บทที่ 4
การจัดตารางเวลาซีพียูและอัลกอริธึมตารางเวลา
(CPU Scheduling and Scheduling Algorithms)

กระบวนการหรือโปรเซส ซึ่งคือหน่วยของงานในระบบ ที่กำลังดำเนินงานที่ซีพียู ดังนั้นระบบที่กำลังทำงานประกอบด้วยชุดของกระบวนการหลาย ๆ กระบวนการ มีทั้งกระบวนการของผู้ใช้และกระบวนการของระบบปฏิบัติการ โดยที่กระบวนการเหล่านี้ทั้งหมดสามารถดำเนินการให้สำเร็จร่วมกันบนซีพียูเดียว หรือหลายซีพียูโดยกระบวนการเหล่านั้นสลับกันเข้าดำเนินการที่ซีพียูตามตารางเวลาของการใช้ซีพียู โดยมีการใช้อัลกอริธึมที่เหมาะสมเพื่อให้ระบบมีการใช้ซีพียูได้อย่างมีประสิทธิภาพ

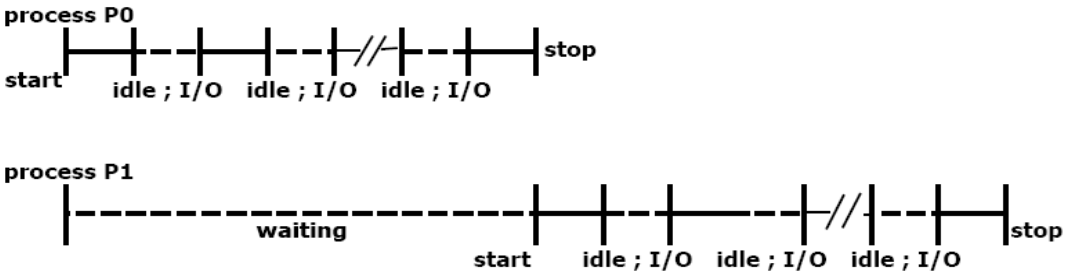
แนวคิดการจัดตารางเวลา

การทำงานของระบบมัลติโปรแกรมมิ่งมีจุดประสงค์เพื่อให้มีหลายกระบวนการเข้าดำเนินการประมวลผลที่ซีพียู เพื่อเป็นการเพิ่มประสิทธิภาพการทำงานของซีพียู สำหรับระบบที่มีซีพียูเดียวจะมีเพียงกระบวนการเดียวเท่านั้นที่จะได้เข้าดำเนินการบนซีพียู กระบวนการที่เหลือต้องรอจนกว่าซีพียูจะว่าง จึงจะได้รับการจัดสรรให้เข้าทำงานได้

แนวความคิดการทำงานของระบบมัลติโปรแกรมมิ่งเป็นการจัดการที่ง่ายสำหรับกระบวนการในการทำงานของระบบคอมพิวเตอร์แบบเรียบง่าย เมื่อกระบวนการหนึ่งกำลังใช้ที่ซีพียู จะต้องใช้จนเสร็จถึงแม้จะต้องทำ ไอ/โอ ด้วยก็ตาม ขณะทำไอ/โอ ซีพียูจะต้องรอเพื่อที่จะประมวลผลกระบวนการนั้นต่อให้เสร็จ ซึ่งเป็นช่วงเวลาที่ซีพียูจะเสียเวลาในการรอ การแก้ปัญหานี้ใช้วิธีการของระบบมัลติโปรแกรมมิ่ง คือในขณะที่ซีพียูรอกระบวนการที่ต้องทำไอ/โอ ระบบปฏิบัติการจะจัดการให้ซีพียูดำเนินการกระบวนการอื่นก่อน สำหรับวิธีนี้สามารถใช้ประโยชน์จากซีพียูได้มาก เนื่องจากในหน่วยความจำเก็บกระบวนการไว้มากมายเพื่อรอดำเนินการ ดังนั้นเมื่อซีพียูว่างลงระบบปฏิบัติการจะเลือกกระบวนที่เหมาะสมเข้าดำเนินการทันที่จะไม่ปล่อยให้ซีพียูว่าง

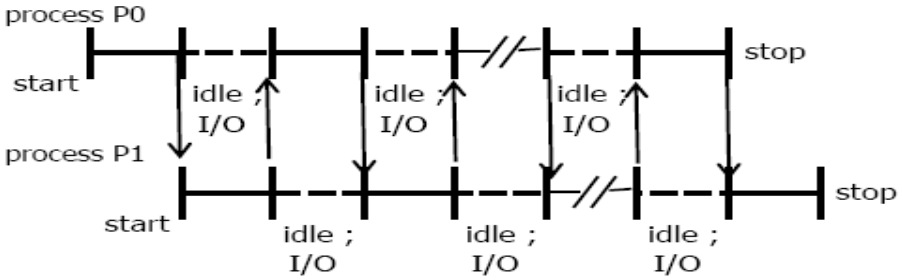
การประมวลผลของกระบวนการใด ๆ ในระบบ จะมีการทำงานหลัก ๆ คือดำเนินการที่ซีพียูหรือดำเนินการที่ไอ/โอ กระบวนการจะต้องนำข้อมูลจากอุปกรณ์อินพุตเข้ามาประมวลผลที่ซีพียู และนำผลลัพธ์ที่ได้ไปแสดงผลที่อุปกรณ์เอาท์พุต ขณะที่ประมวลผลกระบวนการดำเนินการที่ซีพียู และขณะที่รับหรือแสดงผลกระบวนการดำเนินการที่อุปกรณ์ ไอ/โอ ดังนั้นช่วงที่กระบวนการดำเนินการที่

ไอ/โอ ซีพียูจะว่างในระบบมัลติโปรแกรมมิ่งจึงนำกระบวนการอื่น ที่รออยู่ในคิวเข้าเข้าดำเนินการที่ซีพียูแทน ประโยชน์ของมัลติโปรแกรมมิ่งคือเพิ่มประสิทธิภาพการทำงานของซีพียู และเพิ่มผลลัพธ์ที่ได้จากการประมวลผล จะได้จำนวนงานที่ทำได้สำเร็จในช่วงเวลาใดเวลาหนึ่งที่กำหนดไว้เพิ่มมากขึ้น ยกตัวอย่างสมมุติเรามี 2 กระบวนการ คือ P0 และ P1 ในการทำงานจะดำเนินงานที่ซีพียู 1 วินาที ดำเนินการ ไอ/โอหรือรอดำเนินการใด ๆ 1 วินาที ทำซ้ำ ๆ กัน 60 ครั้ง ถ้าเรา ดำเนินการประมวลผล P0 และ P1 ต่อกันไป มันจะใช้เวลาทั้งหมด 4 นาที ในการดำเนินการประมวลผลทั้ง 2 กระบวนการ P0 ใช้เวลาในการดำเนินการ 2 นาทีเสร็จงาน P1 ใช้เวลาทำงาน 2 นาทีแต่ต้องรออีก 2 นาที รวมแล้ว P1 ทำงานเสร็จใช้เวลา 4 นาที โดยที่ 2 นาทีเป็นเวลาที่เสียเปล่า ดังนั้น ซีพียูถูกใช้ให้เป็นประโยชน์เพียง 50% ดังรูปที่ 4.1 แสดง กระบวนการ P0 และ P1 ดำเนินการประมวลผลโดยไม่ใช้มัลติโปรแกรมมิ่ง



รูปที่ 4.1 แสดงกระบวนการ P0 และ P1 ดำเนินการประมวลผลโดยไม่ใช้มัลติโปรแกรมมิ่ง

สำหรับรูปที่ 4.2 แสดง กระบวนการ P0 และ P1 ดำเนินการประมวลผลโดยใช้มัลติโปรแกรมมิ่ง ในการทำงานกระบวนการ P0 และ P1 จะสลับกันเข้าทำงานที่ ซีพียู โดยเริ่มต้น P0 เข้าดำเนินการที่ซีพียู 1 วินาที P1 รอ เมื่อ P0 เข้าทำ I/O 1 วินาที P1สลับ เข้าดำเนินการที่ซีพียู 1 วินาที แล้วพอ P1 เข้าทำ ไอ/โอ 1 วินาที P0 สลับเข้าดำเนินการที่ซีพียู 1 วินาที P0 และP1 สลับกันดำเนินงานจนงานเสร็จจะใช้เวลา 121 วินาที่ P1 จะรอและเสร็จงานช้ากว่า P0 เพียง 1 วินาที

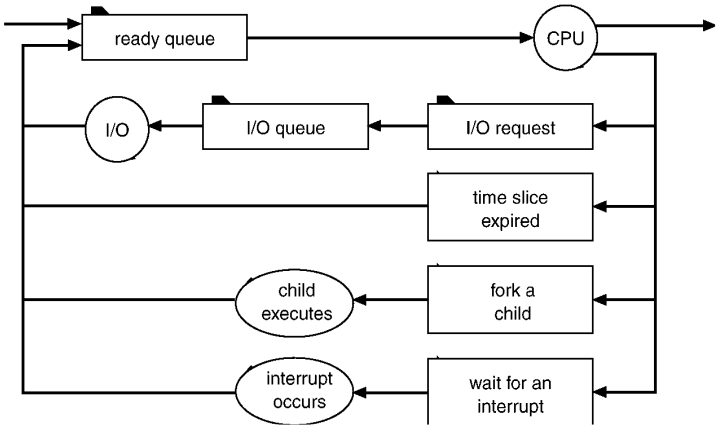


รูปที่ 4.2 แสดงกระบวนการ P0 และ P1 ดำเนินการประมวลผลโดยใช้มัลติโปรแกรมมิ่ง

การจัดตารางเวลาคิว (Scheduling Queues)

กระบวนการที่เข้าไปในระบบจะถูกเก็บไว้ในคิวของงาน (Job Queue) ซึ่งคิว (Queue) ประกอบด้วยกระบวนการทั้งหมดที่อยู่บนสื่อจัดเก็บ (Mass Storage) ที่กำลังรอรับการจัดสรรพื้นที่ของหน่วยความจำหลัก กระบวนการที่อยู่ในหน่วยความจำหลักและพร้อมแล้วกำลังรอการดำเนินการประมวลผลจะถูกเก็บไว้ในลิสต์ ที่เรียกว่าคิวพร้อม (Ready Queue) ซึ่งมีลักษณะเป็นลิสต์ของรายการกระบวนการที่พร้อมในคิวพร้อมจะรับคำสั่งให้ดำเนินการประมวลผลครั้งละ 1 กระบวนการ โดยประกอบด้วยตัวชี้ (Pointer) ที่ชี้ไปยังตำแหน่งถัดไปของกระบวนการในคิวพร้อมที่จะเข้าดำเนินการต่อไป

ภายในระบบมีคิวอื่นด้วย เมื่อกระบวนการได้รับการจัดสรรให้เข้าใช้ซีพียู มันจะดำเนินการประมวลผล (Execute) ในขณะที่กระบวนการกำลังประมวลผลอาจจะต้องการทำ ไอ/โอ กระบวนการต้องถูกขัดจังหวะเพื่อจะให้ไปทำ ไอ/โอ แต่เนื่องจากอาจจะมีหลายกระบวนการที่กำลังใช้อุปกรณ์ ไอ/โอ กระบวนการนี้จะต้องเข้าไปอยู่ในคิวของ ไอ/โอ เพื่อรอเวลาการเข้าใช้ อุปกรณ์บางอย่างสามารถใช้งานร่วมกันได้อาจจะไม่ต้องเข้าคิวรอ โดยทั่วไปแต่ละอุปกรณ์จะมีคิวของตัวเองเรียกว่าคิวอุปกรณ์ (Device Queue) การใช้แผนภาพการจัดคิว (Queuing Diagram) เป็นการอธิบายการจัดตารางเวลาให้กับแต่ละกระบวนการ ดังรูปที่ 4.3 แต่ละกล่องสี่เหลี่ยมผืนผ้าแทนคิวของกระบวนการ มีคิว อยู่ 2 ชนิด คือ คิวพร้อม (Ready Queue) และ คิวของอุปกรณ์ ไอ/โอ วงกลมแทนทรัพยากร ลูกศรชี้เป็นเส้นทางเดินของกระบวนการในระบบ



