บทที่ 4
การจัดเวลาซีพียู

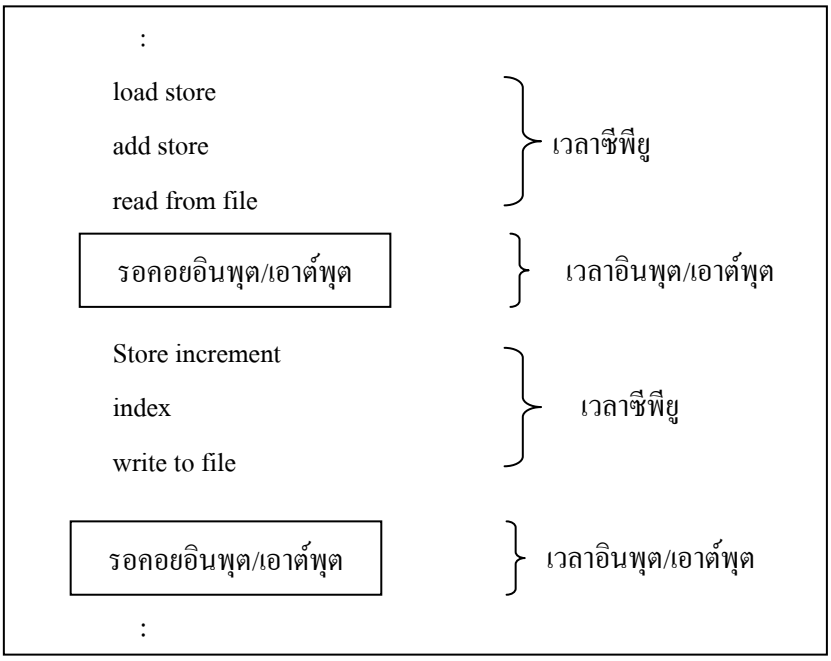
4.1 ความนำ

ในการทำงานของคอมพิวเตอร์นั้น งานทุกอย่างจะต้องถูกกระทำโดยซีพียู ซึ่งในระบบคอมพิวเตอร์โดยทั่วไปมักจะเป็นสถาปัตยกรรมที่มีซีพียูเพียงแค่หนึ่งตัว และจะต้องรองรับการทำงานทั้งหมดที่เกิดขึ้นในระบบคอมพิวเตอร์นั้นๆ หากระบบปฏิบัติการที่ใช้ในระบบคอมพิวเตอร์นั้นไม่มีการจัดสรรการทำงานของซีพียูที่ดีพอ จะทำให้ประสิทธิภาพของการทำงานในระบบคอมพิวเตอร์นั้นลดลง

4.2 การจัดเวลาซีพียู

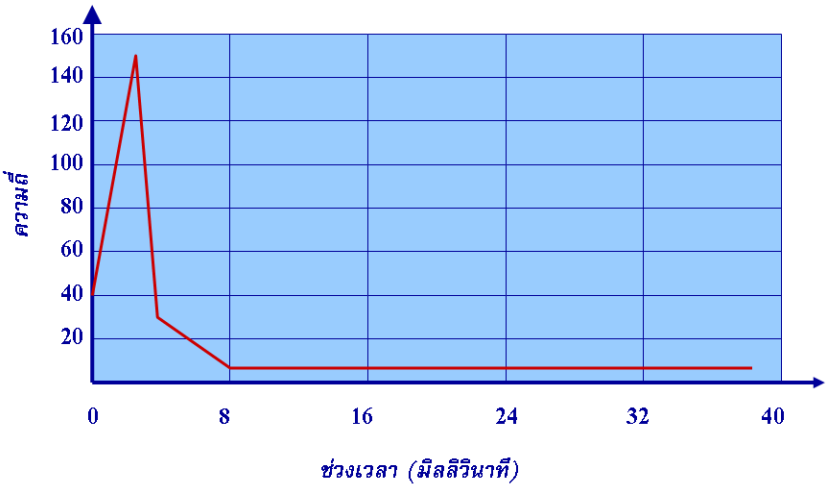
การจัดเวลาซีพียู (CPU Scheduling) เป็นหลักการทำงานหนึ่งของระบบปฏิบัติการที่ทำให้คอมพิวเตอร์มีความสามารถในการเปิดโปรแกรมหลายๆ โปรแกรมในเวลาเดียวกัน ซึ่งการแบ่งเวลาการเข้าใช้ซีพียูให้กับโปรเซส จะทำให้คอมพิวเตอร์สามารถทำงานได้ปริมาณงานที่มากขึ้นกว่าการที่ให้ซีพียูทำงานให้เสร็จทีละโปรเซส

ความสำคัญของการจัดเวลาของซีพียูนั้น ขึ้นอยู่กับคุณลักษณะของโปรเซส โดยทั่วๆ ไป การประมวลผลโปรเซสจะประกอบด้วยเวลาที่ใช้ซีพียู (CPU Burst Cycle) และเวลาที่ต้องคอยอุปกรณ์อินพุต/เอาต์พุต (I/O Burst Cycle) ในขณะที่มีการประมวลผลโปรเซส จะมีการสลับการทำงานระหว่าง 2 ช่วงเวลานี้เท่านั้น และจะเกิดไม่พร้อมกัน การประมวลผลมักจะเริ่มจากการใช้ซีพียู แล้วก็จะตามด้วยการคอยอินพุต/เอาต์พุต เมื่อจบการรอคอย ก็จะตามมาด้วยเวลาของซีพียูสลับกันไปเรื่อย ๆ จนกว่าจะจบการประมวลผล ซึ่งการสิ้นสุดการประมวลผลนี้มักจะเป็นการใช้เวลาซีพียูเพื่อทำการจบหรือสิ้นสุดโปรแกรมมากกว่าการรอคอยอินพุต/เอาต์พุต ดังภาพที่ 4.1



ภาพที่ 4.1 แสดงการทำงานที่หลากหลายที่ใช้เวลาซีพียูและเวลาในการคอยอินพุต/เอาต์พุต ที่มา (http://csnon04.blogspot.com/2008/03/3_06.html, 2552)

ได้มีการศึกษาถึงลักษณะช่วงเวลาของการใช้ซีพียูจนสามารถมองเห็นคร่าวๆ ว่า ลักษณะของช่วงเวลาที่ใช้ซีพียูในแต่ละโปรเซสจะมีรูปแบบอย่างไร ถึงแม้ว่ารูปแบบของเวลาอาจจะมีความแตกต่างกันอยู่บ้าง แต่ก็มีแนวโน้มว่า แต่ละโปรเซสมักจะมีคาบเวลาการเข้าใช้ซีพียูคล้ายๆ กัน ดังภาพที่ 4.2



ภาพที่ 4.2 แสดงลักษณะของช่วงเวลาที่ใช้ซีพียูในแต่ละโปรเซส
ที่มา (http://csnon04.blogspot.com/2008/03/3_06.html, 2552)

จากกราฟอธิบายได้ว่า การใช้ซีพียูของโปรเซสใดๆ นั้น มักจะมีความถี่มากสำหรับเวลาซีพียูช่วงสั้นๆ ส่วนเวลาซีพียูระยะเวลานานๆ นั้นจะมีความถี่น้อยกว่า นอกจากนี้ยังมีแนวโน้มว่าโปรเซสที่มีการติดต่อแลกเปลี่ยนข้อมูลกับอุปกรณ์ภายนอกมากๆ ก็มักจะมีช่วงเวลาของการใช้ซีพียูค่อยข้างสั้นแต่บ่อยครั้ง และมีช่วงเวลาการใช้ซีพียูนานๆ เพียงเล็กน้อย การศึกษาคุณสมบัติของโปรเซสต่างๆ เหล่านี้อย่างละเอียด ทำให้สามารถนำมาใช้เป็นข้อมูลในการเลือกลักษณะของอัลกอริทึมที่เหมาะสม สำหรับการจัดเวลาให้กับซีพียูให้ดีที่สุดในระบบคอมพิวเตอร์แต่ละระบบได้ต่อไป (วิเชษฐ์ พลายมาศ, 2552)

จุดประสงค์ของการใช้งานโปรแกรมหลายโปรแกรมคือ ความต้องการที่จะให้ซีพียูมีการทำงานตลอดเวลา เพื่อให้มีการใช้ซีพียูอย่างเต็มที่ และเต็มประสิทธิภาพ ซึ่งระบบคอมพิวเตอร์ที่มีซีพียูเพียงตัวเดียว ในเวลาใดเวลาหนึ่งซีพียูจะทำงานได้เพียงงานเดียวเท่านั้น ถ้ามีหลายโปรแกรมหรือหลายงาน งานที่เหลือก็ต้องคอยจนกว่าจะมีการจัดการให้เข้าไปใช้ซีพียู

ความคิดและหลักการทำงานกับหลายโปรแกรมในขั้นพื้นฐานนั้น ค่อนข้างจะไม่ซับซ้อน แต่ละโปรแกรมจะถูกทำงาน จนกระทั่งถึงจุดที่มันต้องคอยอะไรซักอย่างเพื่อใช้สำหรับการทำงานในช่วงต่อไป ส่วนมากการคอยเหล่านี้ก็คือการทำงานที่เกี่ยวข้องกับอินพุต/เอาต์พุตนั่นเอง ในระบบคอมพิวเตอร์ที่มีความสามารถเรียกใช้งานโปรแกรมได้ทีละโปรแกรม การทำงานของระบบจะไม่

ซับซ้อน ซีพียูจะหยุดการทำงานในระหว่างที่คอยอินพุต/เอาต์พุตนี้ ซึ่งการคอยเหล่านี้เป็นการเสียเวลาโดยเปล่าประโยชน์ เพราะซีพียูไม่ได้ทำงานเลย

ส่วนหลักในการทำงานหลายโปรแกรม จะพยายามใช้เวลาที่ซีพียูต้องคอยนี้ทำงานอย่างอื่นต่อไป ดังนั้นเมื่อใดก็ตามที่ซีพียูต้องคอย และยังมีโปรแกรมในหน่วยความจำหลายโปรแกรมที่คอยการใช้ซีพียูอยู่ จะจัดให้ซีพียูทำงานในโปรแกรมเหล่านั้นทันที ซึ่งระบบจะจัดการนำเอาโปรแกรมที่ต้องคอยอินพุต/เอาต์พุตออกไปก่อน เพื่อที่จะให้โปรแกรมอื่นที่คอยใช้ซีพียูนี้ สามารถเข้ามาได้ ถ้าทำงานในแบบนี้ไปเรื่อยๆ ซีพียูก็จะได้มีการทำเกือบตลอดเวลากับโปรแกรมหลายๆ โปรแกรมที่อยู่ในระบบ

การจัดเวลาให้กับซีพียู เป็นหลักความต้องการพื้นฐานของระบบปฏิบัติการในคอมพิวเตอร์ ทรัพยากรที่คอมพิวเตอร์มีอยู่ในเครื่องหนึ่งๆ จะถูกจัดสรรการใช้ก่อนการนำไปให้กับโปรแกรมใดๆ ซีพียูเองก็ถือว่าเป็นทรัพยากรของระบบคอมพิวเตอร์ชนิดหนึ่งที่มีความสำคัญมากที่สุดโดยตัวซีพียูนี่เองที่จะเป็นศูนย์กลางของการสร้างระบบปฏิบัติการที่มีความสามารถในการ ทำงานหลายโปรแกรม ในบทนี้จะกล่าวถึงอัลกอริทึมพื้นฐานของการจัดเวลาซีพียู โดยจะอธิบายถึงวิธีการหลักๆ ที่แต่ละอัลกอริทึมมีความแตกต่างกัน ข้อดีข้อเสีย และความเหมาะสมต่อระบบงานแบบต่างๆ เพื่อการเลือกใช้อัลกอริทึมได้อย่างถูกต้อง

