เอกสารประกอบการสอน

วิชา ระบบปฏิบัติการ 2 (Operating Systems 2) รหัส 4121402
บทที่ 3 การจัดการหน่วยประมวลผลกลาง (CPU Management)

หลักสูตรระดับปริญญาตรี

พุทธศักราช 2551 (ปรับปรุง 2554)

โดย

จุฑาวุฒิ จันทรมาลี

สาขาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสวนดุสิต

บทที่ 3
การจัดการหน่วยประมวลกลาง
(CPU Management)

หน่วยประมวลมวลผลกลาง (CPU) เป็นองค์ประกอบสำคัญในระบบคอมพิวเตอร์ ถือว่าเป็นสมองของคอมพิวเตอร์ โดยมีหน้าที่หลักคือ อ่านและแปลคำสั่ง ประมวลผลคำสั่ง ติดต่อกับหน่วยความจำ ติดต่อรับส่งข้อมูลกับผู้ใช้ และย้ายข้อมูลและคำสั่งระหว่างหน่วยงาน เป็นต้น

การจัดการหน่วยประมวลผลกลาง (CPU Management)

ในระบบคอมพิวเตอร์ที่มีการทำงานหลายงานพร้อมๆ กัน (Multi-tasking) ระบบปฏิบัติการ (Operating System) จะทำการแบ่งเวลาให้กับงานหรือกระบวนการ (Process) แต่ละงาน สามารถประมวลผลได้อย่างรวดเร็วเสมือนว่าทำงานได้หลายๆ งานได้ในเวลาเดียวกัน ดังนั้นการจัดการกับหน่วยประมวลผลกลาง (CPU Management) จึงเกี่ยวข้องกับการจัดกระบวนการ (Process) การจัดตารางการทำงาน (CPU Scheduling) การประยุกต์ใช้อัลกอริทึมเข้าไปช่วยในการจัดตารางการทำงาน ในระบบที่มี CPU หลายตัว (Multiprocessor) ระบบทันที (Real Time) เพื่อใช้การจัดการ Processor เกิดประสิทธิภาพสูงสุด

3.1 การจัดตารางการทำงานของหน่วยประมวลผลกลาง (CPU Scheduling)

การจัดตารางการทำงานของหน่วยประมวลผลกลาง (CPU Scheduling) เป็นหน้าที่พื้นฐานของระบบปฏิบัติการ เกือบจะทุกระบบคอมพิวเตอร์ก่อนกระบวนการ (Process) จะมีการใช้งานทรัพยากรต่างๆ CPU จะเป็นส่วนที่ใช้จัดตารางการใช้ทรัพยากร

ระดับการจัดตารางการทำงาน (Level Scheduling) การจัดตารางการในระบบปฏิบัติการ คือการเลือกกระบวนการ (Process) หรือจัดสรรทรัพยากรที่มีจำกัดให้กับกระบวนการแบ่งออกเป็น 2 แบบ คือ

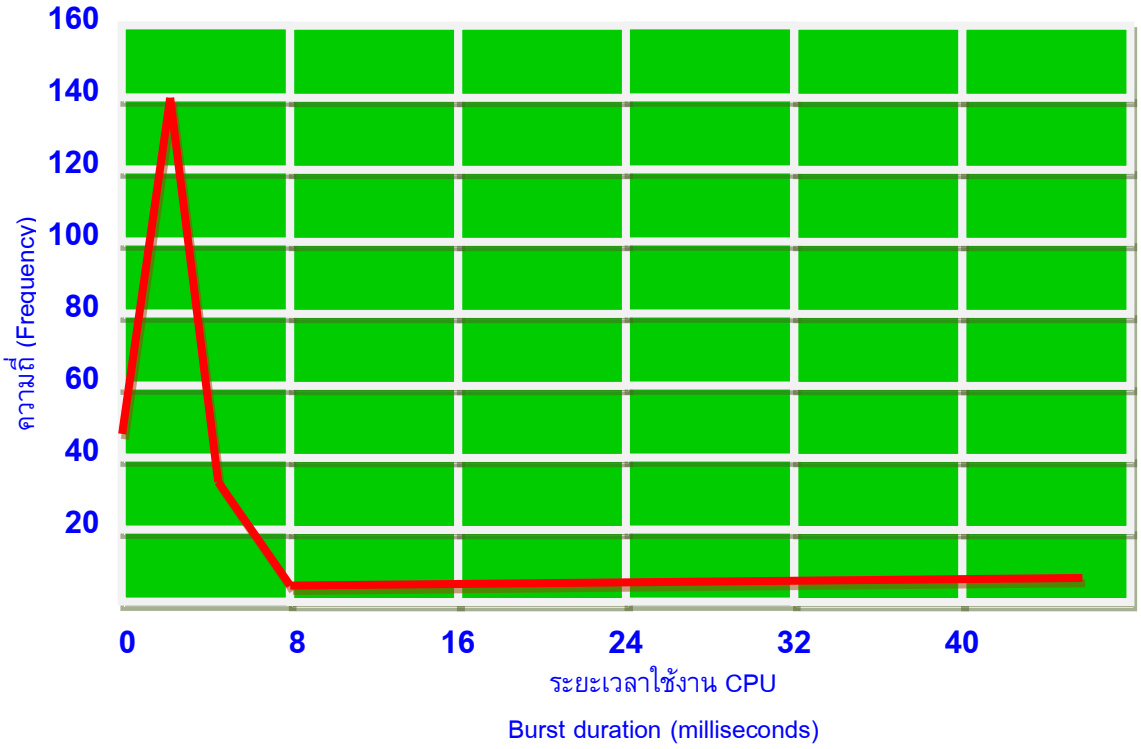
1.การจัดตารางการทำงานในระดับบน (High-Level Scheduling) เป็นการจัดลำดับงานให้กับกระบวนการ (Process) ที่จะเข้าไปใช้งานทรัพยากร CPU แบบกลุ่มซึ่งมีจำนวนกระบวนการ (Process) มากเกินกว่าระบบจะรับได้ในช่วงเวลาเดียวกัน ดังนั้น ระบบปฏิบัติการจะมีหน้าที่จัดลำดับความเหมาะสมให้แต่ละกระบวนการ (Process) เพื่อให้เกิดประโยชน์สูงสุดโดยคำนึงถึงปัจจัยต่างๆ ได้แก่ ลำดับความสำคัญ (Priority) และภาวะติดตาย (Deadlock) เป็นต้น

