แผนการบริหารการสอนประจำบทที่ 2

เนื้อหาประจำบท

บทที่ 2 การจัดการโปรเซส

1. แนวคิดในการจัดการกระบวนการของโปรแกรม
2. การจัดตารางการดำเนินการของโปรเซส
3. การดำเนินการของโปรเซส
4. การสื่อสารระหว่างโปรเซส

จุดประสงค์เชิงพฤติกรรม

เมื่อศึกษาบทที่ 2 แล้วนักศึกษาสามารถ

1. เข้าใจความหมายของโปรเซสเพื่อใช้ในการดำเนินการของโปรแกรม
2. อธิบายส่วนที่สำคัญต่างๆ ของโปรเซส
3. เข้าใจการสื่อสารระหว่างโปรเซส โดยการใช้หน่วยความจำร่วมกันและการส่งข้อความ

กิจกรรมการเรียนการสอนประจำบท

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

สื่อการเรียนการสอน

1. สื่ออิเล็กทรอนิกส์ประกอบการสอนวิชาระบบปฏิบัติการ
2. เอกสารประกอบการสอนวิชาระบบปฏิบัติการ
3. หนังสืออ่านประกอบค้นคว้าเพิ่มเติม

การวัดผลและประเมินผล

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

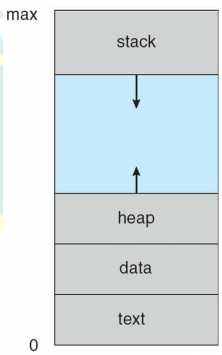
บทที่ 2
การจัดการโพรเซส

2.1 แนวคิดเรื่องโพรเซส

ระบบคอมพิวเตอร์ในสมัยยุคแรก ๆ มีการทำงานในลักษณะแบบโมโนโปรแกรมมิ่ง (Mono programming) คือ ในขณะเวลาใดเวลาหนึ่งจะมีเพียงโพรเซสเดียวเท่านั้นที่ทำงานได้ ข้อเสียคือ ระบบไม่สามารถใช้ทรัพยากรได้อย่างเต็มประสิทธิภาพ เนื่องจากมีบางช่วงที่ทรัพยากรของระบบว่าง แต่ไม่สามารถใช้ทรัพยากรเหล่านั้นได้ ต่อมามีการพัฒนาระบบคอมพิวเตอร์ที่สามารถทำได้หลายงานพร้อมกัน และสามารถใช้ทรัพยากรร่วมกัน โดยมีการแบ่งงานหรือโปรแกรมออกเป็นส่วนย่อย และทำงานแตกต่างกัน เรียกโปรแกรมส่วนย่อยเหล่านี้ว่า กระบวนการ หรือเรียกทับศัพท์ว่า โพรเซส (Process)

(ยรรยง เต็งอำนวย, 2533: 24) นิยามโดยทั่วไปของคำว่าโพรเซสคือ โปรแกรมที่กำลังดำเนินการอยู่ ถ้ากำหนดให้ชัดเจนกว่านี้ต้องบอกว่าโพรเซสคือ กลุ่มของช่องหน่วยความจำ (Memory cells) ซึ่งเปลี่ยนไปตามกฎเกณฑ์หนึ่ง โดยที่กฎเกณฑ์เหล่านี้เรียกว่า โปรแกรม และอุปกรณ์ที่ตีความโปรแกรมเหล่านี้คือ หน่วยประมวลผล (Processor) โพรเซสหมายถึง โปรแกรมที่อยู่ระหว่างการทำงาน ซึ่งในการทำงานจำเป็นต้องใช้ทรัพยากรต่าง ๆ ของระบบ เช่น ซีพียู อุปกรณ์รับ/ส่งข้อมูล หน่วยความจำหลัก ในระบบคอมพิวเตอร์อาจมีโพรเซสมากกว่าหนึ่งโพรเซส และแต่ละโพรเซสอาจมีสถานะการทำงานที่แตกต่างกัน ระบบปฏิบัติการจึงต้องเข้ามาจัดการเกี่ยวกับโพรเซส และจัดการทรัพยากรของระบบที่มีอยู่อย่างจำกัดให้แก่โพรเซส หน้าที่สำคัญของระบบปฏิบัติการ คือ การจัดการโพรเซส ตั้งแต่การสร้างโพรเซส การจัดลำดับการใช้ซีพียูของแต่ละโพรเซส และการทำลายโพรเซส เป็นต้น