4.3 ตัวจัดการเวลาซีพียู

เมื่อใดก็ตามที่ซีพียูว่าง ระบบปฏิบัติการจะต้องเข้ามาเลือกโปรเซสตัวใดตัวหนึ่งที่คอยอยู่ในคิวเข้ามาใช้งานซีพียู การเลือกโปรเซสเพื่อเข้ามาใช้ซีพียูนี้ จะถูกจัดการด้วยส่วนที่เรียกว่าตัวจัดการเวลาช่วงสั้น (Short-term Scheduler) หรือตัวจัดการเวลาซีพียู (CPU Scheduler) ตัวจัดการเวลานี้จะเลือกโปรเซสที่อยู่ในหน่วยความจำที่พร้อมในการประมวลผลที่สุด เพื่อให้ครอบครองเวลาซีพียูและทรัพยากรที่เกี่ยวข้องกับโปรเซสนั้น

คิวของโปรเซสในหน่วยความจำนั้นไม่จำเป็นที่ต้องเป็นแบบมาก่อนได้ก่อน (FIFO : First In First Out) เสมอไป อาจจะเป็นไปตามลำดับความสำคัญ (Priority) โครงร่างต้นไม้ (Tree) หรืออาจจะเป็นลิงค์ลิสต์ (Linked List) ก็ได้ อย่างไรก็ตามโปรเซสทุกโปรเซสที่พร้อมใช้ซีพียู จะต้องมีโอกาสได้เข้าครอบครองเวลาซีพียูไม่เวลาใดก็เวลาหนึ่ง ซึ่งการเข้าและออกจากการครอบครองเวลาซีพียูแต่ละครั้ง จำเป็นต้องมีการเก็บข้อมูลไว้เสมอว่า เข้ามาแล้วได้ทำอะไรไปบ้าง ช่วงต่อไปจะทำอะไร เป็นต้น โดยใช้พื้นที่หน่วยความจำส่วนหนึ่งเก็บข้อมูลของแต่ละโปรเซสหลังเสร็จสิ้นการใช้ซีพียู พื้นที่หน่วยความจำนี้มีชื่อว่า บล็อกควบคุมโปรเซส (PCB : Process Control Block)

4.4 การให้สิทธิการจัดเวลาซีพียู

การตัดสินใจของซีพียูในการเลือกประมวลผลโปรเซสใด ๆ ขึ้นอยู่กับสถานการณ์ดังนี้

1) เมื่อมีการเปลี่ยนสถานะของโปรเซสจากสถานะทำงาน (Run) ไปเป็นสถานะคอย (Wait) เช่นในสภาวะที่คอยอินพุต/เอาต์พุต หรือการคอยให้โปรเซสลูกเสร็จสิ้นไปก่อน

2) เมื่อมีการเปลี่ยนสถานะของโปรเซสจากสถานะทำงาน (Run) เป็นสถานะพร้อม (Ready) เช่นเมื่อมีการขัดจังหวะเกิดขึ้นเป็นต้น

3) เมื่อมีการเปลี่ยนสถานะของโปรเซสจากสถานะคอย (Wait) เป็นสถานะพร้อม (Ready) เช่นเมื่ออินพุต/เอาต์พุตเสร็จสิ้นไปแล้ว เป็นต้น

4) เมื่อโปรเซสเสร็จสิ้นไปแล้ว

หากโปรเซสที่กำลังถูกซีพียูทำการประมวลผลอยู่ เสร็จสิ้นการทำงาน หรือถูกกำหนดให้อยู่ในสถานะสิ้นสุด (Terminate) แม้จะประมวลผลเสร็จสิ้นแบบไม่สมบูรณ์ก็ตาม

ทั้ง 4 สถานการณ์ดังกล่าวข้างต้น ในสถานการณ์ที่ 1 และ 4 นั้นเป็นสถานการณ์ที่จะต้องมีการตัดสินใจทำอะไรอย่างใดอย่างหนึ่งโดยไม่สามารถหลีกเลี่ยงได้ เช่น ต้องเลือกโปรเซสใหม่เข้ามาประมวลผลต่อไป เนื่องจากโปรเซสเดิมไม่ใช้ซีพียูอีกแล้ว แต่สำหรับสถานการณ์ที่ 2 และ 3 นั้น การตัดสินใจต้องอยู่บนพื้นฐานหรือกฎเกณฑ์ของแต่ละอัลกอริทึมที่ใช้ ซึ่งอาจทำให้มีการทำงานแตกต่างกันไป

การตัดสินใจที่เกิดขึ้นเนื่องจากสถานการณ์ที่ 1 และ 4 การจัดเวลาซีพียูจะเป็นแบบไม่ให้สิทธิก่อน (Nonpreemptive) นอกนั้นจะเรียกว่าให้สิทธิก่อน (Preemptive) ภายใต้การทำงานแบบไม่ให้สิทธิก่อนนี้ โปรเซสจะครอบครองเวลาซีพียูไปจนกว่าจะเสร็จสิ้น หรือเปลี่ยนสถานะตัวเองเป็นสถานะคอย การจัดเวลาซีพียูแบบนี้ เป็นวิธีการที่ใช้อยู่ในระบบปฏิบัติการวินโดวส์ และระบบปฏิบัติการแมคอินทอชโปรเซสใด ๆ ในระบบนี้สามารถครอบครองเวลาซีพียูได้จนกว่าจะหมดความต้องการ ถ้าระบบนี้ต้องการทำงานเป็นระบบหลายผู้ใช้ (Multi-user System) การจัดเวลาแบบให้สิทธิก่อนจะเหมาะสมกว่า

ดังนั้นการจัดเวลาซีพียูแบบไม่ให้สิทธิก่อน คือวิธีการจัดสรรซีพียูให้แก่โปรเซส ซึ่งโปรเซสนั้นสามารถใช้ซีพียูไปได้จนกระทั่ง เกิดการคอยอินพุต/เอาต์พุตหรือจนกระทั่งเสร็จงาน

4.5 ข้อพิจารณาในการจัดเวลา

อัลกอริทึมของการจัดเวลาซีพียู มีคุณสมบัติแตกต่างกันไป ซึ่งอาจจะมีการให้ความสำคัญกับโปรเซสบางอย่างมากกว่าโปรเซสชนิดอื่นๆ ดังนั้นการเลือกอัลกอริทึมสำหรับใช้ในสถานการณ์ต่างๆ นั้น เราจำเป็นที่จะต้องพิจารณาถึงคุณสมบัติเหล่านี้ให้ดีว่ามีข้อดี ข้อเสียอย่างไร

ข้อการพิจารณาหลายข้อ ได้ถูกกำหนดไว้เพื่อใช้ในการเปรียบเทียบอัลกอริทึมต่างๆ ซึ่งลักษณะข้อพิจารณาแต่ละชนิดนั้นสามารถสร้างความแตกต่างให้เห็นได้อย่างชัดเจนในการตัดสินว่าอัลกอริทึมได้ดีที่สุด ข้อการพิจารณาดังกล่าวมีดังนี้

4.5.1 อรรถประโยชน์ของซีพียู (CPU Utilization)

คือ คำนึงถึงการใช้ประโยชน์จากซีพียูอย่างสูงสุด โดยทำให้ซีพียูมีงานทำมากที่สุดเท่าที่จะทำได้ ซึ่งเมื่อคิดเป็นเปอร์เซ็นต์แล้วการใช้งานซีพียูสามารถวัดได้ออกมาในระหว่าง 0-100 เปอร์เซ็นต์ สำหรับในการทำงานจริง ซีพียูควรจะถูกใช้อยู่ระหว่าง 40-90 เปอร์เซ็นต์

4.5.2 จำนวนงานที่เสร็จต่อหน่วยเวลา (Throughput)

ถ้าหากว่าซีพียูทำงานตลอดเวลา ก็หมายความว่างานก็กำลังถูกทำให้เสร็จ ซึ่งสามารถจะวัดออกมาได้เป็นจำนวนงานที่เสร็จต่อหน่วยเวลา หรือเรียกว่า ทรูพุต (Throughput) สำหรับงานที่มีความยากมาก อัตราการทำงานหรือค่าของทรูพุต อาจจะเป็นหนึ่งงานต่อชั่วโมง และถ้าเป็นงานที่สั้นก็อาจจะทำให้ทรูพุตของระบบสามารถวัดได้ถึง 10 งานต่อวินาที

4.5.3 เวลาทั้งหมด (Turnaround Time)

คือช่วงเวลาทั้งหมดที่ใช้ในการทำงานใดงานหนึ่งตั้งแต่เริ่มต้นเข้าไปในระบบ จนงานถูกทำจนเสร็จเรียบร้อย ซึ่งจะรวมเอาเวลาที่จะต้องรอที่จะเข้าไปในหน่วยความจำ เวลาที่คอยอยู่ในคิว เวลาที่ใช้ซีพียู และเวลาของอินพุต/เอาต์พุตทั้งหมดเข้าด้วยกัน

4.5.4 เวลารอคอย (Waiting Time)

คือช่วงเวลาที่งานใดงานหนึ่งต้องรอการทำงานของตัวจัดเวลา โดยไม่รวมเวลาของการใช้ซีพียู และเวลาของการติดต่ออินพุต/เอาต์พุต ส่วนใหญ่ก็คือเวลาที่งานต้องคอยอยู่ในคิว (Ready Queue) นั่นเอง

4.5.5 เวลาตอบสนอง (Response Time)

คือเวลาที่วัดระหว่างเวลาที่มีการร้องขอการกระทำใดๆ ต่อระบบแล้วมีการตอบรับกลับออกมา ซึ่งในบางโปรแกรมที่มีการติดต่อกับผู้ใช้ภายนอกมากๆ เวลารวมทั้งหมดอาจจะไม่ใช่จุดประสงค์หลักที่ต้องการ เพราะเมื่อมีการติดต่อระหว่างโปรแกรมกับผู้ใช้ภายนอกนั้น ระบบปฏิบัติการจะสิ้นสุดการรับผิดชอบเมื่อการร้องขออินพุต/เอาต์พุตสิ้นสุดลง ต่อจากนั้นระบบปฏิบัติการก็จะไปจัดการในเรื่องภายในที่เกี่ยวข้องกับซีพียู อุปกรณ์อินพุต/เอาต์พุตต่างๆ ที่ติดต่อกับระบบเหล่านี้จะเป็นผู้ติดต่อกับผู้ใช้เอง ดังนั้นความเร็วของเวลาตอบสนองจึงมักจะขึ้นอยู่กับอุปกรณ์อินพุต/เอาต์พุตเหล่านี้ ตัวอย่างเช่น เครื่องพิมพ์ ซีพียูอาจทำการคำนวณใดๆ สิ้นสุดแล้วและได้มีการให้ผลลัพธ์ออกมา และซีพียูก็กำลังทำงานอื่นต่อไปแล้ว แต่ผลลัพธ์ที่ได้นี้ อาจจะต้องใช้เวลาอีกซักระยะหนึ่งกว่าที่ผู้ใช้จะมองเห็นผลลัพธ์ปรากฏบนกระดาษ อย่างนี้เป็นต้น