รูปที่ 4.3 แสดงแผนภาพตารางเวลาทำงานของกระบวนการ

กระบวนการใหม่จะถูกนำไปเก็บไว้ในแถวพร้อม (Ready) รออยู่ในคิวพร้อมจนกระทั่งมันถูกเลือกให้ไปดำเนินการประมวลผลและได้เข้าใช้ซีพียู ในขณะที่กระบวนการได้รับการจัดสรรให้เข้าใช้ซีพียู และกำลังดำเนินการประมวลผล อาจจะเกิดเหตุการณ์ใดเหตุการณ์หนึ่งดังต่อไปนี้

- กระบวนการอาจจะขอใช้ ไอ/โอ ระบบปฏิบัติการจึงส่งให้ไปอยู่ใน ไอ/โอ คิว
- กระบวนการอาจจะสร้าง (Fork) กระบวนการใหม่ และ การรอการเข้าประมวลผลต่อไป
- กระบวนการอาจจะถูกยกเลิกต้องออกจากการใช้ซีพียู เช่นการหมดช่วงเวลา (Time Slice) หรือได้รับการอินเทอร์รัพจะต้องกลับเข้าไปอยู่ในคิวพร้อม

เมื่อกระบวนการดำเนินการ ไอ/โอ เสร็จจะต้องเข้าไปอยู่ในสถานะพร้อม และกระบวนการใหม่ที่ถูกสร้างขึ้นมายังประมวลผลไม่เสร็จจะต้องเข้าไปอยู่ในสถานะพร้อมเช่นกัน จะเห็นว่าทุกกระบวนการจะอยู่ในวงจรการดำเนินงานขณะเวลาที่มันอยู่ในระบบ จนทำงานเสร็จจึงออกจากวงจรนี้

ตัวจัดตารางเวลาทำงาน (Schedulers)

ระบบปฏิบัติการทำหน้าที่ในการคัดเลือกกระบวนการเข้าสู่คิวตารางเวลา เพื่อเข้าดำเนินการประมวลผลจนกระทั้งทำงานเสร็จ ตัวที่ทำหน้าที่เลือกกระบวนการจากคิวเข้าดำเนินการที่ซีพียูคือ ตัวจัดตารางเวลาทำงาน ซึ่งมีหลายวิธีดังต่อไปนี้

- ตัวจัดตารางเวลาระยะยาวหรือตัวจัดตารางเวลางาน ในระบบการทำงานแบบแบทมีงานหลาย ๆ งานที่ต้องการเข้าประมวลผลงานเหล่านี้จะถูกโหลดเข้าสู่แหล่งดิสก์ ซึ่งมันจะถูกเก็บไว้ประมวลผลต่อไป ตัวจัดตารางเวลาทำงานระยะยาวจะทำการเลือกงานจากแหล่งดิสก์นี้และโหลดเข้าสู่หน่วยความจำเพื่อทำการประมวลผลต่อไป งานที่นำเข้าประมวลผลในลักษณะมัลติโปรแกรมมิ่ง เวลาที่ใช้ในการดำเนินงานแต่ละงานจนเสร็จจะใช้เวลานานกว่าตัวจัดตารางเวลาทำงานระยะสั้นแต่ละงานอาจจะใช้เวลาเป็นวินาที หรือเป็นนาที
- ตัวจัดตารางเวลาทำงานระยะสั้นตัวจัดตารางเวลาซีพียู ในระบบมีกระบวนการจำนวนมากที่จะเข้าดำเนินการประมวลผลมันอยู่ในสถานะพร้อม ตัวจัดตารางเวลาทำงานระยะสั้นจะทำการเลือกกระบวนการที่พร้อมเพียง 1 กระบวนการเพื่อให้ได้รับการจัดสรรให้เข้าประมวลผลที่ซีพียู ซึ่งระยะเวลาที่ใช้ประมวลผลจะเป็นช่วงเวลาสั้น ๆ ไม่กี่มิลลิวินาที สลับการทำงานกับหลาย ๆ กระบวนการ
- ตัวจัดตารางเวลาทำงานระยะกลาง บางระบบปฏิบัติการ เช่นระบบแบ่งเวลา ต้องการเพิ่มประสิทธิภาพในการทำงานของระบบ จึงใช้ตัวจัดตารางเวลาทำงานระยะกลางมาช่วย โดยมีหลักการคือการยกเลิกกระบวนการในหน่วยความจำ การลดระดับของมัลติโปรแกรมมิ่ง หรือเพิ่มระดับมัลติ

โปรแกรมมิ่ง ตามสภาวะของประสิทธิภาพการทำงานของซีพียู โดยกระบวนการจะถูกสลับเข้า (Swap In) และสลับออก (Swap Out)

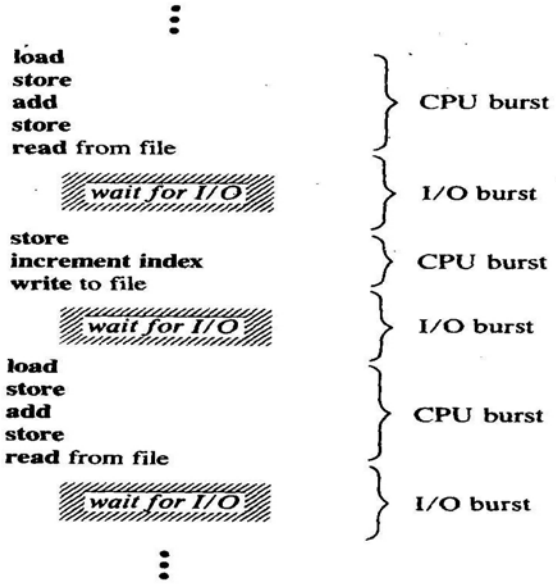
กระบวนการเวลาทำงานส่วนมากสามารถระบุได้ว่าเป็นเขต ไอ/โอ (I/O Bound) หรือเป็นเขตซีพียู (CPU Bound) ถ้าอยู่ในเขตไอ/โอ กระบวนการใช้เวลาในการทำไอ/โอ มากกว่าการคำนวณ ถ้าอยู่ในเขตซีพียูกระบวนการใช้เวลาในการคำนวณมากกว่าทำ ไอ/โอ คือมีการเรียกใช้ไอ/โอ น้อยตัวจัดตารางเวลาทำงานระยะสั้นจะทำงานน้อย ถ้ากระบวนการทั้งหมดเป็นเขตซีพียู การใช้ไอ/โอ คิวจะว่าง และการทำงานของระบบจะไม่สมดุล

ตารางเวลาทำงานของซีพียู (CPU Scheduling) ระบบปฏิบัติการจะทำหน้าที่ในการจัดตารางเวลาการทำงานพื้นฐานของระบบคอมพิวเตอร์ โดยทรัพยากรเกือบทั้งหมดจะกำหนดถูกตารางเวลาก่อนการเข้าใช้ซีพียู ซึ่งเป็นทรัพยากรหลักของระบบคอมพิวเตอร์ ดังนั้นการจัดตารางเวลาการทำงานจึงเป็นหน้าที่ ที่สำคัญของระบบปฏิบัติการ

วงรอบการทำงานของซีพียูและไอ/โอ (CPU-I/O Burst Cycle) การทำงานของซีพียูจะมีประสิทธิภาพขึ้นอยู่กับคุณสมบัติของกระบวนการ โดยทั่วไปการประมวลผลกระบวนการ จะประกอบด้วย วงรอบการทำงานของซีพียู (CPU Burst Cycle) และวงรอบการทำงานของไอ/โอ (I/O Burst Cycle) ขณะที่กระบวนการกำลังประมวลผลจะสลับการทำงานระหว่าง 2 วงรอบนี้เท่านั้น โดยทั่วไปการประมวลผลจะเริ่มที่วงรอบการทำงานของซีพียู จากนั้นจะตามด้วยวงรอบของไอ/โอ เมื่อเสร็จจากไอ/โอ วงรอบการทำงานของซีพียู จะดำเนินการต่อสลับกันไปเรื่อย ๆ จนทำงานเสร็จการสิ้นสุดการประมวลผลกระบวนการส่วนใหญ่จะสิ้นสุดที่วงรอบการทำงานของซีพียู ดังรูป 4.4

ตัวจัดตารางเวลาซีพียู (CPU Scheduler)

เมื่อไรก็ตามทีซีพียูว่าง ระบบปฏิบัติการต้องเลือกกระบวนการหนึ่งกระบวนการในคิวพร้อมเพื่อเข้ามาทำการประมวลผลที่ซีพียู ซึ่งจัดการโดยตัวจัดตารางเวลาทำงานระยะสั้น ตัวจัดตารางเวลานี้จะเลือกจากกระบวนการในหน่วยความจำซึ่งพร้อมที่สุดเข้าดำเนินการที่ซีพียู คิวไม่จำเป็นต้องเป็นแบบมาก่อนได้รับการเลือกก่อน (First-In-First-Out : FIFO) อาจจะเป็นไปตามลำดับความสำคัญ (Priority) ของกระบวนการก็ได้ หรือเป็นไปตามลิงค์ลิสต์ (Linked List) อย่างไรก็ตามทุกกระบวนการจะต้องเข้าประมวลผลที่ซีพียู ซึ่งในการดำเนินการประมวลผลของแต่ละกระบวนการต้องอ่านและบันทึกสถานะการทำงานของกระบวนการ จากบลอกก์ควบคุมกระบวนการของแต่ละกระบวนการ



รูปที่ 4.4 ลำดับเวลาทำงานซีพียู และไอ/โอ

โครงสร้างของตารางเวลา (Scheduling Structure)

การตัดสินใจของของซีพียู ในการเลือกกระบวนใด ๆ การเข้ามาดำเนินการประมวลผลขึ้นกับเหตุการณ์ดังต่อไปนี้

- เมื่อกระบวนการเปลี่ยนจากสถานะจากกำลังประมวลผลเป็นสถานะกำลังรอ ตัวอย่างเช่น การรอใช้ไอ/โอ การรอให้กระบวนการลูกทำงานเสร็จ

- เมื่อกระบวนการเปลี่ยนจากจากกำลังประมวลผลไปยังสถานะพร้อม ตัวอย่างเช่น เมื่อเกิดอินเทอร์รัพหมดเวลาการใช้งานซีพียู

- เมื่อกระบวนการเปลี่ยนจากสถานะกำลังรอไปยังสถานะพร้อม ตัวอย่างเช่น เมื่อทำไอ/โอเสร็จ

- เมื่อมีกระบวนการทำงานเสร็จแล้ว ถูกยกเลิกออกจากหน่วยความจำ

สำหรับกรณีของเหตุกาณ์แรกและเหตุการณ์สุดท้าย จะต้องตัดสินใจเลือกกระบวนการใดกระบวนการหนึ่งเข้ามาดำเนินการในซีพียู ต่อส่วนเหตุการณ์ที่ 2 และ 3 การตัดสินใจเลือกระบวนการต้องอยู่บนพื้นฐานของการใช้กฎเกณฑ์ของอัลกอริธึมการจัดตารางเวลาซีพียู เหตุการณ์ 1 และ 4 จะเป็นแบบไม่ให้สิทธิ์ก่อน (Non Preemptive) นอกจากนั้นเป็นแบบให้สิทธิ์ก่อน (Preemptive)