2.การจัดตารางการทำงานในระดับล่าง (Low-Level Scheduling) เป็นการเลือกและส่งกระบวนการ (Process) ที่จะเข้าไปใช้งานทรัพยากร CPU โดยเลือกกระบวนการที่มีสถานะพร้อม (Ready State) จากคิวโดยจะถูกส่งโดยตัวจัดการเวลา (Dispatcher)

ระยะเวลาใช้งานและรับส่งของมูลของ CPU (CPU-I/O Burst Cycle) ความสำคัญของการจัดเวลาของ CPU นั้น ขึ้นอยู่กับคุณลักษณะของกระบวนการ (Process) ซึ่งโดยทั่วๆ ไป การประมวลผล

กระบวนการ (Process Execute) จะประกอบด้วยรอบเวลา (Cycle) ที่ใช้งาน CPU และเวลาที่ต้องคอยใช้งานอุปกรณ์ I/O ในขณะที่มีการประมวลผลกระบวนการ (Process Execute) จะมีการสลับการทำงานระหว่าง 2 ช่วงเวลานี้เท่านั้น คือ ระยะเวลาใช้งาน (CPU Burst) และระยะเวลาในการรับ - ส่งข้อมูล (I/O Burst) แสดงได้ดังรูปที่ 3.1 ซึ่งจะไม่เกิดขึ้นพร้อมกัน การประมวลผล (Execute) มักจะเริ่มจากการใช้งาน CPU แล้วก็จะตามด้วยการคอยการใช้งานอุปกรณ์ I/O หลังจากใช้งานอุปกรณ์ I/O เสร็จ ก็จะกลับมาใช้งาน CPU ต่อ สลับกันไปเรื่อยๆ จนกว่าจะเสร็จสิ้นการประมวลผล ซึ่งการสิ้นสุดการเสร็จสิ้นการประมวลผลนี้มักจะใช้เวลา CPU (CPU Burst) มากกว่าการรอคอยในระยะเวลาในการรับ - ส่งข้อมูล (I/O Burst) แสดงได้ดังรูปที่ 3.2

รูปที่ 3.1 แสดงการสลับการทำงานระหว่างระยะเวลาใช้งานและระยะเวลาในการรับ– ส่งข้อมูล (Alternating sequence of CPU and I/O bursts)



รูปที่ 3.2 แสดงฮีสโตแกรมของระยะเวลาการใช้งาน CPU (Histogram of CPU-burst times)

ตัวจัดตารางการทำงานของ CPU (CPU Scheduler) เมื่อใดก็ตามที่ CPU ว่างระบบปฏิบัติการ (Operating System) จะต้องเข้ามาดำเนินการนำเอากระบวนการ (Process) ใดกระบวนการหนึ่งที่คอยอยู่ในคิวเข้ามาใช้งาน CPU โดยการเลือกกระบวนการ (Process) เพื่อเข้ามาใช้งาน CPU นี้ จะถูกจัดการด้วยส่วนที่เรียกว่า ตัวจัดการเวลาช่วงสั้น (Short-Term Scheduler) หรือ ตัวจัดการเวลา CPU ตัวจัดการเวลานี้จะเลือกกระบวนการ (Process) ที่อยู่ในหน่วยความจำที่พร้อมในการประมวลผล (Execute) ที่สุด เพื่อให้ครอบครองเวลาการใช้งาน CPU และทรัพยากรที่เกี่ยวข้องกับกระบวนการนั้น (Process) นั้นภายใต้บล็อกควบคุมกระบวนการ (Process Control Block : PCBs) ซึ่งมีอัลกอริทึมในการจัดการคิวที่มีความพร้อม (Ready Queue) ให้เลือกใช้หลากหลาย เช่น คิวแบบมาก่อนได้รับบริการก่อน (FIFO Queue) คิวแบบเรียงตามลำดับอาวุโส (A Priority Queue) ต้นไม้ (Tree) หรือลิงค์ลิสก์แบบไม่เรียงลำดับ (Unordered Linked List) เป็นต้น

การให้สิทธิการจัดเวลา (Preemptive Scheduling) การตัดสินใจของ CPU ในการเลือกว่ากระบวนการใด (Process) จะถูกประมวลผล (Execute) ขึ้นอยู่กับสถานการณ์ดังต่อไปนี้นี้

1. เมื่อมีการกระบวนการ (Process) เปลี่ยนสถานะของจากสถานะที่ประมวลผลอยู่ (Running State) ไปเป็นสถานะคอย (Waiting State) เช่น การร้องขอใช้งานอุปกรณ์ I/O เป็นต้น
2. เมื่อมีการเปลี่ยนสถานะของ process จากสถานะรัน เป็นสถานะพร้อม
3. เมื่อมีการเปลี่ยนสถานะของ process จากสถานะคอย เป็นสถานะพร้อม
4. เมื่อ process เสร็จสิ้นไปแล้ว

ตัวจัดการเวลา (Dispatcher) เป็นคอมโพเนนต์ (Component) ที่สำคัญอีกตัวหนึ่งที่เกี่ยวข้องกับฟังก์ชันในการจัดเวลาในหน่วยประมวลผลกลาง (CPU) ซึ่งเป็นโมดูลที่ทำหน้าที่ควบคุมการครอบครองเวลาใน CPU ของกระบวนการ (Process) โดยฟังก์ชันนี้จะประกอบด้วย

