโพรเซสที่กำลังทำงานอยู่) W เป็น เวลารอคอยเฉลี่ยในแถวคอย λ เป็น อัตราเฉลี่ยของจำนวนโพรเซสที่มาถึงแถวคอย (เช่น 3 โพรเซสต่อวินาที) จะเห็นได้ว่าในช่วงเวลา W ขณะที่โพรเซสหนึ่งต้องรอคอยในแถวคอย มีโพรเซสใหม่บางระบบเท่ากับ λ x W ถ้าระบบอยู่ในภาวะคงที่ จำนวนโพรเซสที่ออกจากแถวคอยจะต้องเท่ากับ จำนวนโพรเซสที่มาถึง ดังนั้น

$$n = \lambda \times W$$

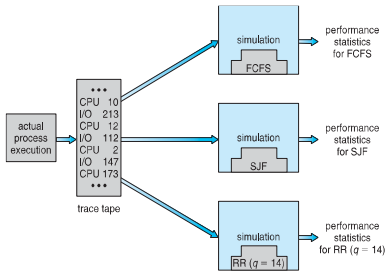
สมการนี้มีชื่อว่า สูตรของ Little (Little's Formula) สูตรนี้มีประโยชน์มากในการคำนวณ เพราะเป็นจริงเสมอ ไม่ว่าจะเป็นวิธีการจัดตารางแบบใด และการกระจายตัวของข้อมูลแบบใดก็ตาม สามารถใช้สูตรของ Little หาค่าตัวแปรหนึ่งในสามตัว เมื่อทราบคู่ตัวแปรอีกสองตัว เช่น เรารู้ว่าโพรเซสมาถึงระบบโดยเฉลี่ย 7 โพรเซสต่อวินาทีและแถวคอยมีความยาวเฉลี่ย 14 โพรเซส ดังนั้นเวลารอคอยเฉลี่ยของแต่ละโพรเซสจะมีค่าเท่ากับ 2 วินาที

6.8.3 การจำลองสถานการณ์ (Simulations)

เพื่อให้ได้ผลลัพธ์ถูกต้องใกล้เคียงมากขึ้น เราอาจจำลองเหตุการณ์จริงขึ้น โดยเขียนโปรแกรมตัวแบบของระบบคอมพิวเตอร์ และจำลองอุปกรณ์ต่าง ๆ ในระบบด้วยโครงสร้างข้อมูลที่เหมาะสม มีโปรแกรมนาฬิกาให้โปรแกรมทำงานเสมือนมีเหตุการณ์เกิดขึ้นจริง (โดยใช้ตัวแปรต่าง ๆ แทนความจริง) ขณะที่ระบบจำลองทำงาน เราสามารถเก็บสถิติต่าง ๆ ของระบบ เพื่อนำมาวิเคราะห์ประสิทธิภาพของวิธีการจัดตารางแบบต่าง ๆ ได้

ข้อมูลของโพรเซสที่เข้าระบบ สร้างโดยใช้ตัวแปรสร้างที่สุ่มค่าตัวเลขขึ้นมา (Random-number generator) ค่าที่มากที่สุด ได้แก่ ช่วงเวลาประมวลผล (CPU-burst time) เวลามาถึง (Arrival) เวลางานเสร็จ (Departure) เป็นต้น การสร้างค่าเหล่านี้ ทำโดยการใช้ความน่าจะเป็น (เช่น แบบคงที่ แบบ Exponential และแบบ Poisson) หรือแบบทั่วไป ซึ่งจะสามารถเลือกได้มากแบบกว่าการใช้วิธีวิเคราะห์แถวคอย เพราะไม่ต้องคำนวณผลลัพธ์เอง ผลลัพธ์จะได้จากโปรแกรมแบบจำลองการทำงานแทนระบบจริง

ผลลัพธ์จากการใช้ตัวแบบสุ่มเหล่านี้อาจไม่ถูกต้องเท่าที่ควร เนื่องจากข้อมูลจริงแตกต่างจากข้อมูลที่ได้จากโปรแกรม ดังนั้นเราอาจใช้ข้อมูลจริงจากระบบจริงเก็บเข้าไปในเทป (Trace tape) แล้วนำมาเป็นข้อมูลเข้าโปรแกรมแบบจำลองเพื่อศึกษาเปรียบเทียบ วิธีการจัดตารางแบบต่าง ๆ ให้ใกล้เคียงความจริงมากที่สุด ดังภาพที่ 6.9