จากการวัดลักษณะต่างๆ ของระบบดังกล่าวข้างต้น สรุปได้ว่าสิ่งที่ผู้ออกแบบระบบการจัดการเวลาต้องการก็คือ การทำให้เวลาการทำงานสั้นที่สุดนั่นเอง แต่ในความเป็นจริงแล้ว เราไม่สามารถที่จะได้ระบบที่สามารถทำให้มีการใช้ซีพียูได้สูงสุด โดยที่มีทรูพุตมากที่สุด มีเวลารวมเร็วที่สุดและได้เวลาตอบสนองเร็วที่สุดได้พร้อมๆ กัน ดังนั้นการหาจุดพอดีจากค่าเฉลี่ยของคุณสมบัติแต่ละชนิดจึงมีความจำเป็น ซึ่งบางครั้งอาจจะต้องมีการกำหนดจุดที่ยอมรับได้ไว้ที่จุดใดจุดหนึ่ง ถ้าหากว่าไม่สามารถหาค่าเฉลี่ยได้

อย่างไรก็ตามได้มีการแนะแนวทางไว้ว่า สำหรับคอมพิวเตอร์ที่ทำงานแบบอินเตอร์แอ็กทีฟ (Interactive) ในระบบแบ่งเวลา (Time Sharing) การทำให้เวลาตอบสนองมีความคงที่แน่นอนสูงสุด มีความสำคัญกว่าการที่ทำให้ระบบมีทรูพุตสูงสุด เพราะว่าผู้ใช้อาจต้องการระบบที่มีเวลาในการตอบสนองที่แน่นอนสำหรับผู้ใช้หลายๆ คน อย่างไรก็ตามความแน่นอนและเชื่อถือได้ของการตอบสนองคำสั่งในระบบแบ่งเวลา ก็อาจเป็นเรื่องที่ไม่สำคัญมากนักในบางระบบของการทำงาน (นรีรัตน์ นิยมไทย, 2549)

4.6 อัลกอริทึมของการจัดเวลาซีพียู

อัลกอริทึมสำหรับการจัดเวลาโปรเซสนั้น มีความสำคัญอยู่ที่การตัดสินใจว่าจะให้โปรเซสใดครอบครองเวลาซีพียูก่อน

4.6.1 การจัดเวลาแบบมาก่อนได้ก่อน (FCFS : First Come First Served)

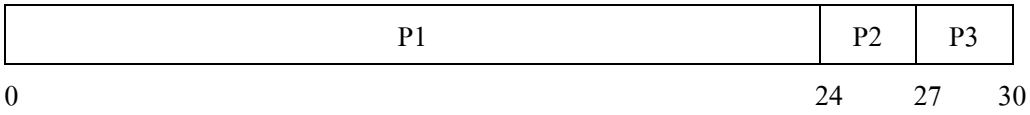
วิธีการในการจัดเวลาที่ง่ายที่สุดสำหรับการคัดเลือกโปรเซสให้ครอบครองเวลาซีพียู คืออัลกอริทึมที่ใช้แนวความคิดของการมาก่อนได้ก่อนซึ่งมีหลักการง่ายๆ คือ โปรเซสใดที่ร้องขอใช้ซีพียูก่อน ก็จะได้รับการจัดสรรให้ครอบครอบเวลาของซีพียูก่อน ซึ่งการสร้างอัลกอริทึมนี้ขึ้นมาทำได้ไม่ยาก เพราะสามารถนำเอาหลักการของคิวมาก่อนได้ก่อน (FIFO Queue) มาใช้ได้เลย เมื่อมีโปรเซสใดเข้ามาอยู่ในคิวแบบนี้ บล็อกควบคุมโปรเซสของงานก็จะถูกเชื่อมไว้กับหางของคิว เมื่อใดที่ซีพียูว่างลง โปรเซสใดที่มีบล็อกควบคุมโปรเซสอยู่ที่หัวของคิวก็จะถูกนำออกมาจากคิวให้เข้าครอบครองเวลาซีพียูได้เลย

ค่าเฉลี่ยของการคอยในคิวแบบมาก่อนได้ก่อนนี้ค่อนข้างจะสูง ซึ่งนี่ก็คือผลเสียของการใช้หลักการแบบนี้ ตัวอย่างที่จะแสดงต่อไปนี้ จะใช้อธิบายได้ว่า ทำไมมาก่อนได้ก่อน จึงทำให้ค่าเฉลี่ยของการคอยจึงมีค่าสูง

ตารางที่ 4.1 แสดงข้อมูลโปรเซสในตัวอย่างการจัดเวลาแบบมาก่อนได้ก่อน

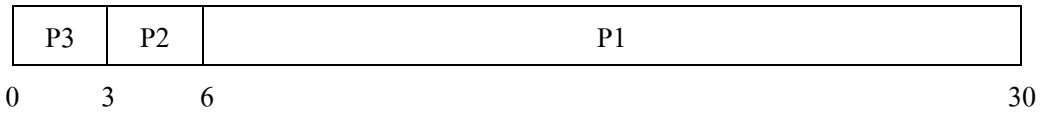
โปรเซส	ระยะเวลาความต้องการซีพียู (วินาที)
P1	24
P2	3
P3	3

สมมติว่างานมาถึงคิวตามลำดับดังนี้ คือ P1, P2 และ P3 แถวคอยในระบบมาก่อนได้ก่อนจะมีลักษณะดังภาพที่ 4.3



ภาพที่ 4.3 แสดงแถวคอยของโปรเซสในระบบมาก่อนได้ก่อน

เวลาการคอย P1 ในที่นี้จะมีค่าเป็น 0 วินาที และของ P2 คือ 24 วินาที และP3 คือ 27 วินาที ค่าเฉลี่ยของการคอยในสถานการณ์เช่นนี้ก็จะเท่ากับ (0+24+27)/3 หรือ 17 วินาที แต่ถ้าสมมติว่าโปรเซสเข้ามาในคิวตามลำดับใหม่ เช่น P3, P2 และ P1 คิวก็จะมีลักษณะที่ต่างออกไปดังภาพที่ 4.4



ภาพที่ 4.4 แสดงแถวคอยของโปรเซสในระบบมาก่อนได้ก่อนในคิวตามลำดับใหม่

จากสถานการณ์สมมติที่สองนี้ จะได้ว่า เวลาคิวของ P3 เท่ากับ 0 วินาที P2 เท่ากับ 6 วินาที และของ P1 เท่ากับ 6 วินาที ซึ่งจะทำให้ค่าเฉลี่ยของการคอยมีค่าเท่ากับ (0+3+6)/3 = 3 วินาทีเท่านั้น

จะเห็นได้ว่าถึงแม้จะมีโปรเซสเหมือนกันรอการเข้าครอบครองซีพียู แต่หากว่าลำดับการเข้าไปในคิวมีความสำคัญแตกต่างกันแล้ว ก็อาจทำให้ค่าเฉลี่ยของเวลาในการคอยมีความแตกต่างกันได้มาก นั่นก็คือเวลาของการคอยในแบบมาก่อนได้ก่อน นั้นยังคงไม่ดีพอ ถ้าหากว่าความต้องใช้ซีพียูของแต่ละโปรเซสมีคาบเวลาที่แตกต่างกันมาก ดังตัวอย่างข้างต้น

จากตัวอย่างที่ผ่านมาเป็นเพียงตัวอย่างของโปรเซสที่มีการร้องขอในซีพียูเพียงครั้งเดียว สมมติว่าโปรเซสที่เข้ามามีทั้งความต้องการใช้ซีพียู และอินพุต/เอาต์พุตสลับกันไป โปรเซสที่ต้องการใช้ซีพียูในระยะเวลาสั้นๆ จะต้องเสียเวลาคอยโปรเซสที่ต้องใช้ซีพียูเป็นระยะเวลานานๆ ครั้งแล้วครั้งเล่า ถ้ามีการสลับกันระหว่างเวลาซีพียูและเวลาอินพุต/เอาต์พุต 100 ครั้ง ค่าเฉลี่ยของเวลาการคอยก็จะเป็นค่าเฉลี่ยที่นานที่สุด ตามตัวอย่างลำดับของโปรเซส P1→P2→P3 ดังกล่าว

ลักษณะการทำงานของการจัดเวลาซีพียูแบบมาก่อนได้ก่อนนี้ เป็นอัลกอริทึมแบบไม่ให้สิทธิ์ก่อน นั่นก็คือเมื่อโปรเซสใดครอบครองเวลาซีพียูแล้ว ซีพียูจะไม่มีโอกาสได้ว่างจนกว่าความต้องการใช้ซีพียูของโปรเซสนั้นจะสิ้นสุดลงด้วยการสลับไปยังเวลาอินพุต/เอาต์พุต ซึ่งจะก่อให้เกิดปัญหาใหญ่ให้กับระบบคอมพิวเตอร์แบบแบ่งเวลา เพราะว่าผู้ใช้คนอื่นๆ อาจจะต้องคอยเวลาให้โปรเซสของตนเอง ซึ่งอาจเป็นโปรเซสสั้นๆ เสร็จลงพร้อมกับโปรเซสของผู้อื่นที่ใช้เวลายาวนานกว่ามากๆ

4.6.2 การจัดเวลาแบบงานสั้นทำก่อน (SJF : Short-Job-First Scheduling)

จากการที่ได้พบเห็นปัญหาในหลักการของอัลกอริทึมมาก่อนได้ก่อน ทำให้มีการคิดค้นต่อไปว่า ทำงานอย่างไรจึงจะเกิดความพอดีระหว่างโปรเซสที่ต้องการเวลาซีพียูสั้นๆ กับโปรเซสที่ต้องการเวลาซีพียูนานๆ ผลลัพธ์ก็คือ แนวความคิดที่จะทำให้โปรเซสที่ต้องการคาบเวลาของซีพียูในเวลาถัดไปสั้นที่สุด จะได้รับเลือกให้เข้ามาครอบครองเวลาซีพียูก่อน และถ้ามีโปรเซสหลายตัวที่

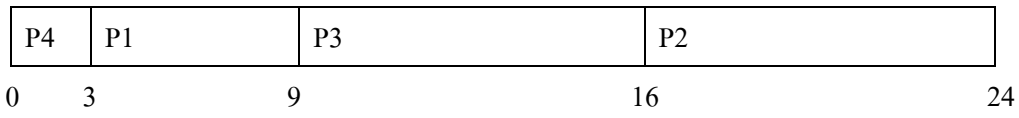
มีคาบเวลาของซีพียูของช่วงต่อไปเท่า ๆ กันก็จะใช้หลักการมาก่อนได้ก่อนในการคัดเลือก ซึ่งอาจมีชื่อเรียกการคัดงานแบบนี้ว่า สั้นที่สุดได้ใช้เวลาซีพียูต่อไป (Shortest next CPU burst) เพราะแนวความคิดนี้จะมีการจัดคาบเวลาของเวลาซีพียูช่วงต่อไปเพียงช่วงเดียวมากกว่าการวัดรวมว่าตลอดทั้งโปรแกรมต้องการใช้ซีพียูนานเท่าไหร่

ตัวอย่างการทำงานของการจัดเวลาแบบงานสั้นทำก่อน โดยสมมติข้อมูลโปรเซสที่มีเวลาซีพียูดังตารางที่ 4.2

ตารางที่ 4.2 แสดงข้อมูลโปรเซสในตัวอย่างการจัดเวลาแบบงานสั้นทำก่อน

โปรเซส	ระยะเวลาความต้องการซีพียู (วินาที)
P1	6
P2	8
P3	7
P4	3

ถึงแม้ว่าลำดับการเข้ามาในคิวอาจจะเป็น P1, P2, P3 และ P4 แต่การใช้หลักการของการจัดเวลาแบบงานสั้นทำก่อน จะจัดคิวออกมาได้ในลักษณะดังภาพที่ 4.5