- การย้ายไปในส่วนของอธิบายขยายความ (Switching Context)
- การย้ายไปในส่วนของผู้ใช้ (User Mode)
- การกระโดด (Jumping) ไปยังตำแหน่งที่เหมาะสมของโปรแกรม เพื่อที่จะเริ่มรันโปรแกรมใหม่อีกครั้ง

นอกจากนี้ตัวจัดการเวลา (Dispatcher) ควรมีการทำงานที่เร็วที่สุดเท่าที่จะทำได้ เพราะว่าตัวจัดการเวลาจะต้องทำงานทุกครั้งที่มีการย้ายกระบวนการ (Process) ซึ่งเวลาที่ถูกใช้ไปกับการทำการแบบนี้เรียกว่า Dispatch Latency

เกณฑ์ในการจัดตารางการทำงาน (Scheduling Criteria) ใช้ในการเปรียบเทียบความแตกต่างของอัลกอริทึมที่ใช้สำหรับกระบวนการ (Process) ในการเข้ามาใช้งาน CPU หรือเปรียบเทียบวิธีการจัดตารางการทำงาน CPU ว่าวิธีใดเหมาะสมที่สุด ดังนี้

1. การใช้ CPU ให้เกิดประโยชน์ (CPU Utilization) สูงสุด (0-100%) เช่น 40 % สำหรับการโหลดเบาๆ (Lightly Load) และ ไม่ควรเกิน 90 % สำหรับการโหลดหนัก (Hardily Load)
2. ปริมาณงาน (Throughput) ขณะที่มี CPU การประมวลมีจำนวนกระบวนการ (Process) ที่ทำงานเสร็จสมบูรณ์ภายในหนึ่งหน่วยเวลา (Per Time Unit) เท่าใด
3. การวนรอบการทำงาน (Turnaround Time) คือ ระยะเวลาที่กระบวนการ (Process) เริ่มเข้าไปประมวลผลจนกระทั้งทำงานเสร็จสมบูรณ์ หรืออาจจะเป็นเวลาที่กระบวนการ (Process) รอเพื่อที่จะเข้าไปใช้งาน CPU รออยู่ในแถวคอยเพื่อเปลี่ยนสถานะเป็นพร้อม (Ready) ในการประมวลผล (Execute) บน CPU หรือรอเพื่อใช้งานอุปกรณ์ I/O เป็นต้น
4. เวลาที่ใช้คอย (Waiting Time) รออยู่ในแถวคอยเพื่อเปลี่ยนสถานะเป็นพร้อม (Ready)
5. เวลาในการตอบสนอง (Respond Time) ระยะเวลาที่กระบวนการ (Process) ใช้ในการสนองต่อคำสั่งจากผู้ใช้ที่ร้องขอ (Request) จนกระทั้งได้ผลลัพธ์ส่งกลับมายังผู้ใช้ โดยที่เวลาในการตอบสนองมักจะขึ้นอยู่กับความเร็ว (Speed) ของอุปกรณ์แสดงผล (Output Device)

3.2 อัลกอริทึมสำหรับการจัดตารางการทำงาน (Scheduling Algorithms)

หน้าที่ของตัวจัดคิวคือ คัดเลือกโปรเซสซึ่งรออยู่ในสถานะพร้อมที่เหมาะสมที่สุดให้เข้าไปอยู่ในสถานะ รัน (ได้ครอบครองซีพียู) โดยแท้จริงแล้วการส่งโปรเซสที่ถูกเลือกแล้วให้เข้าไปอยู่ในสถานะรันเป็นหน้าที่ของโปรเซสของ OS ที่ชื่อตัวส่ง (dispatcher) ใน แง่การทำงานแล้วตัวจัดคิวจะเป็นผู้คัดเลือกโปรเซสและเรียกให้ตัวส่งส่งโปรเซสที่ถูกเลือกแล้วเข้าไปในสถานะรันเป็นความรับผิดชอบของตัวจัดคิว

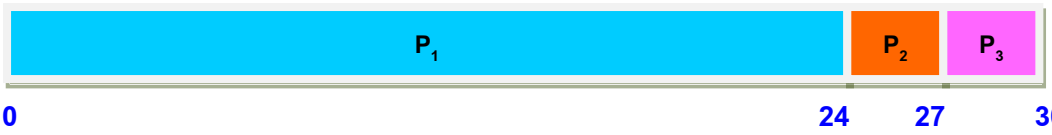
1. การจัดเวลาแบบมาก่อนได้ก่อน (FCFS : First-come First-served Scheduling)

การจัดคิวแบบ FCFS (first-come-first-served) วิธีการคัดเลือกแบบ FCFS นี้ เป็นวิธีที่ง่ายที่สุด คือ โปรเซสไหนเข้ามารอในคิวก่อนจะได้ครอบครองซีพียูก่อน ตามลำดับเวลาของการเข้ามาอยู่ในคิว คือ

"มาก่อนได้ก่อน" โปรเซสที่ได้ครอบครองซีพียูจะทำงานไปจนเสร็จ ไม่มีระยะเวลาควอนตัมซึ่งจำกัดเวลาการครอบครองซีพียู แต่ถ้าโปรเซสมีการเรียกใช้งานอุปกรณ์อินพุต-เอาต์พุต หรือรอเหตุการณ์บางอย่าง โปรเซสนั้นต้องปลดปล่อยซีพียู และออกจากสถานะรันไปอยู่ในสถานะติดขัด เมื่อใดที่งานเสร็จสิ้นลง หรือเกิดเหตุการณ์ที่กำลังรออยู่ โปรเซสนั้นจึงค่อยกลับเข้าไปอยู่ต่อท้ายคิวของสถานะพร้อมใหม่อีกครั้ง เราอาจแสดงการเปลี่ยนสถานะของโปรเซสโดยใช้การจัดคิวแบบ FCFS ซึ่ง จะเห็นว่าแตกต่างกับรูปแสดงการเปลี่ยนสถานะของโปรเซสที่เคยกล่าวมาคือ ไม่มีการเปลี่ยนสถานะของโปรเซสจากสถานะรันมายังสถานะพร้อมโดยตรง