ภาพที่ 6.9 การประเมินโดยจำลองการตัวจัดตาราง CPU
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.303)

การจำลองเหตุการณ์จริงนี้อาจมีค่าใช้จ่ายมากได้ เช่น ต้องใช้เวลาประมวลผลนาน ยิ่งต้องการผลลัพธ์ละเอียดถูกต้องยิ่งต้องทดลองนาน นอกจากนี้การออกแบบและเขียนโปรแกรมนี้ยังเสียเวลามากอีกด้วย

6.8.4 การปฏิบัติจริง (Implementation)

วิธีนี้มีปัญหาสำคัญ คือ ค่าใช้จ่ายสูงมาก ไม่เพียงแต่ค่าใช้จ่ายในการเขียนโปรแกรมและแก้ไขระบบปฏิบัติการเท่านั้น แต่ยังมีผลกระทบต่อผู้ใช้โดยตรงด้วย เพราะจะต้องมีการแก้ไขระบบบ่อย ๆ นอกจากนี้การเปรียบเทียบวิธีต่าง ๆ ด้วยวิธีนี้อาจได้ผลไม่ถูกต้อง เพราะสภาพแวดล้อมของระบบอาจเปลี่ยนไป

เราอาจกำหนดตัวแปรในระบบให้ผู้ควบคุมระบบสามารถแก้ไขวิธีจัดตารางได้ตลอดเวลา (ขณะทำงานจริง) เช่น ถ้าต้องการใช้พิมพ์เช็คอย่างเร่งด่วน (ปกติถือเป็นงานแบบกลุ่มและมีลำดับความสำคัญต่ำ) ผู้ควบคุมระบบอาจกำหนดให้งานนี้มีศักดิ์สูงเป็นการชั่วคราว แต่เป็นที่น่าเสียดายว่ามีน้อยระบบที่มีตัวแปรแบบนี้

6.9 สรุป

การจัดตารางการทำงานของซีพียูให้ได้ผลดี ต้องทราบถึงลักษณะการทำงานของโพรเซส ซึ่งโพรเซสโดยทั่วไปจะทำงานเป็นวงจรคือ ทำงานในซีพียูและรอคอยการรับส่งข้อมูลสลับกันไป โดยเริ่มต้นที่ทำงานในซีพียูก่อนเรียกว่า ช่วงประมวลผล แล้วก็จะหยุดเพื่อทำงานในอุปกรณ์ที่มีการรับส่งข้อมูลเรียกว่า ช่วงรับส่งข้อมูล และต่อมาจะกลับมาช่วงประมวลผลอีกสลับกันไปจนสุดท้ายจะเป็นช่วงประมวลผล และก็สิ้นสุดโพรเซสจัดเวลาซีพียู เป็นงานของการจัดเวลาใช้งานซีพียู โดยมีหน้าที่หลักก็คือการตัดสินใจเลือก

งานเข้ามาใช้งาน โดยมี Dispatcher เป็นเครื่องมือในการนำเอางานที่ได้รับคัดเลือกนี้เข้าและออกจากการใช้ซีพียู การมาก่อนบริการก่อน (FCFS) เป็นอัลกอริทึมแบบให้สิทธิ์ก่อน (Preemptive) ที่ง่ายที่สุดสำหรับการจัดเวลาซีพียู แต่ก็เป็นระบบการจัดการที่ทำให้งานที่ต้องการใช้ซีพียูเพียงแค่ระยะเวลาสั้น ๆ ต้องคอยงานที่ใช้ซีพียูนาน ๆ ส่วนงานสั้นได้ทำก่อน (SJF) ก็เป็นอัลกอริทึมที่สามารถเป็นทั้งแบบให้สิทธิ์ก่อนและไม่ให้สิทธิ์ก่อน SJF จะให้ค่าเฉลี่ยของการรอคอย (Waiting time) สั้นที่สุด แต่ในการใช้งานจริงจะมีปัญหาในเรื่องของการวัดระยะซีพียูถัดไปของแต่ละงาน SJF มีการทำงานคล้ายกับการมีลำดับขั้นความสำคัญของงาน โดยงานที่มีเวลาซีพียูสั้น ๆ จะเป็นงานที่มีความสำคัญมาก ทำให้มีโอกาสเข้าไปใช้ซีพียูก่อน ซึ่งอาจทำให้เกิดปัญหาการตกค้างของงานที่มีความสำคัญต่ำ ถ้าหากว่าระบบคอมพิวเตอร์มีงานมาก การแก้ไขก็ด้วยการใช้วิธีเพิ่มความสำคัญให้กับงานตามระยะเวลาที่ต้องคอย