ภาพที่ 4.5 แสดงแถวคอยของโปรเซสในระบบงานสั้นทำก่อน

ค่าเฉลี่ยของการคอยของงานในระบบก็จะเท่ากับ (0+3+9+16) / 4 = 7 วินาที ซึ่งหากว่าใช้หลักการของมาก่อนได้ก่อน ค่าเฉลี่ยของการคอยจะเท่ากับ (0+6+14+21) /4 = 10.25 จะเห็นได้ว่ามีความแตกต่างกันถึง 3.25 วินาที

จากการทดลองจะพบว่าการจัดเวลาแบบงานสั้นทำก่อน จะให้ค่าเฉลี่ยการคอยได้ต่ำที่สุดสำหรับชุดของโปรเซสใดๆ ก็ตาม เพราะการเลื่อนโปรเซสที่มีเวลาซีพียูน้อยสุดมาไว้หน้าคิว จะมี

การลดเวลาการคอยในโปรเซสสั้นมากกว่าการเพิ่มเวลาของโปรเซสที่ยาวเสมอ ดังนั้นค่าเฉลี่ยของการคอยแบบนี้จึงต่ำที่สุด

สิ่งที่ยากมากสำหรับการใช้วิธีการจัดเวลาแบบงานสั้นทำก่อน ก็คือการวัดค่าเวลาซีพียูถัดไปของแต่ละโปรเซสที่ร้องขอซีพียูเข้ามา ซึ่งในระบบการทำงานแบบแบตช์ (Batch System) เจ้าของโปรเซสจะเป็นผู้บอกช่วงเวลาที่จำกัดของการใช้ซีพียูสำหรับโปรเซสให้กับระบบ เพื่อใช้เป็นข้อมูลอ้างอิงสำหรับการจัดลำดับดังกล่าว ซึ่งเจ้าของโปรเซสก็จะพยายามที่จะให้ค่าต่ำที่สุด เพื่อโอกาสในการเข้าไปใช้ซีพียูก่อนมีมากที่สุดแต่ถ้าคาบเวลาที่กำหนดไว้นี้มีค่าน้อยเกินไป ก็จะทำให้โปรแกรมนั้นไม่สามารถคำนวณสำเร็จได้ ซึ่งจะต้องเสียเวลามาเริ่มต้นใหม่อีก การจัดเวลาแบบ SJF นั้น เป็นวิธีการที่ใช้กันมากในระบบแบตช์หรือการจัดเวลาช่วงยาว (Long-term Scheduling)

อย่างไรก็ตาม ถึงแม้ว่าการจัดเวลาแบบงานสั้นทำก่อนจะมีข้อได้เปรียบในแง่ของค่าเฉลี่ยของการคอย การจัดเวลาแบบงานสั้นทำก่อนก็ยังคงมีปัญหาในการนำมาใช้ในการจัดเวลาช่วงสั้น (Short-term Scheduling) เพราะว่าเราไม่มีทางจะรู้ได้ว่าความยาวของคาบเวลาการใช้ซีพียูในช่วงต่อไปของแต่ละโปรเซสเป็นเท่าใด วิธีการที่ใช้มักจะเป็นแค่การประมาณเอาเท่านั้น โดยประมาณเอาจากระยะเวลาของเวลาซีพียูในคาบเวลาที่ผ่านมาด้วยการหาค่าเฉลี่ย เป็นต้น

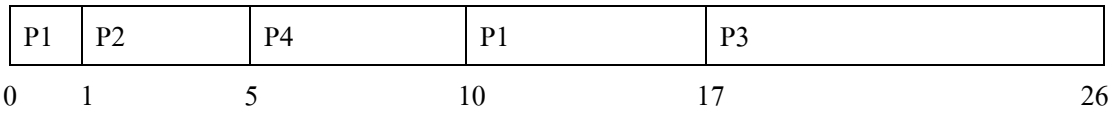
การทำงานของอัลกอริทึม การจัดเวลาแบบงานสั้นทำก่อนสามารถเป็นได้ทั้งแบบให้สิทธิ์ก่อน หรือไม่ให้สิทธิ์ก่อน การตัดสินใจเกิดขึ้นทุกครั้งเมื่อมีโปรเซสใหม่เข้ามาในคิว ในขณะที่ยังมีโปรเซสอื่นใช้งานซีพียูอยู่ เพราะว่าโปรเซสใหม่อาจมีคาบเวลาของเวลาซีพียูสั้นกว่าเวลาที่เหลืออยู่ของเวลาซีพียูของโปรเซสที่กำลังใช้ซีพียู ถ้าเป็นแบบนั้นอัลกอริทึมที่เป็นแบบให้สิทธิ์ก่อนจะทำการหยุดงานที่กำลังใช้ซีพียูอยู่ แต่ถ้าเป็นแบบไม่ให้สิทธิ์ก่อนอัลกอริทึมนี้ก็จะปล่อยให้งานที่ทำงานอยู่นั้นใช้ซีพียูจนเสร็จสิ้นเวลาซีพียูของช่วงเวลานั้น

ต่อไปนี้จะเป็นตัวอย่าง เพื่อความเข้าใจในเรื่องการจัดเวลาแบบงานสั้นทำก่อนแบบให้สิทธิ์ก่อน โดยสมมติให้มีโปรเซสที่จะเข้ามาในระบบ 4 โปรเซสดังตารางที่ 4.3

ตารางที่ 4.3 แสดงข้อมูลโปรเซสในตัวอย่างการจัดเวลาแบบงานสั้นทำก่อนแบบให้สิทธิ์ก่อน

โปรเซส	เวลาที่มาถึง	เวลาที่ใช้
P1	0	8
P2	1	4
P3	2	9
P4	3	5

เมื่องานมาถึงคิวตามเวลาที่แสดงไว้ ลำดับของการเข้าใช้ซีพียู จะเป็นดังภาพที่ 4.6



ภาพที่ 4.6 แสดงแถวคอยของโปรเซสในระบบงานสั้นทำก่อนแบบให้สิทธิ์ก่อน

อธิบายได้ว่า เมื่อโปรเซส P1 เข้ามาเมื่อเวลา 0 วินาที โปรเซส P 1 ก็ได้เข้าไปที่ซีพียูทันที เนื่องจากว่ายังไม่มีโปรเซสอื่นเข้ามา เมื่อถึงเวลาที่ 1 วินาที งาน P2 ซึ่งมีคาบเวลาซีพียูสั้นกว่าคือ 4 วินาที เมื่อเทียบกับ P1 ซึ่งยังเหลืออีก 7 วินาทีดังนั้นงาน P1 ยังเหลือมากกว่า P4 แต่น้อยกว่า P3 การจัดลำดับการเข้าใช้ซีพียู จึงออกมาตามภาพที่ 4.6 โดยถ้าให้ P2 แทรก P1 กลางคัน การจัดลำดับการเข้าใช้ซีพียู จึงออกมาดังภาพที่ 4.7 ซึ่งจะมีเวลาคอยเฉลี่ยจะเท่ากับ ((10-1)+(1-1)+(17-2)+(5-3))/4 = 26/4 =6.5 วินาที แต่ถ้าห้ามแทรกกลางคัน เวลารอคอยเฉลี่ยจะเท่ากับ ((0)+(8-1)+(17-2)+(12-3))/4 = 31/4 =7.75 วินาที

ภาพที่ 4.7 แสดงแถวคอยของโปรเซสในระบบงานสั้นทำก่อนแบบให้สิทธิ์ก่อนเมื่อมีการแทรกคิว

4.6.3 การจัดเวลาตามลำดับความสำคัญ (Priority Scheduling)

วิธีการของอัลกอริทึมการจัดเวลาแบบงานสั้นทำก่อน สามารถมองได้อีกแง่หนึ่งว่าเป็นการจัดเวลาตามลำดับความสำคัญชนิดหนึ่งลำดับความสำคัญของแต่ละโปรเซสจะถูกกำหนดไว้ด้วยวิธีใดวิธีหนึ่ง เพื่อว่าการซีพียูจะมีการใช้กับโปรเซสที่มีลำดับความสำคัญสูงที่สุดเท่านั้น แต่ถ้ามีงานที่มีลำดับความสำคัญเท่ากัน ก็จะมีการนำแบบมาก่อนได้ก่อนมาใช้ ในที่นี้วิธีการของการจัดเวลาแบบงานสั้นทำก่อน ได้มีการกำหนดลำดับความสำคัญของโปรเซสด้วยคาบระยะเวลาของความต้องการใช้ซีพียู หรือเวลาซีพียูของแต่ละโปรเซส เช่นโปรเซสใดมีคาบเวลาสั้นที่สุดก็จะมีลำดับความสำคัญมากที่สุด ทีนี้เมื่อเราพูดถึงความมากน้อยของลำดับความสำคัญ ก็มีความจำเป็นที่จะต้องกำกับไว้ด้วยสัญลักษณ์ที่เปรียบเทียบกันได้เช่น ตัวเลขจำนวนนับ เป็นต้น แต่จะกำหนดให้มีกี่ระดับนั้นก็สุดแล้วแต่ผู้ออกแบบ อาจเป็น 0-9 หรือ 0-6789 ก็ได้และก็ไม่มีกฎตายตัวว่าเลขมากหรือเลขน้อยจะมี

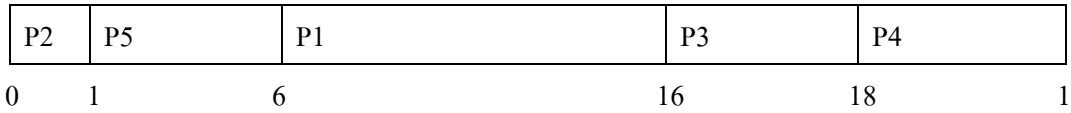
ความหมายในเรื่องของลำดับความสำคัญอย่างไร ซึ่งการไม่มีข้อตกลงในการใช้ระบบการกำกับของความสำคัญเหล่านี้ อาจก่อให้เกิดความสับสนขึ้นได้ดังนั้นในหนังสือเล่มนี้จะมีการกำหนดว่าตัวเลขน้อยมีลำดับความสำคัญมาก เพื่อเป็นที่เข้าใจร่วมกัน

ตัวอย่างต่อไปนี้เป็นการแสดงถึงการจัดเวลาซีพียู แบบใช้ตัวเลขเป็นตัวบอกลำดับความสำคัญ โดยมีตัวอย่างโปรเซส 5 ตัวอย่างดังตารางที่ 4.4

ตารางที่ 4.4 แสดงข้อมูลโปรเซสในตัวอย่างการจัดเวลาตามลำดับความสำคัญ

โปรเซส	เวลาที่ใช้	ลำดับความสำคัญ
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

หากใช้การจัดเวลาแบบการใช้ลำดับความสำคัญเราจะสามารถจัดลำดับของโปรเซสให้เข้าไปทำงานซีพียูได้ดังภาพที่ 4.8