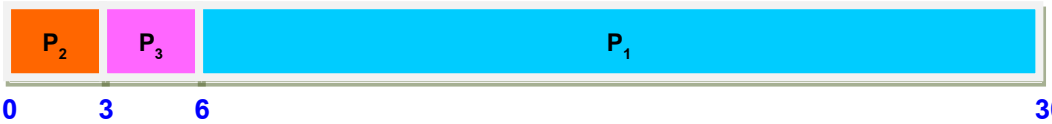
กระบวนการ (Process)	เวลาในการใช้งาน CPU (Burst Time)
P_1	24
P_2	3
P_3	3

EX. ถ้ามีกระบวนการ (Process) เข้ามาใช้งานตามลำดับดังนี้ P_1, P_2, P_3 และใช้วิธีการแบบ FCFS แสดงอยู่ในรูปแบบ Gantt chart ได้ดังนี้



P_1 ใช้เวลาในการคอย 0 มิลลิวินาที (millisecond), P_2 ใช้เวลาในการคอย 24 มิลลิวินาที (millisecond), P_3 ใช้เวลาในการคอย 27 มิลลิวินาที (millisecond) ดังนั้นใช้เวลาเฉลี่ยในการค่อยเป็น (0 + 24 + 27)/3 = 17 มิลลิวินาที (millisecond)

แต่ถ้ากระบวนการ (Process) เข้ามาใช้งานเปลี่ยนลำดับเป็น P_2, P_3, P_1 แสดงอยู่ในรูปแบบ Gantt chart ได้ดังนี้



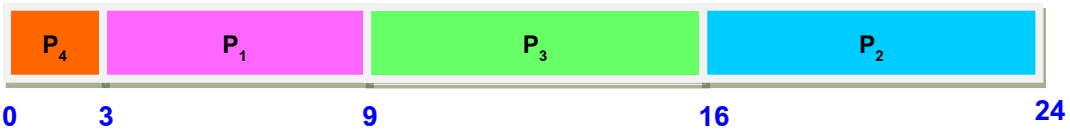
ดังนั้นใช้เวลาเฉลี่ยในการค่อยเป็น (6 + 0 + 3)/3 = 3 มิลลิวินาที (millisecond) ซึ่งจะเห็นว่าเวลาเฉลี่ยในการคอยลดลง ดังนั้นสรุปได้ว่าเวลาเฉลี่ยในการคอยจะขึ้นอยู่กับวิธีการเรียงลำดับหรืออัลกอริทึมที่เลือกใช้และอาจเปลี่ยนแปลงไปตามขนาดของกระบวนการ (Process) ที่เข้ามาใช้งานเวลา CPU

2. การจัดเวลาแบบงานสั้นทำก่อน (SJF: Short-Job-First Scheduling)

การจัดคิวแบบ SJN (shortest job next) การ คัดเลือกโปรเซสด้วยวิธีนี้ จะคัดเลือกเอาโปรเซสที่ต้องการเวลาในการทำงานน้อยที่สุด ทำให้โปรเซสที่ต้องการเวลาในการทำงานน้อยจบออกไปได้เร็วขึ้น จำนวนโปรเซสในระบบที่รออยู่ในคิวมีก็จะมีจำนวนลดลง และทำให้เวลาโดยเฉลี่ยในการทำงาน 1 งานเสร็จหรือเวลาครบงาน (turnaround time) น้อยลงแต่การจัดคิวแบบนี้เป็นผลเสียต่อโปรเซสที่ต้องการเวลาในการทำงานนาน

กระบวนการ (Process)	เวลาในการใช้งาน CPU (Burst Time)
P_1	6
P_2	8
P_3	7
P_4	3

EX. ถ้ามีกระบวนการ (Process) เข้ามาใช้งานตามลำดับดังนี้ P_1, P_2, P_3, P_4 และใช้วิธีการแบบ SJF แสดงอยู่ในรูปแบบ Gantt chart ได้ดังนี้



P_1 ใช้เวลาในการคอย 3 มิลลิวินาที (millisecond), P_2 ใช้เวลาในการคอย 16 มิลลิวินาที (millisecond), P_3 ใช้เวลาในการคอย 9 มิลลิวินาที (millisecond) และ P_4 ใช้เวลาในการคอย 0 มิลลิวินาที (millisecond) ดังนั้นใช้เวลาเฉลี่ยในการค่อยเป็น (3 + 16 + 9 + 0)/4 = 7 มิลลิวินาที (millisecond) แต่ถ้าใช้วิธีการแบบ FCFS จะใช้เวลาเฉลี่ยในการค่อยถึง (0 + 6 +14 + 21)/4 = 10.25 มิลลิวินาที (millisecond)

3. การจัดเวลาตามลำดับความสำคัญ (Priority Scheduling)

การจัดคิวแบบลำดับความสำคัญ (priority queue) คิว แบบลำดับความสำคัญมีลักษณะแตกต่างกับคิวธรรมดา ภายในคิวจะมีการจัดเรียงลำดับของโปรเซสต่าง ๆ ตามลำดับความสำคัญของโปรเซสนั้น โปรเซสที่อยู่ต้นคิวจะมีลำดับความสำคัญมากที่สุด และลดลงเรื่อย ๆ โปรเซสที่อยู่ท้ายคิวคือโปรเซสที่มีลำดับความสำคัญต่ำสุด การคัดเลือกโปรเซสจะเอาโปรเซสที่อยู่ต้นคิว (มีลำดับความสำคัญสูงสุด) เข้าไปครอบครองซีพียูก่อน ดังนั้นถึงแม้ว่าโปรเซสที่เข้าคิวทีหลังแต่มีลำดับความสำคัญสูงกว่าก็อาจได้ เข้าไปครอบครองซีพียูก่อน เช่น โปรเซส P_5 เข้าคิวเป็นโปรเซสหลังสุด แต่จะได้ครอบครองซีพียูก่อนโปรเซส P_3 และ P_4 เพราะมีความสำคัญสูงกว่า แต่ถ้ากรณีที่โปรเซสมีลำดับความสำคัญเท่ากันจะใช้วิธีมาก่อนได้ก่อนในการช่วยพิจารณา