การสับเปลี่ยนงาน (Context Switch) การเปลี่ยนกระบวนการเข้าใช้งานที่ซีพียู ระบบจะทำการบันทึกสถานะการทำงานของกระบวนการเก่าไว้ในบล็อกควบคุมกระบวนการของมัน และทำการโหลดสถานะการทำงานของกระบวนการใหม่จากบล็อกควบคุมกระบวนการขึ้นมาทำงาน เรียกว่าการสับเปลี่ยนงาน (Context Switch) เวลาที่ใช้สำหรับการสับเปลี่ยนงานเป็นเวลาที่ต้องสูญเสียไปจะใช้เวลามากหรือน้อย ขึ้นอยู่กับความเร็วของหน่วยความจำ จำนวนของรีจีสเตอร์ และคำสั่งพิเศษ เวลาที่ใช้ในการเกิดจากการสับเปลี่ยนงานจะอยู่ในช่วง 1-100 ไมโครวินาที (Microseconds) หรือ10^{-6} วินาที

ผู้จัดการ (Dispatcher) ส่วนประกอบที่สำคัญของการจัดตารางเวลาซีพียูอีกส่วนหนึ่งคือผู้จัดการ ทำหน้าที่เป็นหน่วยควบคุมการจัดตารางเวลาของซีพียู เพื่อเลือกกระบวนการเข้ามาดำเนินการ โดยมีหน้าที่ดังนี้

- การเปลี่ยนเนื้อความ (Context)
- การเปลี่ยนโหมดผู้ใช้
- การกระโดดไปสู่ตำแหน่งที่เหมาะสมในโปรแกรมผู้ใช้เพื่อเริ่มต้นโปรแกรมอีกครั้ง

อัลกอริธึมของตารางเวลา

การจัดตารางเวลาการของการเข้าใช้ซีพียู โดยการเลือกกระบวนการในคิวพร้อมเข้าทำงานที่ซีพียูต้องใช้อัลกอริธึมที่เหมาะสมกับคุณสมบัติของกระบวนการและสถานการณ์ต่าง ๆ ข้อที่ควรพิจารณาเพื่อใช้ในการเลือกใช้อัลกอริธึมที่ดีทสุด มีกฎเกณฑ์ที่ใช้ดังต่อไปนี้

- การใช้ประโยชน์ซีพียู (CPU Utilization) ต้องคำนึงถึงการใช้ประโยชน์จากซีพียูให้มากที่สุด ซีพียูจะต้องประมวลผลให้ได้มากที่สุดจะอยู่ในช่วงจาก 0 - 100 เปอร์เซ็น ในระบบทำงานจริงควรอยู่ในช่วง 40 – 90 เปอร์เซ็น
- ผลผลิต (Throughput) ถ้าซีพียูทำการประมวลผลงานอย่างต่อเนื่องตลอดเวลา ผลผลิตที่ได้จากการงานที่เสร็จ โดยวัดจากจำนวนงานที่เสร็จสิ้นต่อหน่วยเวลา จะต้องมากขึ้น
- เวลาที่ใช้ทั้งหมด (Turnaround Time) ช่วงเวลาตั้งแต่เริ่มต้นเข้าไปในระบบ จนงานเสร็จเรียบร้อยได้ผลลัพธ์ เป็นผลรวมของเวลาที่เสียไปในการรอเพื่อเข้าสู่หน่วยความจำ รอในคิวพร้อม รอการประมวลผล การเข้าใช้ซีพียู และการทำ ไอ/โอ เวลาที่ใช้ควรจะน้อย
- เวลารอ (Waiting Time) เป็นช่วงเวลาที่รอให้ตัวจัดตารางเวลาเลือกเข้าดำเนินการที่ซีพียู โดยทั่วไปเป็นเวลาที่กระบวนการรออยู่ในคิวพร้อมเวลารอควรจะน้อย

- เวลาตอบสนอง (Response Time) คือเวลาตั้งแต่ส่งคำสั่งเข้าดำเนินการจนกระทั่งได้ผลลัพธ์ออกมาครั้งแรก ถ้ามีหลายกระบวนการโดยส่วนมากเราจะวัดจากค่าเฉลี่ย เวลาตอบสนองควรจะน้อย

อัลกอริธึมสำหรับการจัดตารางเวลาของกระบวนการ จะช่วยตัดสินใจว่าควรจะเลือกกระบวนการไหนเข้าใช้ซีพียูก่อนซึ่งมีหลายอัลกอริธึมดังต่อไปนี้

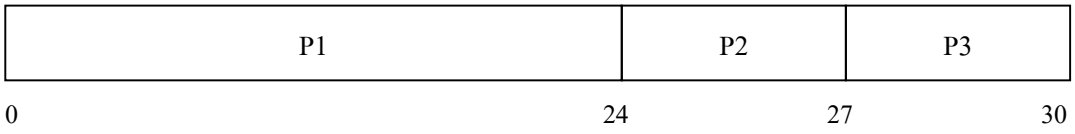
การจัดตารางเวลาแบบมาก่อนได้ก่อน (First-Come First-Served Scheduling)

เป็นอัลกอริธึมที่ง่ายที่สุด การจัดตารางเวลาแบบมาก่อนได้ก่อน เรียกสั้น ๆ ว่า FCFS ใช้หลักการคือ กระบวนการที่ร้องขอใช้ซีพียู ก่อนจะต้องได้รับการจัดสรรให้ใช้ซีพียูก่อนการพัฒนาอัลกอริธึมของ FCFS จัดการโดยคิวแบบมาก่อนได้ก่อน (FIFO Queue) เมื่อกระบวนการเข้ามาในคิวพร้อมบล็อกควบคุมกระบวนการของมันจะต่อเข้าที่หางคิว เมื่อซีพียูว่างระบบจะจัดสรรซีพียูให้กับบล็อกควบคุมกระบวนการของกระบวนการที่อยู่หัวคิว และกระบวนการที่หัวของคิวพร้อมจะถูกลบออกจากคิว

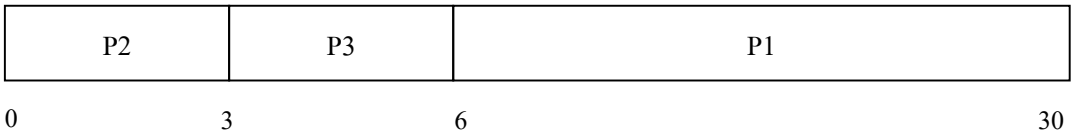
ตัวอย่างการหาเวลาการรอเฉลี่ยภายใต้ FCFS พิจารณาชุดข้อมูลของกระบวนการ ซึ่งเข้ามาถึงระบบที่เวลา 0 กับเวลาของการใช้ซีพียู มีหน่วยเป็นมิลลิวินาที (Milliseconds) คือ10^{-3} วินาที ดังนี้

กระบวนการ	เวลาที่ต้องใช้ซีพียู
P1	24
P2	3
P3	3

กระบวนการที่เข้ามาขอใช้ซีพียู ทั้ง 3 กระบวนการคือ P1, P2, P3 ตามลำดับและใช้อัลกอริธึม FCFS ในการจัดตารางเวลาการเข้าใช้ซีพียู แสดงดัง ต่อไปนี้



สำหรับเวลาที่ใช้รอกระบวนการ P1 รอ 0 มิลลิวินาที กระบวนการ P2 รอ 24 มิลลิวินาที และกระบวนการ P3 รอ 27 มิลลิวินาที เวลารอเฉลี่ยคือ (0+24+27)/3 = 17 มิลลิวินาที ถ้าให้กระบวนการที่เข้ามาถึงในระบบเป็น P2, P3, P1 จะแสดงดังผลลัพธ์ดังนี้



กระบวนการ P1 รอ 6 มิลลิวินาที P2 รอ 0 มิลลิวินาที และ P3 รอ 3 มิลลิวินาที เวลารอเฉลี่ยคือ (6+0+3)/3 = 3 มิลลิวินาที จะพบว่าทั้ง 3 กระบวนการที่เข้าใช้ซีพียูเหมือนกันแต่เวลาการเข้าใช้ซีพียูเริ่มต้นต่างกันจะพบว่าเวลาที่รอลดลงอย่างเป็นรูปธรรม

อัลกอริธึมการจัดตารางเวลาแบบมาก่อนได้ก่อน เป็นการจัดการแบบไม่ให้สิทธิ์ก่อน (No-Preemptive) คือเมื่อกระบวนการใดได้รับการจัดสรรให้เข้าใช้ซีพียู กระบวนการนั้นจะดำเนินการบนซีพียูจนกว่ามันต้องการที่จะออกจากซีพียูเอง โดยการยกเลิกเมื่อหมดเวลาเข้าใช้ซีพียูหรือร้องขอเข้าใช้ไอ/โอ (Request I/O) อัลกอริธึมนี้ยุติธรรมกับทุกกระบวนการ แต่มีปัญหาเรื่องเวลาในการรอของกระบวนการที่ใช้เวลาน้อย ดังนั้นจึงมีการพัฒนาอัลกอริธึมใหม่ขึ้นมา

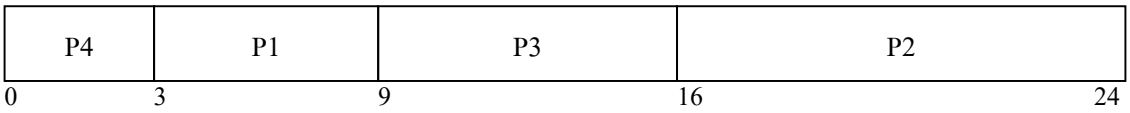
การจัดตารางเวลาแบบงานสั้นสุดได้ก่อน (Shortest-Job-First Scheduling)

อัลกอริธึมนี้สร้างขึ้นมาเพื่อแก้ปัญหาเรื่องเวลาที่ต้องรอนานของอัลกอริธึม FCFS การทำงานของอัลกอริธึมการจัดตารางเวลาแบบงานสั้นสุดได้ก่อน (Shortlist-Job-First Scheduling) หรือเรียกสั้น ๆ ว่าอัลกอริธึม SJF แต่ละกระบวนการที่รอเข้าทำการประมวลผลที่ซีพียูจะรู้เวลาของการใช้งานซีพียู เมื่อซีพียูว่างอัลกอริธึม SJF จะเลือกกระบวนการที่มีเวลาการใช้งานซีพียูน้อยที่สุดให้เข้าใช้งานที่ซีพียูก่อนกระบวนการอื่น

ตัวอย่างดังต่อไปนี้พิจารณาชุดของกระบวนการที่ขอใช้ซีพียูตามลำดับ กับเวลาที่ต้องใช้ซีพียูมีหน่วยเป็นมิลลิวินาที (10^{-3} วินาที)

กระบวนการ	เวลาที่ต้องใช้ซีพียู
P1	6
P2	8
P3	7
P4	3

กระบวนการที่เข้ามาขอใช้ซีพียู ทั้ง 4 กระบวนการคือ P1, P2, P3 และ P4 ตามลำดับและใช้อัลกอริธึม SJF ในการจัดตารางเวลาการเข้าใช้ซีพียู แสดงดังนี้



เวลาที่แต่ละกระบวนการรอ คือ P1 รอ 3 มิลลิวินาที P2 รอ 16 มิลลิวินาที P3 รอ 9 มิลลิวินาที และ P4 รอ 0 มิลลิวินาที หาเวลาการรอเฉลี่ยคือ (3+16+9+0)/4 = 7 มิลลิวินาที ถ้าเราใช้อัลกอริธึม ของ FCFS แล้วเวลาการรอเฉลี่ยคือ 10.25 มิลลิวินาที จะเห็นว่าอัลกอริธึม SJF จะให้เวลาการรอเฉลี่ยน้อยที่สุดสำหรับกระบวนการชุดเดียวกันนี้ ซึ่งพิสูจน์ได้ว่าการเลือกระบวนการที่ใช้เวลาสั้นเข้าดำเนินการก่อนกระบวนการที่ใช้เวลายาว จะลดเวลาการรอคอยของกระบวนการที่ใช้เวลาสั้นได้มาก แต่มันเพิ่มเวลาที่รอของกระบวนการที่ใช้เวลายาว แต่ถึงอย่างไรกตามเวลาการรอโดยเฉลี่ยจะลดลงได้มาก