ภาพที่ 4.8 แสดงแถวคอยของโปรเซสในระบบการจัดเวลาตามลำดับความสำคัญ

ซึ่งค่าเฉลี่ยของการคอยในคิวของแต่ละโปรเซสมีค่าเท่ากับ (1+6+16+18)/5 = 8.2 วินาที การกำหนดลำดับความสำคัญของแต่ละโปรเซสสามารถกำหนดได้ทั้งภายใน และภายนอก การกำหนดภายในคือ การกำหนดลำดับความด้วยวิธีการใดวิธีการหนึ่งจากการวัดและคำนวณคุณสมบัติบางอย่างของแต่ละโปรเซสที่สามารถวัดค่าได้ คุณสมบัติที่วัดค่าได้เหล่านี้เช่น เวลาของการจำกัดการใช้ซีพียู ความต้องการขนาดของหน่วยความจำ จำนวนไฟล์ที่ต้องเกี่ยวข้องหรือแม้แต่อัตราส่วนของจำนวนเวลาอินพุต/เอาต์พุตต่อเวลาซีพียู เป็นต้น ส่วนการกำหนดจากภายนอกหมายถึง การกำหนดลำดับความสำคัญของโปรเซสใดๆ จากวิธีการอื่นนอกเหนือจาก

ระบบปฏิบัติการ ตัวอย่างเช่น ราคาค่าเช่าเวลาซีพียู แหล่งที่มาของโปรเซส หรือความเร่งด่วนของโปรเซส เหล่านี้ล้วนสามารถนำมาเป็นตัวที่ใช้สำหรับคำนวณค่าของลำดับความสำคัญของโปรเซสก่อนที่ระบบปฏิบัติการจะรับเอาโปรเซสเข้ามานั่นเอง

การจัดเวลาแบบมีลำดับความสำคัญ สามารถเป็นได้ทั้งแบบให้สิทธิ์ก่อนหรือไม่ให้สิทธิ์ก่อนก็ได้ โดยมีลักษณะการทำงานเหมือนกับการจัดเวลาแบบงานสั้นทำก่อน ดังกล่าวมาแล้วทุกประการ ปัญหาใหญ่ของการจัดเวลาซีพียู โดยใช้ลำดับความสำคัญของโปรเซสมาเกี่ยวข้องก็คือ โปรเซสที่มีลำดับความสำคัญต่ำอาจจะไม่มีโอกาสได้เข้าไปใช้ซีพียูเลย ถ้าหากว่ามีโปรเซสที่มีลำดับความสำคัญสูงกว่าอยู่เป็นจำนวนมาก ซึ่งโดยปกติแล้วโปรเซสที่มีลำดับความสำคัญต่ำเหล่านี้กว่าจะได้รับซีพียูมาทำงาน ก็อาจจะต้องรอเป็นเวลานานหรือไม่ก็ต้องถูกยกเลิกไปในที่สุดเพราะระบบคอมพิวเตอร์มีการติดขัดและต้องเริ่มเปิดเครื่องกันใหม่

การแก้ปัญหาให้กับสถานการณ์ที่มีโปรเซสหลงเหลือ เนื่องจากมีลำดับความสำคัญต่ำก็คือการเพิ่มลำดับความสำคัญให้กับโปรเซสที่ยังไม่เสร็จเหล่านี้ ตามระยะเวลาที่คอยอยู่ในคิว ตัวอย่างเช่น สมมติว่าเรามีการออกแบบให้มีลำดับความสำคัญจาก 0-127 ขั้น เราอาจจะเพิ่มอัลกอริทึมพิเศษลงไปว่า ถ้าโปรเซสใดคอยการใช้ซีพียูครบ 15 นาที ก็ให้ลดตัวเลขลำดับขึ้นลงทีละขั้น และจะลดลงไปเรื่อยๆ ทุกๆ 15 นาที ซึ่งการทำแบบนี้ แม้โปรเซสที่เข้ามาในระบบมีลำดับความสำคัญต่ำสุดที่ 127 ก็จะมีโอกาสเข้าไปใช้ซีพียูภายในเวลาไม่เกิน 32 ชั่วโมง เพราะในที่สุดโปรเซสนี้ก็จะมีลำดับความสำคัญเท่ากับ 0

4.6.4 การจัดเวลาแบบวนรอบ (RR : Round-Robin Scheduling)

การจัดเวลาแบบวนรอบ เป็นวิธีการที่คิดขึ้นมาเพื่อใช้กับระบบคอมพิวเตอร์แบบแบ่งเวลาโดยเฉพาะ โดยมีลักษณะการทำงานแบบมาก่อนได้ก่อน แต่ให้มีกรรมวิธีของสิทธิ์ก่อนรวมอยู่ด้วย แต่ละโปรเซสที่เข้ามาในระบบจะถูกจำกัดเวลาการเข้าไปใช้ซีพียูเท่าๆ กัน ซึ่งช่วงเวลานี้จะเป็นช่วงเวลาสั้นๆ เรียกว่า เวลาควอนตัม (Quantum Time) หรือช่วงเวลา (Time Slice) ระยะเวลาควอนตัมนี้ มีความยาวระหว่าง 10 ถึง 100 มิลลิวินาที และคิวที่ใช้ก็เป็นแบบวงกลม (Circular Queue) ตัวจัดเวลาจะมีการให้ซีพียูกับโปรเซสที่อยู่ในคิวแบบวนไปรอบๆ ในแต่ละคาบเวลาที่ให้นั้น โดยจะมีความยาวนานของการได้รับซีพียูมากที่สุดคือ 1 ควอนตัม ถ้าโปรเซสใดไม่สามารถกระทำได้สำเร็จภายใน 1 ควอนตัมนี้ โปรเซสก็จะต้องถูกนำกลับไปไว้ในคิวเช่นเดิม สถานภาพต่างๆ ของโปรเซสที่ยังทำไม่เสร็จก็จะถูกบันทึกไว้ เพื่อว่าเมื่อถึงโอกาสได้ครอบครองซีพียูอีก ก็จะได้เริ่มต้นรันต่อจากครั้งที่แล้วโดยไม่ต้องเริ่มใหม่ทั้งหมด

การสร้างระบบการทำงานแบบวนรอบ เราจะทำคิวที่พร้อมทำงาน (Ready Queue) เป็นแบบมาก่อนได้ก่อนไว้สำหรับเก็บโปรเซสต่างๆ โปรเซสที่เข้ามาใหม่จะถูกนำมาต่อไว้ที่หางของคิว ตัวจัดเวลาจะเลือกเอาโปรเซสที่อยู่ตรงหัวคิวออกมา แล้วกำหนดให้ไทม์เมอร์หยุดการให้เวลาซีพียูหลังจากนั้น 1 ควอนตัม แล้วนำโปรเซสออกไปต่อที่หางคิว ถ้าหากว่าโปรแกรมยังไม่สิ้นสุดการทำงาน

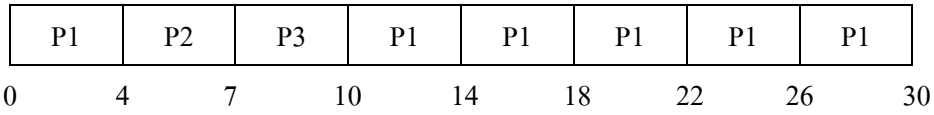
โปรเซสบางโปรเซสอาจต้องการใช้ซีพียูน้อยกว่า 1 ควอนตัม ในกรณีนี้โปรเซสที่เสร็จก่อนถึงเวลา 1 ควอนตัม จะต้องให้ออกจากการครอบครองซีพียู เพื่อโปรเซสอื่นที่อยู่ตรงหัวคิวเข้ามาทำงานได้

เวลาเฉลี่ยของการคอยในกรรมวิธีของวนรอบจะค่อนข้างนาน ให้ลองพิจารณาตัวอย่างการประมวลผล 3 โปรเซสดังต่อไปนี้ในแบบวนรอบ โดยที่โปรเซสทั้ง 3 เข้ามาถึงระบบพร้อมๆ กัน

ตารางที่ 4.5 แสดงข้อมูลโปรเซสในตัวอย่างการจัดเวลาแบบวนรอบ

โปรเซส	เวลาที่ใช้ไป
P1	24
P2	3
P3	3

หากกำหนดให้เวลาควอนตัมเท่ากับ 4 วินาที โปรเซส P1 ครอบครองซีพียูในครั้งแรก 4 วินาที แต่เวลาที่ต้องการจริงคือ 24 วินาที ดังนั้นโปรเซส P1 จึงเหลือเวลาที่ต้องการอีก 20 วินาที แต่เมื่อเวลาควอนตัมแรกหมดลงที่วินาทีที่ 4 โปรเซส P1 ก็จะออกจากการครองครองซีพียู โปรเซส P2 ซึ่งเป็นโปรเซสที่อยู่ถัดไปในคิว ก็จะได้ครอบครองซีพียู แต่โปรเซส P2 ต้องการครองครองซีพียูแค่ 3 วินาที ดังนั้นโปรเซส P2 จึงทำเสร็จก่อนที่เวลาของควอนตัมจะหมดลง โปรเซส P3 ที่คอยอยู่ถัดไปจึงได้เข้าครอบครองซีพียู ที่เวลา 7 วินาที ผลของการทำงานแบบควอนตัมต่อโปรเซสดังกล่าวสามารถแสดงได้ดังภาพที่ 4.9 และมีค่าเฉลี่ยของเวลาที่ต้องคอยคือ 17/3 = 5.66 วินาที



ภาพที่ 4.9 แสดงแถวคอยของโปรเซสในระบบการจัดเวลาแบบวนรอบ

ในหลักการทำงานของการวนรอบจะไม่มีโปรเซสใดๆ ที่จะได้รับโอกาสในการครอบครองซีพียูเกินหนึ่งควอนตัม ไม่ว่าโปรเซสที่กำลังใช้งานอยู่นั้นจะเสร็จหรือไม่ก็ตาม เมื่อหมดเวลาของหนึ่งควอนตัม โปรเซสที่กำลังครอบครองซีพียูอยู่นั้นก็จะต้องถูกนำออกมาจากซีพียู และถ้ายังไม่เสร็จก็จะถูกนำเข้าไปในคิว การทำงานแบบวนรอบจึงเป็นการทำงานแบบให้สิทธิ์ก่อนถ้ามีโปรเซสอยู่ในคิวจำนวน n โปรเซส และระยะเวลาของควอนตัมเท่ากับ q หน่วย แต่ละโปรเซสที่อยู่ในคิวจะมีเวลาเฉลี่ยของการคอยไม่นานไปกว่า (n-1) x q หน่วย ก่อนที่จะได้รับการเข้าไปใช้ซีพียูอีกครั้ง ตัวอย่าง เช่น ถ้ามีโปรเซส 5 โปรเซส และระยะของควอนตัมคือ 20 วินาทีแต่ละโปรเซสจะต้องคอยในคิวโดยเฉลี่ยประมาณไม่เกิน 80 วินาที ที่บอกว่าไม่เกินก็เพราะว่าการคอยอาจจะน้อยกว่านี้ ถ้าหากว่ามีโปรเซสใดๆ สามารถทำงานเสร็จโดยใช้เวลาน้อยกว่าเวลาควอนตัมนั่นเอง

ประสิทธิภาพของการวนรอบขึ้นอยู่กับการกำหนดขนาดของควอนตัมเป็นอย่างยิ่ง ถ้าขนาดของควอนตัมใหญ่เกินไป ประสิทธิภาพการวนรอบก็จะใกล้เคียงกับแบบมาก่อนได้ก่อน แต่ถ้าขนาดของควอนตัมเล็กมากเกินไป ทรูพุต (Throughput) ของระบบก็จะช้าลง เนื่องจากการนำเอาโปรเซสเข้าและออกจากการครอบครองซีพียูจะต้องเสียเวลาบางส่วนในการทำ Dispatcher ซึ่งถ้าขนาดของควอนตัมเล็กใกล้เคียงกับเวลาของ Dispatcher เวลาของระบบรวมก็จะหมดไปกับการเอาโปรเซสเข้าและออก (Context Switch) นั่นเอง