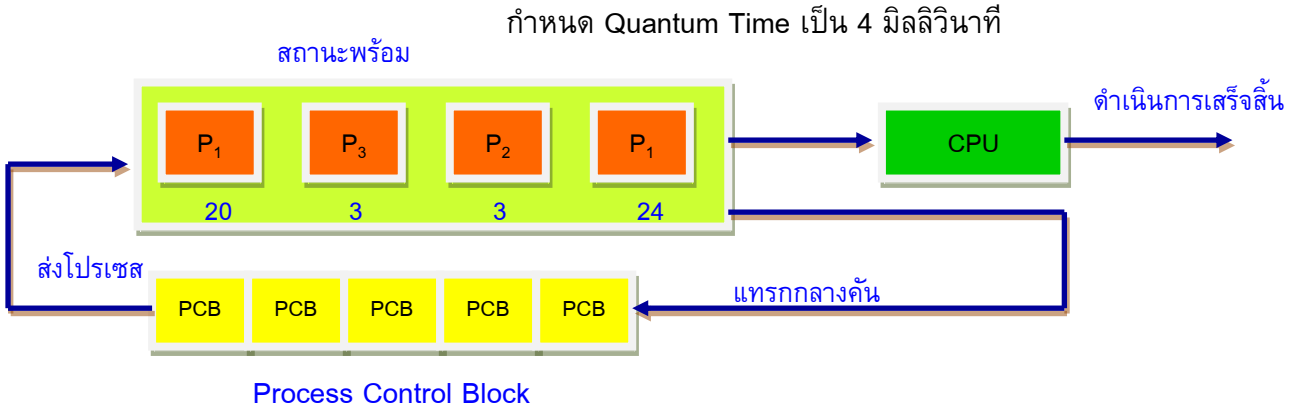
กระบวนการ (Process)	เวลาในการใช้งาน CPU (Burst Time)	ลำดับความสำคัญ (Priority)
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

EX. ถ้ามีกระบวนการ (Process) เข้ามาใช้งานตามลำดับดังนี้ P_1, P_2, P_3, P_4, P_5 และใช้วิธีการแบบตามลำดับความสำคัญ Priority Queue แสดงอยู่ในรูปแบบ Gantt chart ได้ดังนี้

P_1 ใช้เวลาในการคอย 6 มิลลิวินาที (millisecond), P_2 ใช้เวลาในการคอย 0 มิลลิวินาที (millisecond), P_3 ใช้เวลาในการคอย 16 มิลลิวินาที (millisecond) และ P_4 ใช้เวลาในการคอย 18 มิลลิวินาที (millisecond) และ P_1 ใช้เวลาในการคอย 1 มิลลิวินาที (millisecond) ดังนั้นใช้เวลาเฉลี่ยในการค่อยเป็น (6 + 0 + 16 + 18 + 1)/5 = 8.2 มิลลิวินาที (millisecond)

4. การจัดเวลาแบบวนรอบ (RR : Round-Robin Scheduling)

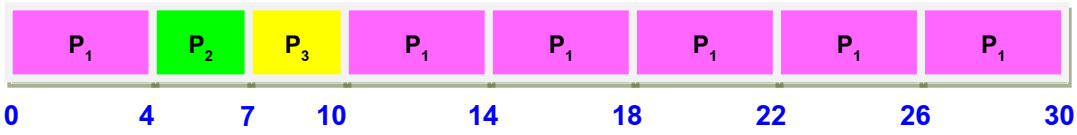
การจัดคิวแบบ RR (round-robin) การจัดคิวแบบ RR อาจเรียกว่าเป็นการจัดคิวแบบมีการวนรอบ ลักษณะการคัดเลือก โปรเซสในคิวจะเป็นแบบ FCFS คือ "มาก่อนได้ก่อน" แต่ต่างกันนิดหน่อยตรงที่การครอบครองซีพียูของโปรเซสในสถานะรันจะถูกจำกัด เวลาไว้ด้วยระยะเวลาควอนตัม ทำให้โปรเซสที่ต้องการเวลาในการทำงานนานจะต้องเปลี่ยนสถานะหมุนระหว่างสถานะ พร้อมและสถานะรัน การจัดคิวแบบ RR สามารถ แก้ปัญหาการคอยนานของโปรเซสที่ต้องการเวลาทำงานน้อย ๆ ถ้าระบบกำหนดเวลาควอนตัมเป็น 4 มิลลิวินาที โปรเซส P_1 ต้องมีการวนรอบเปลี่ยนสถานะระหว่างสถานะรันและสถานะพร้อม 6 ครั้ง โปรเซส P_2 1 ครั้งโปรเซส P_3 1 ครั้ง เมื่อโปรเซส P_1 เข้าไปอยู่ในสถานะรันครั้งแรกและกลับออกมา โปรเซส P_2 จะได้ครอบครองซีพียูได้และ ทำงานเสร็จโปรเซส P2 จบและออกจากระบบไปเลย โปรเซสถัดไปที่จัดได้ครอบครองซีพียูคือ P_3 โปรเซส P_3 จะครอบครองซีพียู 3 มิลลิวินาที จนกระทั่งโปรเซส P_3 จบ เหลือโปรเซส P_1 เพียงโปรเซสเดียว จนเข้ามาใช้ CPU จนดำเนินการเสร็จสิ้น โดยมีตัวควบคุมการส่งโปรเซสเข้าไปในคิวที่เรียกว่า Process Control Block ดังรูปที่ 3.3 เป็นแผนภาพแสดงการสลับกันทำงานของโปรเซสทั้ง 3 และเป็นผลที่เกิดขึ้นจากการจัดคิวแบบ RR



รูปที่ 3.3 แสดงการครอบครอง CPU แบบวนรอบ (round-robin)