อัลกอริทึมแบบวนรอบ เป็นระบบการจัดเวลาแบบให้สิทธิ์ก่อนเหมาะสมกับระบบคอมพิวเตอร์แบบแบ่งเวลา หรือการทำงานแบบ Interactive เพราะจะทำให้มีเวลาวนรอบที่คงที่แน่นอน เนื่องจากว่าอัลกอริทึมแบบวนรอบมีการกำหนดช่วงเวลาที่จำกัดในการเข้าใช้ซีพียูของแต่ละงานที่คงที่ที่เรียกว่าเวลาควอนตัม (Quantum time) ซึ่งงานทุกงานจะมีโอกาสได้รับซีพียูมาใช้อย่างมากก็เพียงแค่ระยะเวลาของควอนตัมที่กำหนดเท่านั้น ซึ่งการกำหนดค่าของควอนตัมนี้ ต้องมีการกำหนดให้พอดี ถ้ามีการกำหนดที่ยาวเกินไปการทำงานก็จะมีประสิทธิภาพใกล้เคียงกับ FCFS แต่ถ้าควอนตัมถูกกำหนดไว้สั้นเกินไป ระบบก็จะเสียเวลาไปกับการนำงานเข้าและออก หรือเวลาในการทำ Context switch มากเกินไปจนทำให้อัตรางานของระบบต่ำลง

คิวแบบหลายระดับ อาจมีการรวมเอาอัลกอริทึมหลายชนิดเข้าไว้ด้วยกันเพื่อให้สามารถรองรับระบบงานที่มีงานหลายประเภทมากขึ้น โดยจะให้บริการแก่งานต่าง ๆ อย่างเท่าเทียมกันตามระดับความสำคัญของงานนั้น ๆ และการใช้คิวแบบ Multilevel feedback ก็จะยอมให้มีการปรับเปลี่ยนระดับความสำคัญของตัวงานได้ ด้วยการยอมให้งานสามารถเลื่อนขึ้นหรือลงในคิวแบบหลายระดับเพื่อแก้ปัญหาของการที่งานบางชนิดจะไม่มีโอกาสได้รับซีพียูมาใช้เลย

การตัดสินใจว่าจะนำเอาอัลกอริทึมชนิดไหนมาใช้ก็มีวิธีการอยู่มากมาย การใช้วิธีคำนวณจากสูตรนั้นมีประโยชน์ในการคำนวณหาประสิทธิภาพของระบบโดยรวมแบบคร่าว ๆ ในขณะที่วิธีการจำลองแบบหรือ Simulation นั้น จะสามารถทำให้การตัดสินใจมีความถูกต้องมากขึ้น แต่ก็ต้องแลกมาด้วยแรงงานและเวลา ซึ่งจะทำให้มีผลกระทบต่อต้นทุนที่สูงขึ้น แต่ในบางระบบคอมพิวเตอร์ที่ต้องการประสิทธิภาพสูงสุดโดยไม่สนใจว่าต้องใช้ต้นทุนเท่าไร วิธี Simulation ก็จะต้องเป็นเรื่องที่หลีกเลี่ยงไม่ได้ อย่างไรก็ตาม วิธีที่ดีที่สุดก็คือ การสร้างอัลกอริทึมต่าง ๆ ขึ้นมาและทดลองใช้ระบบคอมพิวเตอร์ที่ทำงานกับงานจริง ในปัจจุบันสิ่งที่เป็นความต้องการของระบบคอมพิวเตอร์ก็คือ ความต้องการที่จะให้ระบบปฏิบัติการมีความสามารถในการปรับเปลี่ยนหรือปรับแต่งอัลกอริทึม ที่ใช้ได้ตามลักษณะของงานที่เปลี่ยนแปลงไป