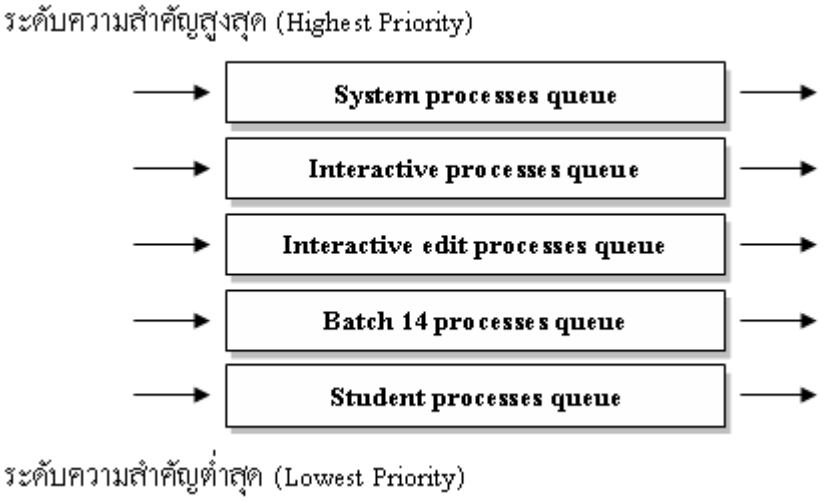
ในการทำระบบปฏิบัติการแบบการวนรอบนั้น การกำหนดขนาดของควอนตัม เป็นเรื่องที่ต้องทำกันอย่างพิถีพิถัน ซึ่งอาจจะต้องมีการทดลองใช้ขนาดของควอนตัมหลายๆ ขนาดกับระบบที่จะใช้จริงเนื่องจากความเร็วของซีพียูและอุปกรณ์ต่างๆ ที่เกี่ยวข้องกับการทำคอนเท็กซ์สวิตช์มีความเร็วไม่เท่ากันในแต่ละคอมพิวเตอร์ ส่วนใหญ่แล้วขนาดของควอนตัมมักจะขนาดใหญ่กว่าเวลาของคอนเท็กซ์สวิตช์พอสมควร เช่น ประมาณ 10 เท่า เป็นต้น ซึ่งหมายความว่าถ้าเวลาของคอนเท็กซ์สวิตช์ที่ต้องใช้คือ 1 วินาที ขนาดของควอนตัมควรจะเป็นประมาณ 10 วินาทีเป็นต้น แต่นั่นไม่ใช่เป็นสูตรตายตัวสำหรับระบบงานคอมพิวเตอร์ทุก ๆ เครื่อง การกำหนดขนาดที่เหมาะสมยังต้องคำนึงถึงลักษณะของโปรเซส ความเร็วของซีพียู และความต้องการในลักษณะการตอบสนองของคอมพิวเตอร์ในสภาวการณ์ต่างๆ ประกอบกันด้วย

เวลาในการวนรอบ (Turnaround time) ก็เป็นอีกส่วนหนึ่งที่จะมีผลกระทบจากขนาดของควอนตัม เวลาในการวนรอบไม่จำเป็นที่จะต้องดีขึ้นจากการที่ขนาดของควอนตัมที่เพิ่มขึ้นเสมอไป ในทางปฏิบัติทั่ว ๆ ไปแล้ว ค่าเฉลี่ยของเวลาในการวนรอบจะดีขึ้นถ้างานแต่ละงานสามารถทำเสร็จได้ภายในระยะเวลาของ 1 ควอนตัม

4.6.5 การจัดเวลาแบบคิวหลายระดับ (Multilevel Queue Scheduling)

เป็นการจัดเวลาของการนำโปรเซสเข้ามาครอบครองซีพียูอีกแบบหนึ่ง สำหรับระบบที่สามารถแบ่งระดับชั้นของงานได้อย่างชัดเจน เช่น งานที่เป็นฟอร์กราวนด์ (Foreground) หรืออินเตอร์แอ็กทีฟ (Interactive) กับงานที่เป็นแบ็คกราวนด์ (Background) หรือแบ็ตช์ (Batch) ซึ่งงานทั้งสองแบบนี้ต้องการเวลาตอบสนอง (Response time) ที่แตกต่างกัน ซึ่งสามารถใช้ระบบการจัดเวลาที่แตกต่างกันได้ โดยทั่วไปแล้วงานที่เป็นฟอร์กราวนด์จะมีระดับชั้นความสำคัญเหนือกว่างานที่เป็นแบ็คกราวนด์ การจัดเวลาแบบคิวหลายระดับนี้จะใช้วิธีแบ่งคิวออกเป็นหลายๆ ระดับโดยที่แต่ละระดับหมายถึงระดับโปรเซสที่มีความสำคัญแตกต่างกัน ซึ่งการแบ่งระดับความสำคัญของโปรเซสนั้น สามารถแบ่งได้หลายลักษณะไม่ว่าจะแบ่งตามขนาดของโปรเซส จำนวนหน่วยความจำที่ต้องใช้หรือจำนวนอินพุต/เอาต์พุต เป็นต้น โดยที่แต่ละคิว ยังสามารถใช้หลักการของการจัดเวลาที่แตกต่างกันได้ด้วย เช่น งานที่เป็นฟอร์กราวนด์ก็อาจใช้การจัดตารางแบบการวนรอบ ส่วนงานที่เป็นแบบแบ็คกราวนด์ก็อาจใช้วิธี การจัดตารางเวลาแบบมาก่อนได้ก่อน (FCFS) นอกจากนี้ยังมีการเพิ่มการจัดเวลาระหว่างคิวอีกด้วย ซึ่งอาจเป็นได้ทั้งแบบการจัดลำดับความสำคัญแบบคงที่ (Fixed-priority preemptive scheduling) หรือ การจัดลำดับความสำคัญแบบเปลี่ยนแปลงได้ (Variable-priority preemptive scheduling) ก็ได้ ซึ่งในแบบหลังนี้โปรเซสอาจมีการปรับตัวเองในเรื่องของระดับความสำคัญด้วยการเลื่อนไปมาระหว่างคิวได้

ตัวอย่างลักษณะของการจัดลำดับความสำคัญแบบคงที่ที่มีคิว 5 คิวดังภาพที่ 4.10



ภาพที่ 4.10 การจัดเวลาแบบคิวหลายระดับ

ที่มา (http://csnon04.blogspot.com/2008/03/3_06.html, 2552)

แต่ละคิวจะมีความสำคัญเหนือกว่าคิวที่อยู่ด้านล่างถัดลงไปเสมอ นั่นก็คือโปรเซสที่คอยอยู่ในคิวที่มีความสำคัญต่ำจะมีโอกาสได้ออกมาใช้ซีพียูก็ต่อเมื่อคิวที่มีความสำคัญสูงกว่าไม่มีโปรเซสที่ต้องทำเหลืออยู่เท่านั้น หรือทำในขณะที่โปรเซสที่มีลำดับความสำคัญกำลังครอบครองซีพียู แล้วมีโปรเซสที่มีลำดับความสำคัญสูงกว่าเข้ามาคอยอยู่ในคิวที่สูงกว่า โปรเซสนี้ก็จะถูกนำออกมาจากซีพียูทันที

อย่างไรก็ตามเพื่อป้องกันไม่ให้โปรเซสที่อยู่ในคิวต่ำ ต้องคอยอยู่นานเกินไป หรืออาจจะไม่มีโอกาสได้เข้าไปใช้ซีพียูเลย เพราะว่าในคิวบนๆ ไม่เคยว่างเลย ก็อาจจะต้องมีการกำหนดสัดส่วนเวลาให้กับแต่ละคิวในการเข้าไปใช้ซีพียู เช่น การกำหนดให้เวลา 80 เปอร์เซ็นต์เป็นของโปรเซสที่เป็นฟอร์กราวนด์ และอีก 20 เปอร์เซ็นต์เป็นของงานแบ็คกราวนด์ เป็นต้น

ส่วนการทำงานแบบการจัดลำดับความสำคัญแบบเปลี่ยนแปลงได้นั้น อาจมีชื่อเรียกอีกอย่างได้ว่าเป็นการทำงานแบบ Multilevel Feedback Queue Scheduling เพราะว่าโปรเซสในแต่ละคิวสามารถมีการเลื่อนชั้นระดับความสำคัญขึ้นลงได้ ซึ่งเมื่อเปรียบเทียบกับแบบการจัดลำดับความสำคัญแบบการจัดลำดับความสำคัญคงที่แล้วอาจจะต้องเสียเวลาเพิ่มอีกนิดหน่อยในการคำนวณหาระดับความสำคัญใหม่ให้กับโปรเซส ซึ่งส่วนมากแล้วจะแบ่งระดับความสำคัญตามระยะเวลาของเวลาซีพียู เช่น ทำงานได้ดี เวลาซีพียูนานขึ้นก็อาจจะถูกลดชั้นลงมาสู่คิวที่มีลำดับความสำคัญต่ำได้ ซึ่งจะทำให้โปรเซสที่มีการใช้ซีพียูน้อยแต่มีอินพุต/เอาต์พุต หรืออินเตอร์แอ็กทีฟมากๆ มีโอกาสเข้าไปอยู่ในคิวที่มีความสำคัญมากได้ วิธีนี้ยังเป็นการป้องกันไม่ให้โปรเซสที่มีความสำคัญน้อยถูกดองอยู่ในคิว เพราะโปรเซสที่อยู่ในคิวที่ต่ำก็สามารถเลื่อนขึ้นไปสู่คิวที่สูงขึ้นถ้าคอยอยู่นานเกินไป

พิจารณาตัวอย่างของ Multilevel Feedback Queue ดังภาพที่ 4.7

ภาพที่ 4.11 Multilevel Feedback Queues

ที่มา (http://csnon04.blogspot.com/2008/03/3_06.html, 2552)

เมื่อโปรเซสแรกเข้ามาในคิวแรก ก็จะได้โอกาสเข้าไปครอบครองซีพียู ภายใต้การกำหนดขนาดของควอนตัม 8 วินาที ซึ่งถ้าไม่สามารถเสร็จได้ในเวลาควอนตัมที่กำหนดนี้ ก็จะถูกลดชั้นไปอยู่ในคิวที่สองที่มีการกำหนดขนาดควอนตัมไว้ 16 วินาที และถ้าเมื่อไหร่คิวแรกไม่มีโปรเซสอยู่ โปรเซสที่อยู่ในคิวที่มีการกำหนดขนาดควอนตัมไว้ 16 วินาที ก็จะมีโอกาสเข้าไปครอบครองซีพียู และถ้าโปรเซสนี้ยังไม่เสร็จใน 16 วินาที จะถูกลดชั้นลงมาอยู่ในคิวแบบมาก่อนได้ก่อน ซึ่งจะได้โอกาสครอบครองซีพียูก็ต่อเมื่อคิวควอนตัมขนาด 8 วินาทีและ 16 วินาทีว่างเท่านั้น อย่างไรก็ตาม ถ้ามีโปรเซสที่ถูกแช่อยู่ในคิวมาก่อนได้ก่อนนานเกินไป โปรเซสเหล่านี้ก็อาจจะมีโอกาสที่จะถูกนำกลับขึ้นไปยังคิวด้านบนได้เช่นกัน