กระบวนการ (Process)	เวลาในการใช้งาน CPU (Burst Time)
P_1	24
P_2	3
P_3	3

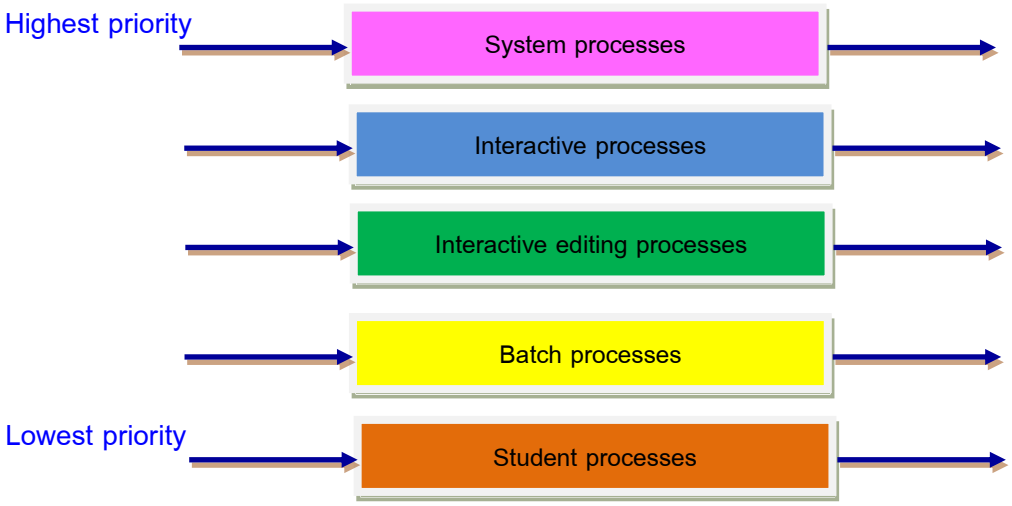
EX. ถ้ามีกระบวนการ (Process) เข้ามาใช้งานตามลำดับดังนี้ P_1, P_2, P_3 และใช้วิธีการแบบวนรอบ (Round-Robin) และใช้เวลาควอนตัม (Time Quantum) เป็น 4 มิลลิวินาที (millisecond) แสดงอยู่ในรูปแบบ Gantt chart ได้ดังนี้



P_1 ใช้เวลาในการคอย 0 มิลลิวินาที (millisecond), P_2 ใช้เวลาในการคอย 4 มิลลิวินาที (millisecond), P_3 ใช้เวลาในการคอย 3 มิลลิวินาที (millisecond) ดังนั้นใช้เวลาเฉลี่ยในการค่อยเป็น [(10-4)+ 4 + 7]/3 = 17/3 = 5.66 มิลลิวินาที (millisecond)

5. การจัดเวลาแบบคิวหลายระดับ (Multilevel Queue Scheduling)

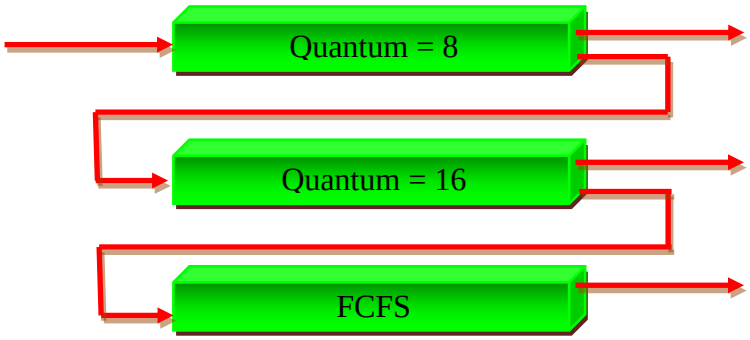
เป็นวิธีการจัดคิวโดยการแบ่งคิวออกเป็นระดับ (Partition) แสดงดังรูปที่ 3.3 มีการกำหนดคุณลักษณะของแต่ละกระบวนการ (Process) เช่น ขนาดของหน่วยความจำ ลำดับความสำคัญของโปรเซส และชนิดของโปรเซส แต่ละคิวจะมีอัลกอริทึมการจัดเวลาเป็นของตัวเอง และมีจัดลำดับความสำคัญของแต่ละโปรเซสในรูปแบบโปรเซสลำดับความสำคัญต่ำสุด (Lowest Priority) ไปจนถึงโปรเซสลำดับความสำคัญสูงสุด (Highest Priority)



รูปที่ 3.4 แสดงการจัดเวลาแบบคิวหลายระดับ (Multilevel Queue Scheduling)

6. การจัดเวลาแบบคิวหลายระดับแบบย้อนกลับ (Multilevel Feedback Queue Scheduling)

เพื่อ ให้การจัดคิวเป็นไปอย่างมีประสิทธิภาพ จึงมีการนำหลาย ๆ เทคนิคมาประยุกต์เข้าด้วยกัน เป็นการจัดคำแบบวนรอบ ที่คำนึงถึงความสำคัญของงาน ทำให้งานที่มีความสำคัญเหมือนกันอยู่ในคิวเดียวกัน และให้งานสำคัญน้อย ๆ อยู่ในคิวที่สำคัญน้อยเช่นกัน แสดงดังรูปที่ 3.4



รูปที่ 3.5 แสดงการจัดเวลาแบบคิวหลายระดับ (Multilevel Queue Scheduling)

3.3 การจัดตารางการทำงานของระบบหลายกระบวนการ (Multiple-Processor Scheduling)

การทำงานในลักษณะที่มี CPU หลายตัวต่างก็มีระบบการจัดเวลาของตัวเอง จะต้องมีการออกแบบอย่างระมัดระวังในเรื่องของการที่โปรเซสแต่ละโปรเซสอาจจะต้องการใช้ข้อมูลในฐานข้อมูลในเวลาเดียวกัน รวมทั้งต้องระวังไม่ให้ CPU ว่างงานพร้อมกัน