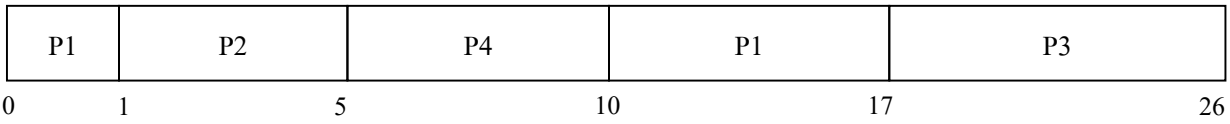
ถึงแม้ว่าว่าอัลกอริธึม SJF จะให้เวลารอเฉลี่ยดีที่สุด แต่เราไม่สามารถที่จะรู้ได้ว่า หลังจากที่กระบวนการได้ดำเนินการไปบ้างแล้ว ช่วงเวลาที่เหลือครั้งต่อไปของกระบวนการที่ต้องใช้ซีพียู (CPU Burst) มีเหลืออีกเท่าไร เราสามารถคำนวณหาค่าของความยาวเวลาที่มีเหลือของเวลาที่ต้องใช้ซีพียู โดยหาค่าเอ็กซ์โปเนนเชี่ยล (Exponential) เฉลี่ยของของความยาวเวลาของกระบวนการที่ต้องใช้ซีพียูก่อนหน้านี้ ให้ T_n คือความยาวเวลาของกระบวนการที่ n และให้ T_{n+1} เป็นค่าที่ทำนาย สำหรับความยาวเวลาของกระบวนการถัดไป จะได้ $T_{n+1} = \alpha t_n + (1 - \alpha)T_n$ สำหรับ α มีค่ามีค่าอยู่ระหว่าง 0 ถึง 1 โดยที่ตัวแปร α ใช้ควบคุมความสำคัญของเวลาที่ต้องใช้ซีพียูในอดีตที่ผ่านและปัจจุบัน ถ้า α มีค่าเป็น 1 แสดงว่าความยาวเวลาของกระบวนการที่เพิ่งผ่านมามีความสำคัญมากสุด โดยไม่คำนึงถึงความยาวเวลาที่ใช้ซีพียูของช่วงอื่น แต่ถ้า α มีค่าเป็น 0 แสดงว่าความยาวของเวลาที่ต้องใช้ซีพียูของกระบวนการในอดีตที่ผ่านมามีความสำคัญมากสุด โดยทั่วไปค่าที่นิยมใช้คือ α มีค่าเป็น 0.5 แสดงว่าข้อมูลเวลาการใช้ซีพียูในอดีตและปัจจุบันมีความสำคัญเท่ากัน

อัลกอริธึม SJF สามารถจัดการได้ทั้งแบบให้สิทธิ์ก่อน และแบบไม่ให้สิทธิ์ก่อน จากตัวอย่างข้างต้นเป็นการจัดการแบบไม่ให้สิทธิ์ก่อน แต่ในการทำงานของระบบ ถ้ามีกระบวนการใหม่เข้ามาในระบบและขอเข้าใช้งานที่ซีพียู ซึ่งกระบวนการใหม่ที่เข้ามานั้นอาจจะมีเวลาที่ต้องใช้ซีพียูน้อยกว่าของกระบวนการอื่น ๆ ที่กำลังใช้งานที่ซีพียูอยู่ อัลกอริธึม SJF สามารถจัดการได้ทั้งสองแบบ สำหรับแบบให้สิทธิ์ก่อน (Preemptive) อัลกอริธึม SJF จะจะตรวจสอบเวลาที่ต้องใช้ซีพียูของกระบวนการที่กำลังดำเนินการที่ซีพียูว่าเหลือมากกว่าหรือน้อยกว่ากับเวลาของกระบวนการใหม่ที่เข้ามา ถ้าเหลือน้อยกว่าจะได้ดำเนินการในซีพียูต่อไป แต่ถ้าเหลือมากกว่าจะต้องออกการการทำงานที่ซีพียู เพื่อให้สิทธิ์แก่กระบวนการใหม่ที่เข้ามาได้เข้าทำงานที่ซีพียูแทน ดังตัวอย่างต่อไปนี้

มีกระบวนการ 4 กระบวนการ คือ P1 P2 P3 และ P4 แต่ละกระบวนการเข้ามาถึงระบบตามลำดับของเวลาที่กำหนด มีความต้องการใช้เวลาที่ซีพียูของแต่ละกระบวนการ มีหน่วยเป็นมิลลิวินาที (10^{-3} วินาที)

กระบวนการ	เวลาที่เข้าถึงระบบ	เวลาที่ต้องใช้ซีพียู
P1	0	8
P2	1	4
P3	2	9
P4	3	5

อัลกอริธึม SJF แบบให้สิทธิ์ก่อนจัดตารางเวลาการเข้าใช้ซีพียู ได้ดังต่อไปนี้



การจัดการของ SJF โดยที่กระบวนการ P1 เริ่มต้นดำเนินการที่ซีพียูเวลาที่ 0 เมื่อทำได้ 1 หน่วยเวลา กระบวนการ P2 เข้ามา ณ หน่วยเวลาที่ 1 ซึ่งต้องใช้เวลาที่ซีพียู 4 มิลลิวินาที แต่กระบวนการ P1 เหลือเวลาที่ต้องใช้ซีพียู อีก 7 มิลลิวินาที (เริ่มต้น 8 – ใช้แล้ว 1) ซึ่งเวลาที่ใช้เหลือมากกว่า P2 ดังนั้น SJF จะจัดการให้ P1 ออกจากการใช้ซีพียู และให้ P2 เข้าใช้แทน คือ P1 ให้สิทธิ์ P2 เมื่อ P2 ทำเสร็จจึงออกจากระบบไปในช่วงเวลาที่ 5 ซึ่ง ณ เวลานี้มีทุกกระบวนการเข้ามาแล้ว ขณะนี้เหลือกระบวนการ P1 P3 และ P4 อยู่ในระบบ SJF จะจัดการให้ P4 เข้าใช้ซีพียู เพราะ P4 เหลือเวลาที่ต้องใช้ซีพียู น้อยที่สุด เมื่อ P4 ทำงานเสร็จจึงออกจากระบบเหลือกระบวนการ P1 และ P3 ดังนั้น SJF จะจัดการให้ P1 และ P3 เข้าใช้ซีพียูตามลำดับเสร็จครบทุกกระบวนการ การหาเวลาการรอเฉลี่ยสำหรับตัวอย่างนี้คือ ((10-1)+(1-1)+(17-2)+(5-3))/4 = 26/4 = 6.5 มิลลิวินาที โดยใช้อัลกิริทึม SJF แบบให้สิทธิ์ก่อน แต่ถ้าใช้ SJF แบบไม่ให้สิทธิ์ก่อน จะได้ผลลัพธ์ของเวลาการรอเฉลี่ยคือ

((1-1)+ (8-1) + (17-2) +(12-3))/4 = 31/4 =7.75 มิลลิวินาที

และถ้าใช้อัลกอรทึม FCFS ได้เวลาการรอเฉลี่ยคือ

((0)+ (8-1) + (12-2) +(12-3))/4 = 35/4 =8.75 มิลลิวินาที

จะเห็นว่าแต่ละอัลกอลิทึมจะให้เวลารอเฉลี่ยที่ต่างกัน

การจัดตารางเวลาแบบลำดับความสำคัญ (Priority Scheduling)

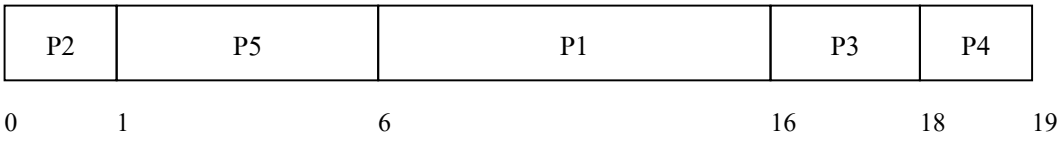
อัลกอริทึม SJF และ FCFS เป็นการจัดตารางเวลาการเข้าใช้ซีพียูที่ใช้การกำหนดความสำคัญแบบหนึ่งคือ SJF จะกำหนดให้กระบวนการที่ใช้เวลาซีพียูน้อยสุด มีความสำคัญสูงสุดสำหรับ FCFS จะกำหนดให้กระบวนการที่เข้ามารอใช้ซีพียูก่อนมีความสำคัญสูงสุด การจัดตารางเวลาที่ใช้ลำดับความสำคัญระบบปฏิบัติการจะจัดให้กระบวนการที่มีลำความสำคัญสูงกว่าได้เข้าใช้ซีพียูก่อน

อัลกอริธึมสำหรับการจัดตารางเวลาแบบลำดับความสำคัญมากได้ก่อน บางครั้งเรียกว่า Priority จะมีการกำหนดลำดับความสำคัญของกระบวนการโดยใช้ลำดับตัวเลข เช่น กำหนดลำดับความสำคัญเป็น 1 ถึง 9 ให้ 1 มีความสำคัญมากสุด รองลงมาเป็น 2 3 4 ตามลำดับไป หรือจะให้ 9 มีความสำคัญมากสุด รองลงมาเป็น 8 7 6 ตามลำดับไป หรือจะเป็นค่าตัวเลขช่วงอื่นๆก็ได้ แต่สำหรับการกำหนดลำดับความสำคัญของกระบวนการในที่นี้ จะกำหนดให้ 1 มีความสำคัญมากสุดและลำดับต่อไปจะมีความสำคัญรองลงมาต่อไป

ตัวอย่างการหาเวลาการรอเฉลี่ยโดยใช้อัลกอริธึม ลำดับความสำคัญพิจารณาชุดข้อมูลของกระบวนการ 5 กระบวนการ ซึ่งเข้ามาถึงระบบที่เวลา 0 เวลาที่ต้องใช้ซีพียู มีหน่วยเป็น มิลลิวินาที (10^{-3} วินาที) และลำดับความสำคัญของกระบวนการ ดังนี้

กระบวนการ	เวลาที่ต้องใช้ซีพียู	ลำดับความสำคัญ
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

การใช้อัลกอริธึม Priority จัดตารางเวลาให้กระบวนการเข้าใช้ซีพียูตามลำดับ ดังต่อไปนี้



เนื่องจากทุกกระบวนการเข้ามาขอใช้ซีพียูพร้อมกัน ณ เวลาที่ 0 อัลกอริธึม Priority จึงจัดตารางเวลาการเข้าทำงานตามลำดับความสำคัญองกระบวนการ คือ P2 P5 P1 P3 และ P4 เนื่องจาก P1 และ P3 มีลำดับ

ความสำคัญเท่ากันจึง FCFS P1จึงถูกเลือกก่อน P3 คำนวณหาเวลาที่รอเฉลี่ย คือ

(0+1+6+16+18)/5 = 8.2 มิลลิวินาที

ลำดับความสำคัญสามารถถูกกำหนดอย่างภายในหรืออย่างภายนอก การกำหนดแบบภายในกำหนดโดยระบบปฏิบัติการจะการวัดคุณสมบัติบางอย่างของกระบวนการที่สามารถวัดได้ เช่นเวลาการใช้ซีพียู หน่วยความจำ จำนวนไฟล์ จำนวน ไอ/โอ เป็นต้น ส่วนการกำหนดภายนอกคือการกำหนดลำดับความสำคัญของกระบวนการด้วยวิธีอื่นนอกเหนือจากระบบปฏิบัติการ ที่มาของกระบวนการ (แบทหรือ ออนไลน์) ค่าเช่าซีพียู เช่นค่าเช่าแพงกว่าจะมีความสำคัญมากกว่า