หลักการทำงานแบบนี้จะให้ความสำคัญกับโปรเซสใด ๆ ก็ตามที่มีเวลาซีพียูน้อยกว่าหรือเท่ากับ 8 วินาที ส่วนงานที่มีเวลาซีพียูมากกว่า 8 วินาที แต่น้อยกว่าหรือเท่ากับ 24 วินาที ก็จะตกลงมาอยู่ในคิวมาก่อนได้ก่อน

สิ่งสำคัญที่ต้องคำนึงถึงเมื่อต้องการที่จะออกแบบการจัดเวลาแบบนี้ มีดังนี้

1. จำนวนของคิว
2. วิธีของการจัดเวลาของแต่ละคิว
3. หลักเกณฑ์ในการตัดสินใจเพิ่มความสำคัญของโปรเซส
4. หลักเกณฑ์ในการตัดสินใจลดความสำคัญของโปรเซส
5. หลักเกณฑ์ในการตัดสินใจนำเอาโปรเซสที่ต้องการครอบครองซีพียูมาเข้าในคิว

จากการทำงานของ Multilevel Feedback Queue ดังกล่าว ทำให้เป็นระบบการจัดเวลาที่ใช้กันอย่างแพร่หลาย เพราะมันสามารถปรับเปลี่ยนส่วนต่างๆ ให้เหมาะสมได้ง่ายกับระบบคอมพิวเตอร์ที่จะต้องควบคุม แต่การที่มันมีค่าที่ต้องปรับเปลี่ยนเป็นจำนวนมากตามที่กล่าวมาทั้ง 5 ข้อข้างต้นนั้น ก็ทำให้การตัดสินใจหาค่าที่เหมาะสมเป็นสิ่งที่กระทำได้ไม่ง่ายนัก เนื่องจากเป็นการปรับแต่งที่ซับซ้อนมาก (พิรพร หมุนสนิท และคนอื่น ๆ, 2553)

4.7 การคัดเลือกอัลกอริทึมสำหรับการจัดเวลาซีพียู

หลักเกณฑ์ในการเปรียบเทียบอัลกอริทึมต่างๆ มักจะมาจากอรรถประโยชน์ของซีพียู (CPU utilization) เวลาตอบสนอง (Response time) และทรูพุต (Throughput) ซึ่งต้องมีการให้ค่าความสำคัญของแต่ละคุณสมบัติไว้

หลักเกณฑ์การพิจารณาอาจกำหนดได้เป็น ดังตัวอย่างดังนี้

1. ให้มีการใช้ซีพียูสูงสุด โดยให้สามารถมีช่วงเวลาตอบสนองที่ไม่นานไปกว่า 1 วินาที

2. ให้มีทรูพุตสูงสุด โดยที่เวลาวนรอบเป็นอัตราส่วนโดยตรงอย่างพอเหมาะกับเวลาที่ต้องใช้ในการรันทั้งหมด

ในการคัดเลือกอัลกอริทึมสำหรับการจัดเวลาซีพียูนั้น มีอยู่ด้วยกันหลากหลายวิธี ซึ่งแต่ละระบบปฏิบัติการที่ถูกพัฒนาขึ้นมาได้คัดเลือกอัลกอริทึมให้เหมาะสมกับตัวเอง วิธีที่ใช้คัดเลือกอัลกอริทึมสำหรับการจัดเวลาซีพียู มีดังนี้

4.7.1 Deterministic Modeling

วิธีนี้เป็นวิธีการคัดเลือกที่เรียกว่า Analytic Evaluation ซึ่งจะนำเอาอัลกอริทึมชนิดต่าง ๆ และลักษณะของงานมาสร้างสูตร เพื่อใช้ในการคำนวณหาตัวเลขของประสิทธิภาพที่สามารถวัดและเปรียบเทียบได้

ตัวอย่างต่อไปนี้ สมมติว่ามีงาน 5 โปรเซสนี้มาถึงระบบที่เวลา 0 วินาทีตามลำดับที่ให้ไว้ในตารางโดยมีระยะเวลาของซีพียูตามกำหนด

ตารางที่ 4.6 แสดงข้อมูลโปรเซสในตัวอย่างวิธีการคัดเลือกอัลกอริทึมสำหรับการจัดเวลาซีพียู

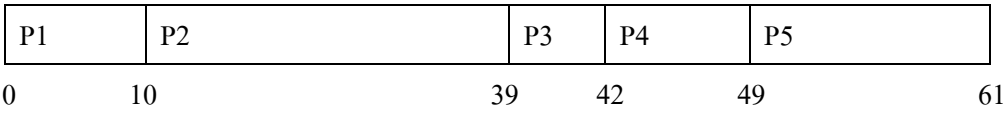
โปรเซส	เวลา
P1	10
P2	29
P3	3
P4	7
P5	12

อัลกอริทึมที่จะนำมาพิจารณา สมมติว่ามี 3 ชนิดคือ มาก่อนได้ก่อน (FCFS) งานสั้นได้ทำก่อน (SJF) และเวลาวนรอบ (RR) โดยมีควอนตัม เท่ากับ 10 วินาที ถ้าจะพิจารณาว่าอัลกอริทึมชนิดใดที่จะให้ค่าเฉลี่ยของการคอยต่ำสุด จะมีวิธีการคำนวณดังนี้

สำหรับการจัดเวลาแบบมาก่อนได้ก่อน

การจัดลำดับของโปรเซสให้เข้าไปทำงานซีพียูได้ดังภาพที่ 4.12

เวลาการคอยของการจัดเวลาแบบมาก่อนได้ก่อน คือ (0+10+39+42+49)/5 = 28 วินาที

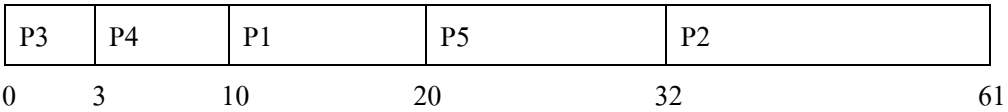


ภาพที่ 4.12 คิวการจัดเวลาแบบมาก่อนได้ก่อน การคัดเลือกอัลกอริทึมสำหรับการจัดเวลาซีพียู

สำหรับการจัดเวลาแบบงานสั้นได้ทำก่อน

จัดลำดับของโปรเซสให้เข้าไปทำงานซีพียูได้ดังภาพที่ 4.13

เวลาการคอยของการจัดเวลาแบบงานสั้นได้ทำก่อน คือ (10+32+0+3+20)/5 = 13 วินาที

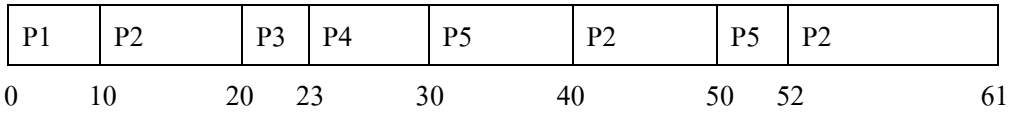


ภาพที่ 4.13 คิวการจัดเวลาแบบงานสั้นได้ทำก่อน การคัดเลือกอัลกอริทึมสำหรับการจัดเวลาซีพียู

สำหรับการจัดเวลาแบบวนรอบ

การจัดลำดับของโปรเซสให้เข้าไปทำงานซีพียูได้ดังภาพที่ 4.14

เวลาการคอยของ FCFS คือ (0+32+20+23+40)/5 = 23 วินาที



ภาพที่ 4.14 คิวการจัดเวลาแบบวนรอบ การคัดเลือกอัลกอริทึมสำหรับการจัดเวลาซีพียู

จากตัวอย่างจะเห็นได้ว่า หากใช้เกณฑ์ตัดสินจากค่าเฉลี่ยการคอยของการจัดเวลาแบบงานสั้นได้ทำก่อน จะเป็นอัลกอริทึมที่ดีที่สุด เนื่องจากว่ามีค่าเฉลี่ยของการคอยต่ำที่สุด

วิธี Deterministic Modeling เป็นวิธีที่ง่ายและรวดเร็ว ได้ผลลัพธ์ที่เป็นตัวเลขแน่นอน เพื่อการเปรียบเทียบที่สามารถเห็นได้อย่างชัดเจนว่าอัลกอริทึมแบบไหนดีที่สุด อย่างไรก็ตาม ผลลัพธ์ที่ได้อาจจะไม่สามารถเป็นจริงได้เสมอในเหตุการณ์จริง เนื่องจากว่าข้อมูลที่ใช้ในการคำนวณนั้นเป็นข้อมูลสมมติที่มีเพียงค่าเดียว หรือไม่ก็มีเพียงจำนวนน้อย เมื่อเทียบกับสถานการณ์จริงแล้ว ข้อมูลสมมติเหล่านี้ก็ไม่อาจจะเป็นตัวแทนของเหตุการณ์จริงได้อย่างสมบูรณ์ แต่สำหรับระบบที่ไม่ซับซ้อนมากจนเกินไป การคำนวณจากข้อมูลที่เปลี่ยนไปหลายๆ ครั้งก็อาจทำให้เราสามารถสรุปได้ว่าอัลกอริทึมไหน ควรจะได้รับการคัดเลือก

ในทางปฏิบัติแล้ว ระบบงานและระบบคอมพิวเตอร์มีความซับซ้อนมากกว่านี้มากมาย การใช้วิธี Deterministic Modeling นี้มีข้อจำกัดมากเกินไปสำหรับการนำมาใช้กับระบบคอมพิวเตอร์ในปัจจุบัน

4.7.2 โมเดลการจัดคิว

ลักษณะของงานที่เข้ามาในระบบคอมพิวเตอร์นั้น มักจะมีลักษณะไม่แน่นอน ในแต่ละวันที่มีการใช้ระบบคอมพิวเตอร์นั้น งานต่างๆ ที่เข้ามาอาจมีลักษณะที่ไม่ซ้ำกันเลย อย่างไรก็ตาม มีสิ่งหนึ่งที่อาจจะสามารถทำนายหรือกำหนดได้ นั่นคือการกระจายของเวลาในการใช้ซีพียู และของการใช้อินพุต/เอาต์พุต ซึ่งสามารถที่จะกำหนดแบบคร่าวๆ ได้ ซึ่งก็เช่นเดียวกันกับเวลาของการมาถึงระบบของงานต่าง ๆ ที่สามารถกำหนดไว้แบบการกระจายเช่นกัน จากการกำหนดค่ากระจายดังกล่าว ก็จะทำให้สามารถที่จะคำนวณว่าค่าเฉลี่ยของทรูพุต การใช้งานซีพียู เวลาคอย หรือคุณสมบัติอื่นๆ ได้อีกมากมาย

ระบบคอมพิวเตอร์สามารถอธิบายได้ว่า เป็นระบบเครือข่ายของการให้บริการแบบสถานี หรือมองเป็นสายงานการผลิต โดยที่แต่ละสถานีมีคิวของตัวเอง ไม่ว่าจะเป็นตัวซีพียูหรืออุปกรณ์ต่างๆ ถ้ารู้ว่าเวลาของการให้บริการของแต่ละสถานีนั้นนานแค่ไหน และรู้ว่าเวลาที่งานต่างๆ จะเข้าที่คิว เราก็สามารถคำนวณหาค่าที่ต้องการต่างๆ ได้แล้ว วิธีการนี้จึงมีชื่อเรียกว่าการวิเคราะห์การจัดคิดแบบแบบเน็ตเวิร์ค (Queuing-network analysis)