ในระบบคอมพิวเตอร์บางระบบที่มีการพัฒนาเพื่อให้ได้ประสิทธิภาพสูงสุด ได้มีการใช้ CPU แยกต่างหากระหว่างงานภายในของระบบ และกับงานที่เป็นของผู้ใช้ซึ่งการทำงานแยกกันของ CPU แบบนี้ทำให้การออกแบบระบบปฏิบัติการมีความซับซ้อนน้อยลงมาก เนื่องจากมี CPUเพียงตัวเดียวที่จะเข้าไปใช้ข้อมูลของระบบปฏิบัติการ เพราะว่า CPU ตัวอื่นๆ ไม่ว่าจะกี่ตัวก็ตาม จะเข้าถึงได้เฉพาะข้อมูลของ

ผู้ใช้ภายนอกเท่านั้น

3.4 การจัดตารางการทำงานของระบบแบบทันที (Real-Time Scheduling)

Hard real-time คือระบบที่สามารถทำงานใดงานหนึ่งให้เสร็จตามเวลาที่กำหนดได้ ซึ่งงานที่จะรับเข้ามาแต่ละงานนั้นจะมีความต้องการของเวลาที่ต้องการให้ เสร็จมาด้วย ดังนั้นตัวจัดเวลาจะต้องเป็นตัวตัดสินใจว่าจะรับงานเข้ามาทำหรือไม่

Soft real-time คือระบบที่แบ่งเวลาธรรมดาที่มีการให้ระดับความสำคัญแก่งานบางประเภท หรืองานที่ถูกเลือกไว้ล่วงหน้าว่าเป็นงานเร่งด่วน ซึ่งอาจทำให้เกิดปัญหาของการทำงานในระดับต่ำๆ อาจไม่ได้รับเวลาของ CPU เลย

3.5 วิวัฒนาการของอัลกอริทึม (Algorithm Evolution)

การกำหนดกฎเกณฑ์และวิธีการเลือกใช้อัลกอริทึมมักจะถูกกำหนดจากการเข้าไปใช้งาน CPU เวลาในการโต้ตอบ (Respond Time) หรือปริมาณงาน (Throughput) ที่ได้ของแต่ละโปรเซส ดังนั้นการเลือกใช้อัลกอริทึมจะต้องคำนึงถึงการวัดประสิทธิภาพการใช้งานโดยพิจารณาจาก

1. การใช้ประโยชน์จาก CPU ได้สูงสุดภายใต้ข้อกำหนดของเวลาในการตอบสนอง (Maximum Respond Time) ให้ได้อย่างรวดเร็วภายในหนึ่งวินาที (หนึ่งหน่วยเวลา)

2. การใช้ประโยชน์จาก CPU ได้สูงสุดของปริมาณงาน (Throughput) ที่ได้เมื่อเทียบกับอัตราส่วนของเวลาในการประมวลผล (Execution Time) กับจำนวนโปรเซสที่ทำงานเสร็จสมบูรณ์ภายในหนึ่งหน่วยเวลา ตั้งแต่โปรเซสนั้นถูกส่งไปใช้งานจนกระทั้งได้รับผลลัพธ์กลับมาโดยมีอัลกอริทึมหรือรูปแบบให้เลือกใช้ ดังนี้

Deterministic Modeling

วิธีนี้เป็นวิธีการคัดเลือกที่เรียกว่า analytic evaluation ซึ่งจะนำเอา Algorithm ชนิดต่างๆ และลักษณะของงานมาสร้างสูตร เพื่อใช้ในการคำนวณหาตัวเลขของประสิทธิภาพที่สามารถวัดและเปรียบเทียบได้

Queuing Models

ลักษณะของงานที่เข้ามาในระบบคอมพิวเตอร์นั้นมักจะมีลักษณะที่ไม่แน่นอน ในแต่ละวันที่มีการใช้ระบบคอมพิวเตอร์นั้น งานต่างๆที่เข้ามาอาจมีลักษณะที่ไม่ซ้ำกันเลย อย่างไรก็ตามมีสิ่งหนึ่งที่เราอาจจะสามารถทำนายหรือกำหนดได้ ก็คือ การกระจายของเวลาในการใช้ CPU และของการใช้อุปกรณ์ Input/output ซึ่งเราสามารถที่จะกำหนดแบบคร่าวๆได้ ซึ่งก็เช่นเดียวกันกับเวลาของการมาถึงระบบของงานต่างๆที่เราก็สามารถกำหนดไว้แบบไม่เฉพาะเจาะจงช่นกัน

Simulations

วิธีการจำลองระบบ การที่เราจะเลือกวิธีการ หรือเลือก Algorithm ที่ถูกต้องต่อระบบใดๆ อย่าง

เป็นจริงเป็นจังแล้ว เราสามารถใช้วิธีการของการจำลองระบบ ซึ่งวิธีการนี้จะสามารถคำนวณตัวเลขต่างๆ ออกมาได้อย่างเที่ยงตรงมากขึ้น การทำการจำลองระบบในที่นี้จะเกี่ยวข้องกับการใช้โปรแกรมคอมพิวเตอร์ ซึ่งจะต้องมีการเขียนโปรแกรมเพื่อใช้เป็นตัวแทนหรือหุ่นจำลองของระบบต่างๆ ในคอมพิวเตอร์ นอกจากนี้ยังต้องมีการเขียนโปรแกรมเพื่อเป็นตัวแทนของสิ่งแวดล้อมที่เกี่ยว ข้องกับระบบคอมพิวเตอร์นั้นๆอีกด้วย

Implementation

วิธีการสร้างขึ้นมาจริงๆ อย่างไรก็ตามการสร้างแบบจำลอง ก็ยังคงเป็นการจำลองแบบที่ไม่มีทางจะเหมือนจริงได้ สิ่งที่ดีกว่าก็คือ การสร้าง Algorithm ชนิดต่างๆ เพื่อทดลองใช้กับโปรแกรมขัดการระบบจริงๆในระบบคอมพิวเตอร์ที่ใช้งานในสิ่ง แวดล้อมจริง