อัลกอริธึม Priority สามารถจัดตารางเวลาให้กระบวนการได้ทั้งแบบให้สิทธิ์ก่อน และแบบไม่ให้สิทธิ์ก่อน ซึ่งมีหลักการทำงานเช่นเดียวกับอัลกอริธึม SJF

ปัญหาที่จะต้องระมัดระวัง ของการจัดตารางเวลาให้กระบวนการโดยใช้อัลกอริธึม Priority คืออาจจะมีบางกระบวนการที่มีลำดับความสำคัญต่ำจะถูกปล่อยให้รออย่างไม่มีกำหนดจึงไม่ได้เข้าประมวลการแก้ปัญหานี้คือควรจะเพิ่มลำดับความสำคัญให้กับกระบวนการที่รออยู่นาน ๆ เมื่อถึงช่วงเวลาหนึ่งก็จะมีลำดับความสำคัญมากพอที่จะได้เข้าใช้ซีพียู

การจัดตารางเวลาแบบวนรอบ (Round-Robin Scheduling)

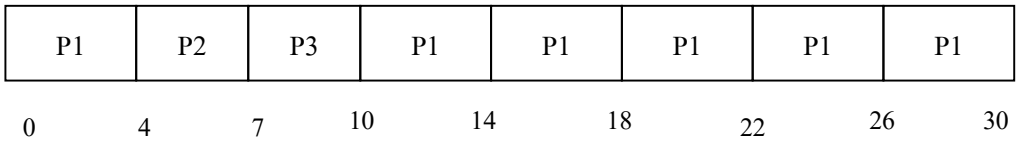
อัลกอริธึมการจัดตารางเวลาแบบวนรอบ หรือบางครั้งเรียกสั้น ๆ ว่า RR ได้ออกแบบเฉพาะให้กับระบบที่แบ่งเวลา ช่วงเวลาสั้นที่แบ่งให้เข้าใช้งานที่ซีพียู เรียกเวลาควอนตัม (Quantum Time) หรือชิ้นส่วนเวลา (Time Slice) การออกแบบได้กำหนด เวลาควอนตัมระหว่าง 10 ถึง 100 มิลลิวินาที การเข้าคิวตารางเวลาใช้ใช้ซีพียูจัดเป็นวงรอบ (Circular Queue) แต่ละกระบวนการได้รับการจัดสรรให้เข้าใช้ซีพียูรอบละ 1 เวลาควอนตัม ถ้ากระบวนการใดทำไม่เสร็จภายใน 1 เวลาควอนตัมนั้น กระบวนการนั้นจะถูกนำกลับไปเข้าคิวแบบวนรอบเพื่อเข้ามาทำต่อในรอบใหม่อีก ดำเนินการซ้ำ ๆ จนกระทั่งเสร็จ

การจัดเก็บของอัลกอริธึม RR จะจัดเก็บกระบวนการในคิวพร้อมแบบเข้าก่อนออกก่อน (FIFO Queue) กระบวนการที่เข้ามาใหม่จะนำไปใส่ต่อไว้ที่ท้ายคิว และกระบวนการที่อยู่หัวคิวจะถูกนำไปใช้ซีพียูก่อน เมื่อซีพียูว่างในการดำเนินการอาจจะมีบางกระบวนการใช้เวลาไม่ถึง 1 ควอนตัมเวลา ก็ทำงานเสร็จ กระบวนการนั้นจะออกจากวงรอบไปอัลกอริธึม RR จะเลือกกระบวนการที่หัวคิวเข้ามาใช้ซีพียูต่อ สำหรับกระบวนการที่ใช้เวลามากกว่า 1 เวลาควอนตัม จะเกิดการสับเปลี่ยนงาน (Context Switch) ทุก ๆ 1 เวลาควอนตัม จนกว่ากระบวนการนั้นจะเสร็จ

อัลกอริธึม RR จะจัดตารางเวลาการทำงานให้กับทุก ๆ กระบวนการได้เข้าใช้ซีพียูครั้งละ 1 เวลาควันตัม ดังตัวอย่าง มีกระบวนการ 3 กระบวนการมาถึงระบบ ณ เวลาที่ 0 และมีเวลาที่ต้องใช้ซีพียู (Burst Time) ของแต่ละกระบวนการมีหน่วยเป็น มิลลิวินาที (0^{-3} วินาที) ดังนี้

กระบวนการ	เวลาต้องใช้ซีพียู
P1	24
P2	3
P3	3

ถ้ากำหนดให้ 1 เวลาควอนตัมใช้ 4 Millisecond ดังนั้น กระบวนการ P1 ต้องใช้ถึง 6 เวลาควอนตัม P2 และ P3 ใช้ภายใน 1 เวลาควอนตัม จึงเสร็จงานในการทำงานอัลกอริธึม RR จะจัดให้ P1 P2 และ P3 สลับกันเข้าใช้ซีพียูครั้งละ1 เวลาควอนตัม ดังนี้เริ่มต้นให้ P1 ได้เข้าใช้ซีพียูก่อน 1 เวลาควอนตัมเป็นเวลา 4 มิลลิวินาที มันจะถูกบังคับให้ออกจากการใช้ซีพียู และให้สิทธิ์แก่กระบวนการถัดไปคือ P2 ซึ่งอยู่ในคิวถัดไป P2 ไม่ต้องการถึง 4 มิลลิวินาที มันออกจากการใช้ซีพียูก่อนเวลาควอนตัมของมันหมดอายุ ซีพียูจะถูกจัดสรรให้กระบวนการถัดไปคือ P3 แต่ละกระบวนการได้ใช้ซีพียู 1 เวลาควอนตัม จากนั้นซีพียูวนกลับมาจัดสรรให้ P1 เป็นเวลาควอนตัมที่ 2 จนกระทั่งทำงานเสร็จ อัลกอริธึม RR จัดตารางเวลา ได้ดังนี้



เวลาที่รอเฉลี่ยคือ 17/3 = 5.66 มิลลิวินาที

ในอัลกอริธึม RR จะไม่มีกระบวนการใดถูกจัดสรรซีพียูมากกว่า 1 เวลาควอนตัมสำหรับกระบวนการที่ต้องใช้เวลาเกิน 1 เวลาควอนตัมซึ่ง กระบวนการนั้นจะให้สิทธิ์กระบวนการอื่น และกลับเข้าไปในคิวพร้อมเพื่อรอเวลาวงรอบต่อไป ถ้ามี n กระบวนการในคิวพร้อม และเวลาควอนตัมคือ q แล้วแต่ละกระบวนการจะรอไม่นานกว่า (n-1)*q หน่วยเวลาควอนตัม ก่อนที่จะได้รับการจัดสรรให้เข้าใช้ซีพียูอีกครั้ง ตัวอย่างเช่น ถ้ามี 5 กระบวนการและมีควอนตัมเวลา 20 มิลลิวินาที ดังนั้นแต่ละกระบวนการจะรอไม่เกิน 80 มิลลิวินาที ที่จะได้รับการจัดสรรให้เข้าใช้ซีพียูอีกครั้ง

ประสิทธิภาพของ RR การวนรอบยึดหลักขนาดของเวลาควอนตัม ถ้าเวลาควอนตัมใหญ่ ผลการทำงานจะเหมือนกับ FCFS ถ้าเวลาควอนตัมเล็กเกินไป ผลการทำงานจะช้าลงเพราะระบบจะเสียเวลาส่วนหนึ่งในการดำเนินการกับกระบวนการเข้า-ออกซีพียู เช่นการคัดเลือกกระบวนการที่พร้อมเข้าใช้ซีพียู การสับเปลี่ยนกระบวนการจากซีพียู กระบวนการที่ให้สิทธิ์ต้องบันทึกสถานะการทำงานต่าง ๆ ไว้ ในบล็อกควบคุมกระบวนการ และจะต้องอ่านสถานะการทำงานต่าง ๆ ของกระบวนการใหม่ขึ้นมาเพื่อทำงาน ดังนั้นถ้า RR กำหนดขนาดของเวลาควันตั้มน้อยมากเกินไปอาจจะเหลือเวลาที่ใช้ประมวลผลที่ซีพียู เท่ากับเวลาที่ใช้จัดการสับเปลี่ยนกระบวนการ

ตัวอย่างเช่น มีกระบวนการ 1 กระบวนการที่ต้องการใช้เวลา 10 Millisecond ในการพิจารณาผลกระทบ การสับเปลี่ยนงาน ของ อัลกอริธึม RR ดังนี้

ถ้ากำหนดเวลาควอนตัมคือ 12 Millisecond กระบวนการจะทำเสร็จก่อนครบ เวลาควอตัม ไม่มีการสับเปลี่ยนงาน (ไม่เสียเวลาส่วนนี้)

ถ้ากำหนดเวลาควอนตัมคือ 6 Millisecond กระบวนการจะใช้ 2 เวลาควอตัม ต้องมีการสับเปลี่ยนงาน 1 ครั้ง (เสียเวลาส่วนนี้ 1 ครั้ง)

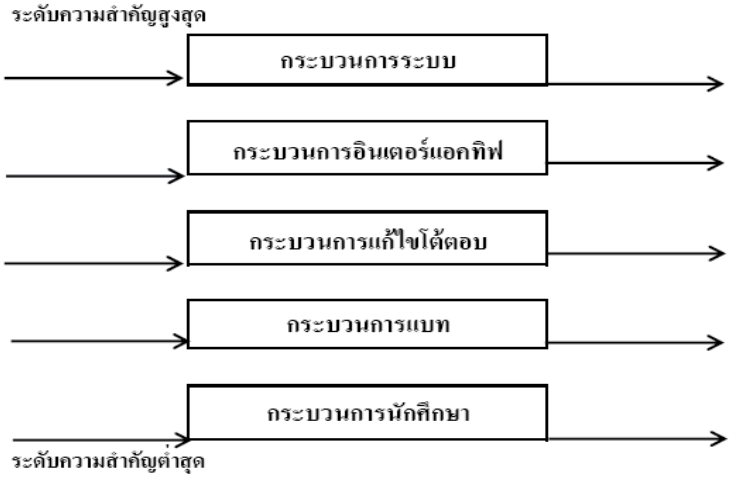
ถ้ากำหนดเวลาควอนตัมคือ 1 Millisecond กระบวนการจะใช้ 10 เวลาควอตัม ต้องมีการสับเปลี่ยนงาน 9 ครั้ง (เสียเวลาส่วนนี้ 9 ครั้ง) ซึ่งจะทำให้กระบวนการเสร็จงานช้าเพราะต้องเสียเวลาในการสับเปลี่ยนงานส่วนนี้ด้วย

โดยทั่วไปยังไม่ทีการคำนวณหาเวลาที่เสียไปจากการสับเปลี่ยนงาน ในการทำงานที่รวมเวลาของการสับเปลี่ยนงานแล้ว การลดเวลาควอนตัมลงในระดับหนึ่งจะทำให้เวลาที่วนรอบมากขึ้น และถ้าให้ขนาดเวลาควอนตัมใหญ่มากจะเป็นการทำงานแบบ FCFS แต่ในทางปฏิบัติโดยทั่วไปจะรู้ว่าแต่ละกระบวนการจะใช้เวลาซีพียูจำนวนเท่าไร จึงสามารถกำหนดขนาดของเวลาควอนตัมไว้ที่ ร้อยละ 80 ของเวลาที่ใช้ซีพียู

การจัดตารางเวลาแบบคิวหลายระดับ (Multilevel Queue Scheduling)

อัลกอริธึมการจัดตารางเวลาแบบคิวหลายระดับได้สร้างขึ้มาสำหรับระบบงานที่มีการแบ่งการทำงานอย่างชัดเจนเช่นงานเบื้องหน้า (Foreground) คืองานที่ทั่วไปที่ทำบ่อย ๆ หรืองานที่ต้องตอบสนองทันที (Interactive) และงานเบื้องหลัง (Background) คือพวกงานแบท (Batch) ต่าง ๆ กระบวนการทั้ง 2 ชนิดเหล่านี้มีความต้องการเวลาตอบสนอง (Response-Time) ที่แตกต่างกัน ดังนั้นอาจจะมีความต้องการการจัดตารางเวลาของงานที่แตกต่างกัน โดยทั่วไปกระบวนการเบื้องหน้า (Foreground Processes) จะมีระดับความสำคัญมากกว่ากระบวนการเบื้องหลัง (Background Processes)