ตัวอย่าง สมมติให้ n คือค่าเฉลี่ยความยาวของคิว โดยไม่รวมงานที่กำลังทำอยู่ ให้ w คือค่าเฉลี่ยของเวลาที่ต้องคอยในคิว ให้ λ เป็นค่าเฉลี่ยของเวลาที่งานใหม่จะเข้ามาในระบบ เช่น 3

งานต่อวินาที เป็นต้น ต่อจากนั้นเราก็จะสามารถคำนวณได้ว่าในขณะที่งานใดงานหนึ่งคอยเป็นเวลา w งานใหม่ก็จะเข้ามาเป็นจำนวน λ x w ดังนั้น n = λ x w

สูตรดังกล่าวมีชื่อเรียกว่า Little's Formula สูตรนี้มีความสำคัญมากในการคำนวณ เพราะว่าสามารถใช้กับอัลกอริทึมใด ๆ ก็ได้ ที่เรารู้ค่าเฉลี่ยการกระจายของเวลาในการมาถึงระบบของงาน และอย่างน้อยเราก็สามารถคำนวณหาค่าของตัวใดตัวหนึ่งใน 3 ตัวแปร n, w, λ นี้ ถ้าเรารู้ค่าของตัวแปรอีกสองตัว ตัวอย่างเช่น ถ้าเรารู้ว่า จะมีงานเข้ามาในระบบ 7 งานต่อวินาทีโดยเฉลี่ย และถ้าจำนวนงานอยู่ในแถวคอย 14 ตัว เราก็สามารถทราบได้ว่าค่าเฉลี่ยของการคอยในแถวคอยเป็น 2 วินาที

การวิเคราะห์การจัดคิวเป็นวิธีที่มีประโยชน์มาก เพราะสามารถใช้คำนวณหาตัวเลขต่าง ๆ เช่น เวลาเฉลี่ยของการคอยของแต่ละอัลกอริทึมเอาไว้เปรียบเทียบกัน อย่างไรก็ตามวิธีนี้ก็ยังมีปัญหาอยู่ตรงที่สามารถใช้คำนวณกับอัลกอริทึมได้ไม่ทั้งหมด และยังมีปัญหาในเรื่องของการคำนวณหาค่าเฉลี่ยของการกระจายต่าง ๆ เพราะถ้าระบบหรืออัลกอริทึมมีความซับซ้อนมาก การตั้งสูตรเพื่อคำนวณหาค่าที่ถูกต้องจริงๆ มักจะทำได้ยาก ดังนั้นการตั้งค่าตัวเลขของค่าเฉลี่ยในการกระจาย มักมีการกำหนดไว้อย่างง่ายๆ ซึ่งอาจทำให้ผลการคำนวณไม่เที่ยงตรงเท่าที่ควร

4.7.3 วิธีการจำลองระบบ (Simulations)

การจำลองระบบจะเกี่ยวข้องกับการใช้โปรแกรมคอมพิวเตอร์ ซึ่งจะต้องมีการเขียนโปรแกรมเพื่อให้เป็นตัวแทนหรือหุ่นจำลองของระบบต่างๆ ในคอมพิวเตอร์ นอกจากนี้ยังต้องมีการเขียนโปรแกรมเพื่อเป็นตัวแทนของสิ่งแวดล้อมที่เกี่ยวข้อกับระบบคอมพิวเตอร์นั้นๆ อีกด้วย

สิ่งสำคัญในการสร้างแบบจำลองคือการมีตัวแปรที่เป็นตัวแทนของเวลา เพื่อเป็นสิ่งอ้างถึงตำแหน่งหรือเป็นหลักในการสร้างเหตุการณ์ต่างๆ ให้เกิดขึ้น ตัวแปรเวลานี้จะมีค่าเพิ่มขึ้นเรื่อยๆ ตามเวลาที่เปลี่ยนไป หรือไม่ก็เปลี่ยนตามเหตุการณ์ที่จะเกิดขึ้น ซึ่งเหตุการณ์ต่างๆ ในคอมพิวเตอร์ที่เกี่ยวข้องกับเวลาทั้งหมดจะต้องใช้ตัวแปรเวลาเดียวกัน

การทำแบบจำลองเพื่อการเปรียบเทียบอัลกอริทึมนี้ ต้องอาศัยเวลาในการทำงานค่อนข้างมากเพื่อให้ได้ผลลัพธ์ที่เที่ยงตรง ทำให้วิธีนี้อาจมีราคาสูง และยิ่งทำให้การจำลองแบบมีความถูกต้องมากเท่าใดก็จะยิ่งทำให้เสียเงินเสียเวลามากเท่านั้น ดังนั้นผู้ที่จะใช้วิธีแบบจำลองจะต้องตัดสินใจว่า จะใช้การจำลองแบบที่มีความละเอียดสูงแค่ไหนที่จะเพียงพอในการตัดสินใจ โดยไม่ทำให้เสียเงินและเวลามากเกินความจำเป็น

4.7.4 วิธีการสร้างขึ้นมาจริง

การสร้างแบบจำลองยังคงเป็นการจำลองแบบที่ไม่มีทางจะเหมือนจริงได้ สิ่งที่ดีกว่าคือการสร้างอัลกอริทึมชนิดต่างๆ เพื่อใช้กับโปรแกรมจัดการระบบจริงๆ ในระบบคอมพิวเตอร์ที่ใช้งานในสิ่งแวดล้อมจริง

แต่วิธีการนี้จะไม่เป็นที่นิยมในการปฏิบัติคือ จะใช้ต้นทุนหรือการลงทุนที่สูงมาก เพราะนอกจากจะต้องเขียนโปรแกรมของอัลกอริทึมแต่ละแบบแล้ว ยังต้องมีการใช้เวลาและแรงงานในการเปลี่ยนการทำงานของโปรแกรมเพื่อให้สามารถรองรับอัลกอริทึมแบบต่างๆ ที่จะนำมาทดลอง ยิ่งไปกว่านั้นแต่ละอัลกอริทึมอาจยังต้องการระบบฮาร์ดแวร์พิเศษที่แตกต่างกันออกไป ทำให้ต้องมีระบบคอมพิวเตอร์ที่ต้องสามารถปรับเปลี่ยนได้ ซึ่งเป็นเรื่องยากมาก และที่เป็นปัญหาที่สุดคือปัญหาที่อาจเกิดจากผู้ใช้คอมพิวเตอร์นั่นเอง เพราะคงมีผู้ใช้ไม่มากนักที่จะยอมเสียเวลานำงานจริงมาเสี่ยงกับระบบคอมพิวเตอร์ที่ยังอยู่ในขั้นทดลอง

(น.ท.ไพศาล โมลิสกุลมงคล และคนอื่น ๆ, 2545)

บทสรุป

การจัดเวลาให้กับซีพียู เป็นหลักความต้องการพื้นฐานของระบบปฏิบัติการ อัลกอริทึมในการจัดตารางการทำงานซีพียูมีหลายวิธี ได้แก่ การจัดเวลาแบบมาก่อนได้ก่อน การจัดเวลาแบบงานสั้นทำก่อน การจัดเวลาตามลำดับความสำคัญ การจัดเวลาแบบวนรอบ การจัดเวลาแบบคิวหลายระดับ ซึ่งมีข้อการพิจารณาจำนวนหลายข้อ ได้ถูกกำหนดไว้เพื่อใช้ในการเปรียบเทียบอัลกอริทึมต่างๆ ที่จะถูกคัดเลือกมาใช้งานในระบบปฏิบัติการเพื่อจัดเวลาการทำงานให้กับซีพียู หลักเกณฑ์ในการเปรียบเทียบอัลกอริทึมต่างๆ มักจะมาจากอรรถประโยชน์ของซีพียู (CPU utilization) เวลาตอบสนอง (Response time) และทรูพุต (Throughput) ซึ่งต้องมีการให้ค่าความสำคัญของแต่และคุณสมบัติไว้

คำถามท้ายบทที่ 4

1. ข้อมูลในระบบสารสนเทศทางภูมิศาสตร์แบ่งออกเป็นกี่ประเภทอะไรบ้าง
2. การให้สิทธิการจัดเวลา (Preemptive Scheduling) มีหลักการทำงานอย่างไร
3. ข้อพิจารณาในการจัดเวลาของซีพียูมีอะไรบ้าง
4. จงอธิบายหลักการทำงานของอัลกอริทึมการจัดเวลาต่อไปนี้มาพอเข้าใจ
   - การจัดเวลาแบบมาก่อนได้ก่อน (FCFS : First-Come First-Served)
   - การจัดเวลาแบบงานสั้นทำก่อน (SJF : Short-Job-First Scheduling)
   - การจัดเวลาตามลำดับความสำคัญ (Priority Scheduling)
   - การจัดเวลาแบบวนรอบ (RR : Round-Robin Scheduling)
   - การจัดเวลาแบบคิวหลายระดับ (Multilevel Queue Scheduling)
5. กำหนดให้โปรเซสเข้ามาสู่ระบบ มีข้อมูลตามตารางนี้

โปรเซส	เวลาที่ใช้	ลำดับความสำคัญ
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

ถ้าโปรเซสเข้ามาตามลำดับคือ P1, P2, P3, P4, P5

- จงแสดงถึงการจัดเวลาซีพียูของโปรเซสตามอัลกอริทึมแบบ FCFS, SJF, Priority (ตัวเลขน้อยคือ ค่าลำดับความสำคัญสูง) และ วิธีการของ RR เมื่อ quantum =1
- จงหาค่าเฉลี่ยของการคอยในคิวของแต่ละโปรเซสโดยใช้อัลกอริทึมแบบ FCFS, SJF และ RR

เอกสารอ้างอิง

น.ท.ไพศาล โมลิสกุลมงคล, ผศ.น.ท.ดร.ประสงค์ ปราณีตพลกรัง, น.ต.เมธา สุนทรสารทูล, น.ต.สุชาติ วีรกุลวัฒนา, ร.ท.อนุโชติ วุฒิพรพงษ์. (2545). **ระบบปฏิบัติการ (Operating Systems)**. กรุงเทพฯ : ไทยเจริญการพิมพ์.

นรีรัตน์ นิยมไทย. (2549). **ระบบปฏิบัติการ**. กรุงเทพฯ : ศูนย์ส่งเสริมวิชาการ.

พิรพร หมุนสนิท, สุธีพงศาสกุลชัย และอัจจิมา เลี้ยงอยู่. (2553). **ระบบปฏิบัติการ (Operating Systems)**. กรุงเทพฯ : เคทีพี.

วิเชษฐ์ พลายมาศ. (2552). **ระบบปฏิบัติการ (Operating Systems)**. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

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

(2552). [Online]. Available HTTP: http://csnon04.blogspot.com/2008/03/3_06.html.

(2552). [Online]. Available HTTP: http://cs1.mcm.edu/~rob/class/csc4340/notes/Chapt06/Chapter6.html.

(2552). [Online]. Available HTTP: http://490702464037.exteen.com/20080818/5-cpu-scheduling.

(2552). [Online]. Available HTTP: http://www.cs.wayne.edu/~tom/guide/os.html.

(2552). [Online]. Available HTTP: http://www.cs.uic.edu/~i385/CourseNotes/5_CPU_Scheduling.html.

(2552). [Online]. Available HTTP: http://os1rash.blogspot.com/2008/08/5-cpu-scheduling.html.

(2552). [Online]. Available HTTP: http://read.cs.ucla.edu/111/notes/lec7.