สรุป

การจัดการหน่วยประมวลผลกลาง (CPU Management) ซึ่งในระบบคอมพิวเตอร์ที่มีการทำงานหลายงานพร้อมๆ กัน (Multi-tasking) ระบบปฏิบัติการ (Operating System) จะทำการแบ่งเวลาให้กับงานหรือกระบวนการ (Process) แต่ละงาน สามารถประมวลผลได้อย่างรวดเร็วเสมือนว่าทำงานได้หลายๆ งานได้ในเวลาเดียวกัน ดังนั้นการจัดการกับหน่วยประมวลผลกลาง (CPU Management) จึงเกี่ยวข้องกับการจัดกระบวนการ (Process) การจัดตารางการทำงาน (CPU Scheduling) การประยุกต์ใช้อัลกอริทึมเข้าไปช่วยในการจัดตารางการทำงานในระบบที่มี CPU หลายตัว (Multiprocessor) ระบบทันที (Real Time) เพื่อใช้การจัดการ Processor เกิดประสิทธิภาพสูงสุด โดยอัลกอริทึมสำหรับการจัดตารางการทำงานมีดังนี้

1. การจัดเวลาแบบมาก่อนได้ก่อน (FCFS : First-come First-served Scheduling) เป็นอัลกอริทึมประเภทที่ไม่สามารถแทรกกลางคันได้ (Non-preemptive) เหมาะกับการใช้งานในระบบงานแบบกลุ่ม (Batch System) มีข้อดีคือง่ายต่อการประยุกต์ใช้งาน ข้อเสียคือไม่สามารถคาดเดาเหตุการณ์ของแต่ละโปรเซสในการเข้าใช้งาน CPU ได้
2. การจัดเวลาแบบงานสั้นทำก่อน (SJF: Short-Job-First Scheduling) เป็นอัลกอริทึมประเภทที่ไม่สามารถแทรกกลางคันได้ (Non-preemptive) เหมาะกับการใช้งานในระบบงานแบบกลุ่ม (Batch System) มีข้อดีคือ มีค่าเฉลี่ยเวลาในการรอคอยในการใช้งาน CPU แต่ละโปรเซสน้อยที่สุด ข้อเสียคืออาจทำให้เกิดปัญหาการขาดแคลนทรัพยากร (Starvation)
3. การจัดเวลาตามลำดับความสำคัญ (Priority Scheduling) เป็นอัลกอริทึมประเภทที่ไม่สามารถแทรกกลางคันได้ (Non-preemptive) เหมาะกับการใช้งานในระบบงานแบบกลุ่ม (Batch System) มีข้อดีคือลำดับงานที่มีความสำคัญจะได้รับการประมวลผลก่อนเสมอ ข้อเสียคืออาจทำให้เกิดปัญหาการขาดแคลนทรัพยากร (Starvation)
4. การจัดเวลาแบบวนรอบ (RR : Round-Robin Scheduling) เป็นอัลกอริทึมประเภทที่สามารถแทรกกลางคันได้ (Preemptive) เหมาะกับการใช้งานในระบบงานแบบโต้ตอบ (Interactive System) มีข้อดีคือ

สามารถตอบสนองต่อผู้ใช้งานได้อย่างรวดเร็ว ข้อเสียคือต้องมีการกำหนดระยะเวลาการใช้งาน (Quantum Time) ที่แน่นอนให้กับแต่ละโปรเซส

5. การจัดเวลาแบบคิวหลายระดับ (Multilevel Queue Scheduling) เป็นอัลกอริทึมประเภทที่สามารถแทรกกลางคันได้หรือไม่สามารถแทรกกลางคันได้ (Preemptive/Non- Preemptive) เหมาะกับการใช้การใช้งานในระบบงานแบบกลุ่ม (Batch System) และงานในระบบงานแบบโต้ตอบ (Interactive System) มีข้อดีคือ มีความยืดหยุ่นในการใช้งาน (Flexibility) ข้อเสียคือมี overhead ในการจากการตรวจสอบลำดับคิว

การจัดตารางงานของระบบตัวประมวลผลหลายตัว (Multiprocessor) จะแบ่งการทำงานออกเป็น 2 ลักษณะ คือ

1. การใช้ตัวประมวลผล (Processor) ที่หลากหลาย (Heterogeneous System) โดยกำหนดให้โปรเซสมีคิวการทำงานเป็นของตนเอง เพื่อจัดให้แต่ละโปรเซสทำงานกับตัวประมวลผลที่เหมาะสม
2. การใช้ตัวประมวลผล (Processor) เดียวกันทั้งหมด (Homogeneous System) โดยแบ่งคิวให้กับตัวประมวลผล (Processor) แต่ละตัว โดยมีการจัดการคิวในลักษณะ “คิวร่วม” ให้กับโปรเซส โดยการจัดคิวร่วมแบ่ง 2 วิธีคือ
   2.1 ตัวจัดตารางการทำงานเอง (Symmetric Multiprogramming)
   2.2 ตัวจัดการให้โปรเซสตัวใดตัวหนึ่งมีหน้าที่จัดตารางการทำงานโดยเฉพาะ (Asymmetric Multiprogramming)

การจัดตารางการทำงานของระบบแบบทันที (Real-Time Scheduling) แบ่งออกเป็น

1. Hard real-time เป็นการจัดตารางทำงานเพื่อกำหนดสิทธิ์ให้กับโปรเซสทำงาน โดยรับประกันว่างานนั้นจะเสร็จสิ้นตามระยะเวลาที่กำหนด โดยมีการกำหนดระยะเวลาที่แน่นอน
2. Soft real-time เป็นการจัดตารางทำงานเพื่ออนุญาตให้โปรเซสทำงานได้อย่างต่อเนื่องจนกว่างานนั้นจะเสร็จสิ้น โดยไม่มีการกำหนดระยะเวลาที่แน่นอน