อัลกอริธึมการจัดตารางเวลาแบบคิวหลายระดับ จะแบ่งคิวต่าง ๆ ในคิวพร้อม ออกเป็นหลายระดับแต่ละระดับ มีความสำคัญที่แตกต่างกันซึ่งสมารถแบ่งได้หลายลักษณะเช่น แบ่งตามขนาดของกระบวนการ ขนาดหน่วยความจำที่ใช้ หรือจำนวน ไอ/โอ โดยแต่ละคิวจะใช้อัลกอริธึมในการจัดตารางตรงเวลาที่ต่างกัน เช่น งานเบื้องหน้าอาจจะใช้ การจัดตารางเวลาแบบ RR สำหรับงานเบื้องหลัง อาจจะใช้การจัดตารางเวลาแบบ FCFS และยังสามารถจัดตารางเวลาระหว่างคิวแต่ละระดับได้ด้วย เช่นจัดตารางเวลาของคิวคงที่ของแต่ละระดับ หรือจัดตารางเวลาของคิวที่ให้เปลี่ยนละระดับได้ ตัวอย่างการจัดคิวแบบหลายระดับแบบคงที่ มี 5 คิว ดังรูป 4.5



รูปที่ 4.5 แสดงการจัดตารางแบบแถวคอยหลายระดับ

1. คิวของกระบวนการระบบ
2. คิวของกระบวนการอินเตอร์แอคทีฟ
3. คิวของกระบวนการการแก้ไขอินเตอร์แอคทีฟ
4. คิวของกระบวนการแบท
5. คิวของกระบวนการนักศึกษา

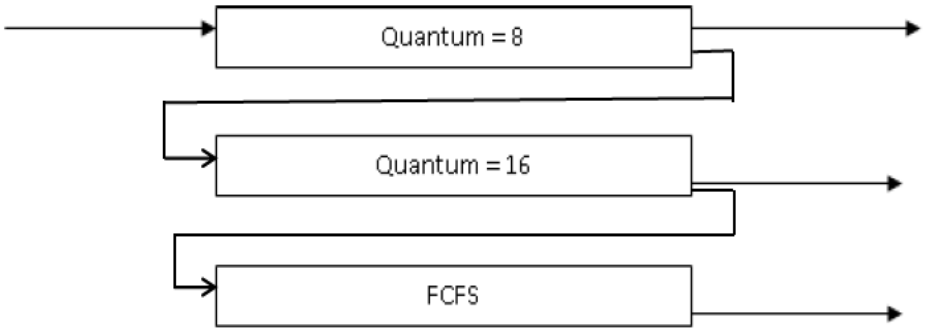
จากรูปที่ 4.5 คิวที่อยู่ด้านบนมีความสำคัญสูงกว่าคิวที่อยู่ด้านล่าง ดังนั้นกระบวนการที่อยู่ในคิวที่มีความสำคัญสูงกว่าจะได้รับการจัดสรรให้เข้าใช้ซีพียูก่อน สำหรับอัลกอริทึมนี้สนับสนุนการให้สิทธิ์ก่อนด้วย เช่น ถ้ากระบวนการกำลังใช้งานซีพียู แต่มีกระบวนการใหม่เข้ามาในคิวที่มีระดับความสำคัญมากกว่ากระบวนการนี้จะต้องให้สิทธิ์การใช้ซีพียู แก่กระบวนการใหม่ อย่างไรก็ตามจะต้องไม่มีปัญหา

เรื่องการรอของกระบวนการที่มีระดับความสำคัญต่ำ เพราะกระบวนการที่มีระดับความสำคัญสูงมีมาก จึงได้ซีพียู ตลอดทำให้ไม่มีคิวสำหรับกระบวนการที่มีระดับความสำคัญต่ำ แนวทางจัดการอาจจะต้องมีการแบ่งสัดส่วนเวลาการใช้ซีพียู ให้กับแต่ละคิว เช่น ร้อละ 80 แบ่งให้กระบวนการเบื้องหน้า และใช้การจัดตารางเวลาแบบ RR และร้อยละ 20 กระบวนการเบื้องหลังและใช้การจัดตารางเวลาแบบ FCFS

การจัดตารางเวลาแบบคิวหลายระดับที่ปรับได้ (Multilevel Feedback Queue Scheduling)

ปรกติในการจัดตารางเวลาแบบคิวหลายระดับจะถูกกำหนดคิวไว้ถาวรในแต่ระดับ ซึ่งไม่สามารถเคลื่อนย้ายคิวระหว่างระดับได้ แต่ในการทำงานของระบบบางครั้งต้องการความยืดหยุ่นสำหรับการจัดการคิวระหว่างระดับ โดยใช้วิธีการจัดตารางเวลาแบบคิวหลายระดับที่ปรับได้ คือ กระบวนการที่อยู่ในแต่ละคิวสามารถปรับระดับความสำคัญขึ้นหรือลงระหว่างระดับได้ แนวความคิดนี้ส่วนใหญ่แบ่งระดับความสำคัญโดยใช้เขตซีพียู ถ้ากระบวนการใช้เวลาซีพียูมากเกินไปมันจะถูกเคลื่อนย้ายไปสู่คิวที่มีระดับความสำคัญต่ำลงมาได้ จะทำให้กระบวนการที่เป็นเขต ไอ/โอ หรือกระบวนการที่เป็นพวกโต้ตอบมาก ๆ มีโอกาสได้เข้าไปอยู่ในคิวที่มีระดับความสำคัญสูงขึ้น ซึ่งเป็นการป้องกันไม่ให้กระบวนการที่มีระดับความสำคัญต่ำต้องรออยู่ในคิวนานเกินไป

ยกตัวอย่างการจัดตารางเวลาแบบคิวหลายระดับที่ปรับได้มีคิว 3 คิว ดังรูปที่ 4.6 เมื่อมีกระบวนการเข้ามาในคิวแรก จะได้รับการจัดสรรให้ได้ใช้ซีพียู 1 เวลาควอนตัม 8 หน่วยเวลา ถ้ากระบวนการนี้ทำไม่เสร็จในช่วง 1 เวลาควอนตัม มันจะถูกลดระดับความสำคัญให้ไปอยู่ในคิวของระดับต่ำลงไปอีก 1 ระดับคือคิวที่ 2 กำหนดเวลาควอนตัม 16 หน่วยเวลา แต่ถ้าทำในคิวที่ 2 ไม่เสร็จในช่วง 1 เวลาควอนตัมอีก มันจะถูกลดระดับความสำคัญให้ไปอยู่ในคิวของ FCFS ถึงแม้เมื่อมันอยู่ในคิว FCFS แล้วมันยังมีโอกาสที่จะได้ไปอยู่คิวด้วนบนได้ ซึ่งการทำงานแบบนี้จะให้ความสำคัญกับกระบวนการที่ใช้เวลาซีพียูไม่เกิน 8 หน่วยเวลา ส่วนกระบวนการที่ใช้เวลาซีพียู เกิน 8 แต่ไม่เกิน 24 หน่วยเวลาจะปรับลงมาอยู่ใน คิว 2 และ FCFS



รูปที่ 4.6 การจัดตารางเวลาแบบคิวหลายระดับที่ปรับได้

โดยทั่วไปการจัดตารางเวลาแบบคิวหลายระดับที่ปรับได้ ถูกกำหนดโดยตัวแปรดังต่อไปนี้

1. จำนวนของคิว
2. อัลกอริธึมการจัดตารางเวลา สำหรับแต่ละคิว
3. วิธีการเพิ่มระดับความสำคัญของคิวกระบวนการ
4. วิธีการลดระดับความสำคัญของคิวกระบวนการ
5. วิธีการนำเอา กระบวนการที่ต้องการใช้ซีพียูเข้ามาในคิว

การจัดตารางเวลาแบบคิวหลายระดับที่ปรับได้เป็นที่นิยมใช้ เพราะสามารถปรับระดับคิวของแต่ละกระบวนการได้เหมาะสมกับการควบคุมการทำงานของคอมพิวเตอร์ แต่ยังเป็นการทำงานที่ซับซ้อนมาก

การจัดตารางเวลาของมัลติโปรเซสเซอร์

สำหรับการจัดตารางเวลาการใช้ซีพียู ที่กล่าวมาข้างต้นจะเป็นลักษณะของระบบที่มีโปรเซสเซอร์เดียว แต่ถ้าจะจัดตารางเวลาการใช้งานหลาย ๆ ซีพียู ระบบปฏิบัติจะต้องมีความซับซ้อนมากขึ้น ซึ่งได้มีการทดลองมาเป็นนานหลาย ๆ วิธี ยังไม่มีการสรุปว่าวิธีไหนแบบไหนดีที่สุด แต่ยังมีวิธีที่พอเป็นไปได้ และมีการใช้กันแพร่หลายสำหรับในที่นี้จะกล่าวถึงแนวทางเท่านั้น

การจัดตารางเวลาของมัลติโปรเซสเซอร์ (Multiple Processor Scheduling) ปัจจัยหลักที่ต้องคำนึงถึง คือชนิดของซีพียู หรือโปรเซสเซอร์ ซึ่งอาจจะเป็นระบบชนิดเดียวกัน (Homogeneous System) หรือระบบต่างชนิดกัน (Heterogeneous System) ถ้าโปรเซสเซอร์ต่างชนิดกันจะมีข้อกำหนดในการจัดการที่ต่างกันแต่ละโปรเซสเซอร์จะมีคิวของมันเองและมีอัลกอริธึมการจัดตารางเวลาการใช้ซีพียูของมันเอง กระบวนการที่จะเข้าประมวลผลที่ซีพียูจะต้องเป็นกระบวนการที่ทำงานบนโครงสร้างนั้น

ได้ สามารถสื่อสารกันได้ การประมวลผลมันจะทำได้บนครื่องประเภทเดียวกันเท่านั้น เช่นโปรแกรมที่พัฒนาขึ้นมาบนเครื่อง CDC จะประมวลผลที่เจาะจงเฉพาะบนเครื่อง CDC เท่านั้น ไม่สามารถนำไปประมวลผลบนเครื่อง IBM ได้ ดังนั้นแต่ละโปรเซสเซอร์สามารถกำหนดตารางเวลาด้วยตัวเอง

ถ้าโปรเซสเซอร์เหมือนกันมันจะสามารถดำเนินการกระบวนการจำนวนมากมาย โดยให้แต่ละโปรเซสเซอร์โหลดกระบวนเข้ามาจัดคิวสำหรับแต่ละโปรเซสเซอร์เอง ในการจัดการแบบนี้อาจจะมีบางโปรเซสเซอร์ ว่างไม่มีคิวของกระบวนการเข้ามาประมวลผลเลย ในขณะที่โปรเซสเซอร์อื่น ๆ มีคิวของกระบวนการเข้ามามากมาย เพื่อป้องกันสถานการณ์แบบนี้ เราจะใช้วิธีการให้กระบวนการทั้งหมดโหลดเข้ามาในคิวพร้อมที่เป็นส่วนกลางร่วมกันจากนั้นจึงให้มี โปรเซสเซอร์หนึ่งเครื่องที่ทำหน้าที่ในการจัดตารางเวลาคิวต่าง ๆ เหล่านั้นแจกไปยังแต่โปรเซสเซอร์ที่มีอยู่ การจัดการแบบนี้เรียกว่าจัดโครงสร้างแบบนายกับบ่าว (Master-Slave)