แบบฝึกหัดท้ายบทที่ 6

จงตอบคำถามต่อไปนี้

1) จงอธิบายการให้สิทธิ์การจัดเวลา มีหลักการทำงานอย่างไร

2) จงอธิบายถึงข้อพิจารณาในการจัดเวลาของซีพียูมีอะไรบ้าง

3) จงอธิบายหลักการทำงานของอัลกอริทึมการจัดเวลาต่อไปนี้มาพอเข้าใจ

- 3.1) การจัดเวลาแบบมาก่อนได้ก่อน
- 3.2) การจัดเวลาแบบงานสั้นทำก่อน
- 3.3) การจัดเวลาตามลำดับความสำคัญ
- 3.4) การจัดเวลาแบบวนรอบ
- 3.5) การจัดเวลาแบบคิวหลายระดับ

4) การกำหนดการใช้ซีพียู โดยใช้อัลกอริทึมในข้อ 3 แต่ละอัลกอริทึมมีปัญหาสำคัญที่อาจเกิดขึ้นได้คืออะไร เกิดขึ้นได้อย่างไร และมีวิธีแก้ปัญหาที่เกิดขึ้นได้อย่างไร จงอธิบายพร้อมยกตัวอย่างประกอบ

5) ถ้าให้โพรเซสเข้าสู่ระบบตามตารางดังนี้

Process	Arrival Time	Burst Time (ms)
P1	0	5
P2	5	20
P3	10	10
P4	15	15

ถ้าโพรเซสเข้ามาตามลำดับคือ P1, P2, P3, P4 จงแสดงถึงการจัดเวลาซีพียูของโพรเซสตามอัลกอริทึมแบบ FCFS, SJF, Priority (ถ้าตัวเลขน้อยคือ ค่า Priority สูง) และ วิธีการของ RR เมื่อ Quantum = 1 จงหาค่าเฉลี่ยของการคอยในคิวของแต่ละโพรเซสโดยใช้อัลกอริทึมแบบ FCFS, SJF และ RR

6) ถ้าให้ set ของโพรเซสเข้ามาสู่ระบบ (ใน Ready queue) ตามตารางดังนี้

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

ถ้าโพรเซสเข้ามาตามลำดับคือ P1, P2, P3, P4, P5

6.1) จงวาด Gantt Chart สำหรับการแสดงลำดับการอยู่ใน Ready Queue และการทำงานในรูปแบบของ FCFS, SJF, Non-preemptive Priority (ถ้าตัวเลขน้อยคือ ค่า Priority สูง) และ วิธีการของ RR เมื่อ quantum =1

6.2) จงหาค่า Turnaround Time ของแต่ละโพรเซสใน Algorithm ต่าง ๆ

6.3) จงหาค่า Waiting Time ของแต่ละโพรเซสใน Algorithm ต่าง ๆ

6.4) Scheduling Algorithm ในข้อใด (รูปแบบใด) ให้ค่า Average Minimum Time น้อยที่สุด

7) โพรเซสมาถึงระบบโดยเฉลี่ย 10 โพรเซสต่อวินาทีและแถวคอยมีความยาวเฉลี่ย 20 โพรเซส จงหาเวลารอคอยเฉลี่ยของแต่ละโพรเซสจะมีค่าเท่ากับเท่าไร

เอกสารอ้างอิง

พิเชษฐ์ ศิริรัตนไพศาลกุล. (2548). **ระบบปฏิบัติการ**. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

ยรรยง เต็งอำนวย. (2533). **ระบบปฏิบัติการ.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

สุจิตรา อดุลย์เกษม. (2552). **ทฤษฎี ระบบปฏิบัติการ Operating Systems**. กรุงเทพฯ : โปรวิชั่น.

Abraham Silberschatz, Peter Baer Galvin, Greg Gagne. (2013). **Operating System Concepts. 9th ed.** Wiley & Sons, Inc.

Andrew S. Tanenbaum, Herbert Bos. (2014). **Modern Operating Systems. 4th ed.** Prentice Hall, Pearson Education International.