ในรูปแบบนั้นการจัดตารางเวลาที่เป็นแบบหลายโปรเซสเซอร์ ในคอมพิวเตอร์บางระบบมีการพัฒนาโครงสร้างให้มีซีพียู แยกออกมาระหว่างงานในระบบเช่น ให้มีโปรเซสเซอร์สำหรับจัดการเวลาและจัดการระบบอื่น ๆ แยกออกจากงานไอ/โอ ต่าง ๆ และงานที่เป็นของผู้ใช้ ซึ่งการแยกโปรเซสเซอร์ทำงานนี้ทำให้ระบบปฏิบัติการมีความซับซ้อนน้อยลงเพราะจะมีซีพียูเพียงตัวเดียวที่ใช้ข้อมูลของระบบปฏิบัติการสำหรับการควบคุม ส่วนโปรเซสเซอร์ตัวอื่น ๆ จะทำหน้าที่ในการเข้าถึงเฉพาะผู้ใช้ภายนอกเท่านั้น ทำให้ระบบทำงานได้มีประสิธิภาพมากขึ้น

การวัดประเมินอัลกอริธึม

จากอัลกอริธึมการจัดตารางเวลาต่าง ๆ ที่กล่าวมาข้างต้นซึ่งมีหลายวิธี อย่างไรก็ตามการเลือกอัลกิริทึมการจัดตารางเวลาของการเข้าใช้ซีพียู มันจะต้องเหมาะสมเฉพาะกับแต่ละระบบ นั่นคือแต่ละอัลกิริทึมมีตัวแปรมากมายที่ต่างกันทำให้การเลือกใช้อัลกิริทึมที่ดีที่สุดนั้นทำได้ยาก

ปัญหาแรกที่ต้องคำนึงถึงคือการกำหนดกฎเกณฑ์เพื่อใช้ในการเลือกอัลกิริทึม ซึ่งเราจะพิจารณาตามข้อที่ควรพิจารณาเพื่อใช้ในการเลือกใช้อัลกอริธึมที่ดีที่สุด โดยมีกฎเกณฑ์ที่ใช้พิจาณา คือ การใช้ประโยชน์ซีพียู (CPU Utilization) ผลผลิต (Throughput) เวลาทั้งหมด (Turnaround Time) เวลารอ (Waiting Time) เวลาตอบสนอง (Response Time) เพื่อเลือกอัลกิริทึมที่ดีให้ความสำคัญของข้อกำหนดที่เกี่ยวข้องกฎเกณฑ์เหล่านี้ นอกจากนี้อาจจะรวมถึงการวัดอื่นที่เกี่ยวข้อง เช่น

- เพิ่มการใช้ประโยชน์ซีพียูให้ได้มากที่สุด ภายใต้ข้อจำกัดเวลา เวลาสนองกลับต้องน้อยที่สุด โดยใช้เวลาสูงที่สุดคือไม่เกิน 1 วินาที หรือ

- เพิ่มผลลัพธ์จากการทำงานให้ได้มากที่สุด นั่นคือเวลาที่วนรอบเป็นสัดส่วนโดยตรงกับเวลาที่ใช้ประมวลผลอย่างพอดี

การกำหนดหลักเกณฑ์ในการพิจารณาอัลกอริธึมที่ดีที่สุดนั้นเป็นเรื่องยาก เพราะเราไม่ทราบแน่นอนว่าความต้องการที่แท้จริงทั้งหมดของคอมพิวเตอร์แต่ละระบบเป็นอย่างไร สิ่งที่เราทำได้คือสมมุติความต้องการของคอมพิวเตอร์ตามที่เราพอจะทราบได้ และกำหนดตามเกณฑ์ที่ใช้ในการพิจารณา จากนั้นจึงมาเรียนรู้ว่ามีวิธีการในการเปรียบเทียบในการอัลกอริธึมที่ดีที่สุด ซึ่งบางครั้งอาจจะมีบางปัจจัยที่เรามองข้ามไปก็ได้

การสร้างแบบจำลองตามที่กำหนด (Deterministic Modeling)

การเลือกใช้อัลกอริธึม จะใช้วิธีการประเมิลผลจาการวิเคราะห์ (Analytic Evaluation) คือจะนำอัลกอริธึมต่าง ๆ และรายละเอียดของงาน มาทำการคำนวณหาค่าเชิงตัวเลขจากผลการทำงานของแต่ละ

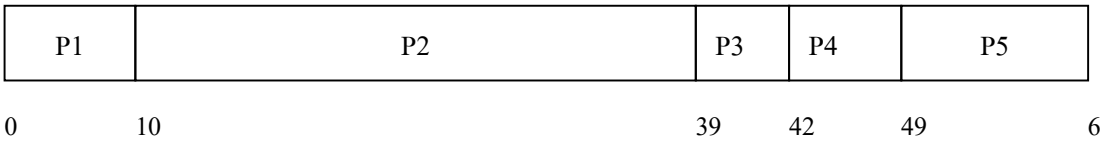
อัลกอริธึมเพื่อนำมาเปรียบเทียบกัน

ตัวอย่างเช่น สมมุติเรามีงานที่ต้องทำในระบบ 5 กระบวนการโดยทุกกระบวนการเข้ามาถึง ณ เวลาที่ 0 ตามลำดับ และมีข้อมูลเวลาที่ต้องใช้ซีพียูของแต่ละกระบวนการ มีหน่วยเวลาเป็นมิลลิวินาที (10^{-3} วินาที) ดังนี้

กระบวนการ	เวลาที่ต้องใช้ซีพียู
P1	10
P2	29
P3	3
P4	7
P5	12

จะทำการพิจารณาอัลกอริธึม 3 แบบคือ FCFS SJF และ RR (กำหนดเวลาควอนตัม = 10 มิลลิวินาที) จะทำการพิจารณาว่าอัลกอริธึมใด จะให้เวลารอเฉลี่ยน้อยที่สุด ดังต่อไปนี้

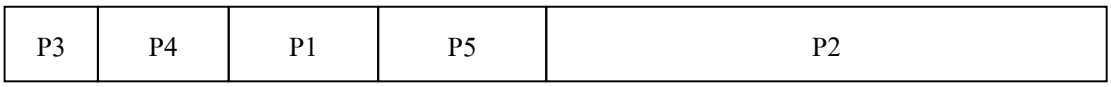
สำหรับ วิธีคิด FCFS จัดตารางเวลาการประมวลผลคือ



เวลาที่รอสำหรับแต่ละกระบวนการคือ P1 รอ 0 มิลลิวินาที P2 รอ 10 มิลลิวินาที P3 รอ 39 มิลลิวินาที P4 รอ 42 มิลลิวินาที และ P5 รอ 49 มิลลิวินาที

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

สำหรับวิธีคิดแบบ SJF แบบไม่ให้สิทธิ์ก่อน จัดตารางเวลาการประมวลผลคือ



0 3 10 20 32 61

เวลาที่รอของแต่ละกระบวนการคือ คือ P1 รอ 10 มิลลิวินาที P2 รอ 32 มิลลิวินาที P3 รอ 0 มิลลิวินาที P4 รอ 3 มิลลิวินาที และ P5 รอ 20 มิลลิวินาที

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

สำหรับวิธีแบบ RR เริ่มต้น P1 ดำเนินการ 1 เวลาควอนตัม 10 มิลลิวินาทีแล้วให้สิทธิ์ P2 ดำเนินการต่อ P1 จะถูกนำไปใส่ข้างหลังสุดของคิว ดังนี้

0 10 20 23 30 40 50 52 61

เวลาที่รอของแต่ละกระบวนการคือ P1 รอ 0 มิลลิวินาที P2 รอ 32 มิลลิวินาที P3 รอ 20 มิลลิวินาที P 4 รอ 23 มิลลิวินาที และP5 รอ 40 มิลลิวินาที

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

จากผลการคำนวณเวลาที่รอโดยเฉลี่ยของแต่ละอัลกิริทึม จะเห็นได้ว่าผลลัพธ์ของวิธี SJF น้อยกว่าครึ่งหนึ่งของเวลาที่รอเฉลี่ยของ FCFS ส่วน RR ให้ค่าปานกลาง

การเลือกอัลกอริธึมด้วยวิธีการสร้างแบบจำลองตามที่กำหนด เป็นวิธีที่ง่ายและเร็ว ผลลัพธ์ที่ได้มีค่าเป็นตัวเลขที่แน่นอนตามอัลกอริธึม ที่ใช้ในการเปรียบเทียบ อย่างไรก็ตามผลลัพธ์อาจจะไม่เป็นจริงเสมอไปสำหรับการทำงานจริง เนื่องจากอินพุตเป็นเพียงข้อมูลสมมุติ และมีจำนวนน้อย ในการทำงานจริงข้อมูลเล่านี้ไม่สามารถประเมินแทนข้อมูลจริงได้ แต่สำหรับระบบที่ไม่ยุ่งยากซับว้อนมาก การคำนวณจากข้อมูล หลาย ๆ ครั้ง อาจจะทำให้พอจะประเมินได้ว่าจะเลือกใช้อัลกอริธึมไหน

แบบจำลองคิว (Queuing Models)

กระบวนการที่เข้ามาขอใช้ซีพียูมีมากมายมากในแต่ละวัน แต่ละงานจะมีคุณสมบัติที่แตกต่างกัน เราอาจจะสามารถดาคการณ์การทำงานของกระบวนการนั้น ๆ จากการกระจายเวลาการเข้าใช้ซีพียู หรือการใช้ ไอ/โอ ซึ่งสามารถนำมาใช้ในการออกแบบการสร้างคิวได้โดย ผลลัพธ์ออกมาได้สูตรหาค่าทางคณิตศาสตร์ การอ้างถึงความเป็นได้ของการกระจายเวลาการใช้ซีพียู เช่นดียวกับการกระจายเวลาเมื่อกระบวนการเข้ามาถึงในระบบ สิ่งแหล่านี้มันสามารถคำนวณหาระดับค่าเฉลี่ยของการใช้ประโยขน์ซีพียู ผลลัพธ์ของการทำงาน เวลารอเฉลี่ย และอื่น ๆ ของอัลกอริธึมทั้งหมด

สำหรับระบบคอมพิวเตอร์ที่เป็นแบบเครือข่ายของเครื่องเซิฟต์เวอร์ แต่ละเครื่องมีการจัดคิวของ กระบวนการในการขอใช้ซีพียูที่เครื่องเซิฟต์เวอร์เอง มีคิวพร้อมของมีคิวของระบบไอ/โอ และคิวอุปกรณ์ต่าง ๆ ของมันเอง การรู้อัตราการใช้บริการ (Service Rates) แต่ละโหนดของเครือข่าย และรู้เวลาที่งานต่าง ๆ ในคิวต้องใช้ เราสามารถคำนวณหาค่าที่ต้องการต่าง ๆ ได้เช่น การใช้ประโยชน์ (Utilization) ความยาวเฉลี่ยของคิว เวลารอเฉลี่ยและอื่น ๆ เราเรียกว่า การวิเคราะห์การเข้าคิวเครือข่าย (Queuing-etwork Analysis)

ตัวอย่างสมมุติให้ N คือความยาว คิวเฉลี่ย ยกเว้นกระบวนการที่กำลังดำเนินการอยู่ ให้ W คือเวลาที่รอเฉลี่ยในคิว และให้ λ คือแวลาฉลี่ยสำหรับ กระบวนการใหม่ที่เข้ามาในคิว เช่น 3 กระบวนการ ต่อวินาที เราสามารถคำนวณหาได้ว่าเวลาที่กระบวนการรอ W มีงานใหม่เข้ามาเฉลี่ยจำนวน λ จะได้ เวลาซึ่งกระบวนการรอในคิวคือ λ * เวลาที่รอเฉลี่ย W ถ้ามีกระบวนการใหม่จะเข้ามาในคิวระบบสม่ำเสมอแล้วจำนวนของกระบวนการที่ออกจากคิวระบบสม่ำเสมอเท่ากับจำนวนของกระบวนการ ที่เข้ามาด้วยจะได้ $N = \lambda * W$

สมการนี้เรียกว่า สูตรของลิตเทิล (Little's Formula) ซึ่งเป็นประโยชน์มากเพราะสามารถใช้กับอัลกอริธึมใด ๆ ก็ได้ และเราสามารถใช้สูตรนี้เพื่อคำนวณ 1 ใน 3 ตัวแปร ถ้าเรารู้ 2 ตัวที่เหลือ ยกตัวอย่างเช่น ถ้าเรารู้ว่า 7 กระบวนการ มาถึงทุก ๆ นาที (โดยเฉลี่ย) และมี 14 กระบวนการอยู่ในคิว แล้วเราสามารถคำนวณเวลาที่รอเฉลี่ยต่อกระบวนการได้ เป็น 2 วินาที

การวิเคราะห์คิวเป็นประโยชน์ในการเปรียบเทียบการอัลกอริธึมการจัดตารางเวลาได้อย่างมาก แต่มันยังมีขีดจำกัด ที่ไม่สามารถคำนวณระดับชั้นของอัลกอริธึมได้ และปัญหาการหาค่าเฉลี่ยของระบบการจายต่าง ๆ เนื่องจากระบบค่อนข้างซับซ้อน ดังนั้นการหาค่าเฉลี่ยสำหรับระบบการกระจายอาจจะให้ผลที่ไม่ถูกต้องนักเป็นเพียงค่าประมาณเท่านั้น

การจำลองระบบ

เพื่อให้ได้การประเมินว่าอัลกอริธึมการจัดตารางเวลาใดดีกว่ากัน เราสามารถใช้วิธีการจำลองระบบ (Simulations) ซึ่งวิธีนี้จะสามารถนำไปสู่การเขียนโปรแกรมแบบจำลองของระบบต่าง ๆ ในคอมพิวเตอร์ได้ โดยมีโครงสร้างข้อมูลซอฟต์แวร์ การแสดงส่วนประกอบหลักของระบบการจำลอง มีการแสดงตัวแปรและค่าของตัวแปรนี้ถูกเพิ่มเข้าไปในระบบการจำลองเพื่อแก้ไขสถานะการทำงานของระบบ (System State) รวมทั้งประมวลผลสถิติการบ่งบอกถึงขั้นตอนวิธี (Indicate Algorithm) ต่าง ๆ ทั้งแสดงผลให้เห็นผลการทำงานและเก็บบันทึกไว้เป็นข้อมูลสำหรับการพัฒนาต่อไป

ข้อมูลการสำหรับการจำลองระบบ สามารถได้มาหลายวิธี โดยทั่วไปได้มาจากการสุ่มจากการทำงานของคอมพิวเตอร์เองโดยใช้ข้อมูลของกระบวนการ, เวลาการใช้ซีพียู การเข้า-ออกระบบ และการกระจายการทำงานตามโหนดต่าง ๆ ความน่าจะเป็นของการประมวลผลแบบกระจายอาจจะถูกคำนวณให้ค่าในเชิงคณิตศาสตร์ หรือค่าจากการสังเกต ถ้าการประมวลผลแบบกระจายกำหนดค่าเชิงประจักษ์ของระบบตามความเป็นจริงโดยดูจากผลลัพธ์ ดังนั้นจากข้อมูลตามความเป็นจริงของเหตุการณ์ในระบบจริงของระบบกระจายนี้สามารถถูกใช้เพื่อการสร้างแบบจำลองระบบได้

อย่างไรก็ตามการตัวขับเคลื่อนการจำลองระบบแบบกระจาย อาจจะมีความคลาดเคลื่อนเนื่องจากความสัมพันธ์ระหว่างเหตุการณ์ในระบบการกระจายไม่ต่อเนื่องกัน การแสดงความถี่ของเหตุการณ์ซึ่งมีมากมายไม่สามารถแสดงได้ทั้งหมด อาจจะแก้ปัญหาโดยการใช้เทปบันทึกร่องรอยของเหตุการณ์ต่าง ๆ ตามความเป็นจริงไว้ ทั้งหมดเพื่อนำไปใช้ได้ต่อไป

การสร้างแบบจำลองระบบเพื่อเปรียบเทียบอัลกอริธึมต้องใช้เวลามาก เพราะต้องใช้ข้อมูลสำหรับการจำลองระบบที่มีรายละเอียดมาก และขึ้นกับผู้พัฒนาแบบจำลองระบบถ้าจะสร้างแบบจำลองระบบที่มีรายละเอียดมากก็จะเสียค่าใช้จ่ายสูงมากและเสียเวลามาก ซึ่งต้องพิจาณาถึงความคุ้มค่าควบคู่กัน

การดำเนินงานจริง

การสร้างแบบจำลองระบบเป็นวิธีการเปรียบเทียบอัลกอริธึมการจัดตารางเวลาได้อย่างดี แต่ต้องนำมันไปใช้ในระบบปฏิบัติการจริงเพื่อให้เห็นการทำงานจริง การวัดประเมินจริงภายใต้เงื่อนไขที่กระทำจริง ๆ ของอัลกอริธึม

อัลกอริธึมการประเมินคือประเมินสภาพแวดล้อมที่จะใช้เปลี่ยนแปลงในการเขียนโปรแกรมใหม่ และชนิดของปัญหาการแลกเปลี่ยนสภาพแวดล้อม ซึ่งจะมีตัวแปรหรือปัจจัยของสภาพแวดล้อม

ค่อนข้างมาก เป็นเหตุผลหนึ่งที่ทำให้วิธีนี้ไม่นิยมใช้ นอกจากความยุ่งยากซับซ้อนแล้วยัง ต้องเขียนโปรแกรมสำหรับประเมินอัลกอริธึมแต่ละแบบ ซึ่งต้องใช้เวลาและค่าใช้จ่ายสูงมาก หรือบางอัลกอริธึมอาจจะต้องหาฮาร์ดแวร์ใหม่

ยกตัวอย่างเช่น ระบบรายงานโดยอัตโนมัติที่ออกแบบทดลองในการแยกประเภทกระบวนการแบบไม่โต้ตอบอัตโนมัติ (Noninteractive) โดยดูที่ผลรวมของไอ/โอ ถ้ากระบวนการไม่นำเข้าหรือส่งออกในช่วงเวลา 1 วินาที มันจะให้สิทธิงานไม่โต้ตอบอัตโนมัติ และมันถูกเคลื่อนย้ายสู่คิวที่มีระดับความสำคัญต่ำกว่า และงานไม่โต้ตอบอัตโนมัติคือโปรแกรมเมอร์ทำการพิมพ์ตัวอักษรใด ๆ ในช่วงเวลาน้อยกว่า 1 วินาที ผลการทำงานระบบให้ลำดับความสำคัญสูงกับโปรแกรมของเขา จึงได้ผลที่แสดงออกจากการพิมพ์อักษรใด ๆ ของโปรแกรมเมอร์

ความยืดหยุ่นสำหรับอัลกอริธึมการจัดตารางเวลา คือคอมพิวเตอร์สามารถปรับเปลี่ยนอัลกอริธึมโดยผู้ควบคุมระบบได้ ซึ่งอาจจะปรับเปลี่ยนในขณะทำงาน หรือช่วงเปิดเครื่องเท่านั้น โดยทั่วไปการทำให้คอมพิวเตอร์ใด ๆ สามารถปรับเปลี่ยนอัลกอริธึมได้ เป็นสิ่งที่ต้องการของคอมพิวเตอร์ทุกระบบ เพราะในแต่ละวันมีงานจำนวนมากที่ต้องการระบบปฏิบัติการที่แตกต่างกันมาก แต่ปัจจุบันมีระบบคอมพิวเตอร์ที่สามารถปรับเปลี่ยนระบบปฏิบัติการได้ยังมีน้อย

ปฏิบัติการที่ 4

1. ให้นักศึกษาเขียนโปรแกรมการหาเวลาการรอเฉลี่ยและเวลาการใช้เฉลี่ยของแต่ละกระบวนการ โดยใช้ข้อมูลจากแบบฝึกหัดข้อ 6 หรือ 7 หรือ 8 ให้ใช้อัลกอริธึม FCFS, SJF แบบ preemptive, SJF แบบ non-preemptive, Priority แบบ preemptive , Priority แบบ non-preemptive และแบบ RR กลุ่มละ 1 อัลกอริธึม โดยวิธีการจับฉลากเลือกอัลกอริธึม หรือวิธีการตกลงกัน
2. ทุกกลุ่มนำผลการทำงานของแต่ละโปแกรมมานำเสนอ แลกเปลี่ยนเรียนรู้กัน

คำถามท้ายบท

1. ขณะที่กระบวนการกำลังดำเนินการใน CPU การจัดการของตารางเวลาของ CPU อาจจะเกิดเหตุการณ์การเปลี่ยนสถานะของกระบวนการ หรือเกิดเหตุการณ์อื่น ๆ อย่างไรบ้าง
2. การเลือกใช้อัลกอริธึมสำหรับการจัดตารางเวลาการของการเข้าใช้ซีพียู เพื่อให้เกิดประโยชน์สูงสุดควรพิจารณาจากกฎเกณฑ์อะไรบ้าง
3. การวัดประเมินอัลกอริธึม (Algorithm Evaluation) พิจารณาจากอะไรบ้าง
4. การจัดตารางเวลาแบบวนรอบ (Round-Robin Scheduling) มีข้อดีข้อเสียอย่างไร
5. การจัดตารางเวลาแบบคิวหลายระดับที่ปรับได้ (Multilevel Feedback Queue Scheduling) มีข้อดีข้อเสียอย่างไร
6. จงหาเวลาการรอเฉลี่ยและเวลาการใช้เฉลี่ยโดยใช้อัลกอริธึม ลำดับความสำคัญ แบบแบบให้สิทธิ์ก่อน โดยพิจารณาชุดข้อมูลของ กระบวนการ 4 กระบวนการ ทุกกระบวนการเข้ามาถึงระบบที่เวลา 0 ตามลำดับของเลขประจำโปรเซส เวลาที่ต้องใช้ซีพียู มีหน่วยเป็น มิลลิวินาที (10^{-3} วินาที) และลำดับความสำคัญของกระบวนการ ดังนี้

กระบวนการ	เวลาที่ต้องใช้ซีพียู	ลำดับความสำคัญ
P1	8	4
P2	2	1
P3	1	3
P4	3	1
P5	1	2

7. จงหาเวลาการรอเฉลี่ยและเวลาการใช้เฉลี่ยโดยใช้อัลกอริธึมการจัดตารางเวลาแบบงานสั้นสุดได้ก่อน แบบแบบให้สิทธิ์ก่อน โดยพิจารณาชุดของกระบวนการ 4 กระบวนการที่ขอใช้ซีพียูตามลำดับของการเข้าถึงระบบ กับเวลาที่ต้องใช้ซีพียู มีหน่วยเป็นมิลลิวินาที (10^{-3} วินาที)

กระบวนการ	เวลาที่เข้าถึงระบบ	เวลาที่ต้องใช้ซีพียู
P1	0	5
P2	1	7
P3	2	6
P4	2	2

8. จงหาเวลาการรอเฉลี่ยและเวลาการใช้เฉลี่ยโดยใช้อัลกอริธึม RR จัดตารางเวลาการทำงานให้กับทุก ๆ กระบวนการได้เข้าใช้ซีพียูครั้งละ 2 เวลาควันตัม ซึ่งมี 4 กระบวนการ ทุกกระบวนการมาถึงระบบ ณ เวลาที่ 0 และมีเวลาที่ต้องใช้ซีพียู (Burst Time) ของแต่ละกระบวนการมีหน่วยเป็น มิลลิวินาที (0 -3 วินาที) ดังนี้

กระบวนการ	เวลาต้องใช้ซีพียู
P1	12
P2	5
P3	3
P4	4