ดังนั้นโปรแกรมจะกลายเป็นโพรเซสเมื่อเรารันไฟล์นั้นขึ้นมาและถูกโหลดขึ้นสู่หน่วยความจำ โดยหน่วยความจำ Stack เป็นพื้นที่ที่ถูกจองไว้ให้โปรแกรมที่ถูกเรียกใช้งานสำหรับเก็บฟังก์ชัน ตัวแปร และพารามิเตอร์ต่าง ๆ ของโปรแกรม ส่วนพื้นที่ Heap เป็นหน่วยความจำที่สงวนไว้ให้โปรแกรมใช้งานสำหรับเป็นหน่วยความจำชั่วคราว ซึ่งขนาดของหน่วยความจำที่ถูกร้องขอในระหว่างที่โปรแกรมกำลังทำงานอยู่ จะไม่สามารถทราบได้จนกว่าจะมีการทำงานของโปรแกรม โปรแกรมที่ถูกเรียกใช้งานสามารถขอจองหน่วยความจำที่ว่างอยู่ได้ หลังจากใช้งานเสร็จแล้วก็สามารถของยกเลิกหน่วยความจำส่วนนั้นได้



ภาพที่ 2.1 แสดงการจัดเก็บข้อมูลในหน่วยความจำ

ถึงแม้ว่าเราจะทำงานโปรแกรมเดียวกันในระบบคอมพิวเตอร์ โพรเซสจะถูกแยกออกมาสองโพรเซสและมีการสลับสับเปลี่ยนโพรเซสในการทำงาน และในบางกรณีโพรเซสอาจจะทำงานอยู่บน Virtual Machine เช่น โปรแกรมของภาษาจาวาที่เมื่อคอมไพล์ตัวโค้ดของภาษาจาวาแล้ว จะได้ไฟล์ชื่อโปรแกรมนามสกุล.class และเมื่อรันโปรแกรมนี้ขึ้นมา โพรเซสจะทำงานอยู่บนตัว Java Virtual Machine (ซึ่งจะแปลงคำสั่งต่าง ๆ เป็นภาษาเครื่อง) เพื่อการทำงานของโพรเซสต่อไป

2.2 คุณสมบัติของโพรเซส

โพรเซสแต่ละตัวจะถูกกำหนดความสำคัญขึ้น (Priority) ขณะที่โพรเซสถูกสร้างขึ้นความสำคัญนี้อาจเปลี่ยนแปลงได้หรือไม่ได้ขึ้นอยู่กับระบบปฏิบัติการ โพรเซสที่มีความสำคัญมากระบบปฏิบัติการจะให้สิทธิพิเศษมากกว่าโพรเซสที่มีความสำคัญน้อย เช่น ให้เวลาในการครอบครองซีพียู ได้นานกว่าอำนาจหน้าที่ (Authority) เป็นสิ่งที่บ่งบอกว่าโพรเซสนั้น ๆ สามารถทำอะไรได้บ้าง ใช้อุปกรณ์ชิ้นไหนได้บ้าง เป็นต้น ตัวอย่างเช่นโพรเซส A ไม่สามารถเข้าใช้ดิสก์ใด ๆ ทั้งสิ้น แต่สามารถรับข้อมูลจากทุก ๆ โพรเซสในระบบได้คุณสมบัติอื่น ๆ ที่ระบบปฏิบัติการกำหนดให้มี

2.3 สภาวะของโพรเซส (Process State)

ในขณะที่แต่ละโพรเซสกำลังทำงานอยู่ จะมีการเปลี่ยนแปลงสถานะของโพรเซสเกิดขึ้น โดยสถานะของโพรเซส (Process state) ที่สำคัญมี 5 สถานะ คือ

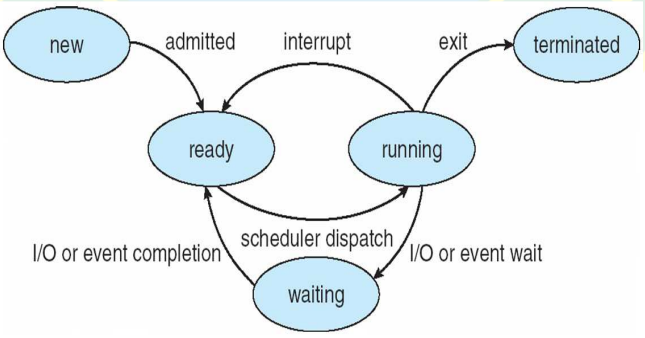
1) สถานะสร้าง (New state) หมายถึง โพรเซสใหม่กำลังถูกสร้างขึ้น

2) สถานะพร้อม (Ready state) หมายถึง สถานะที่โพรเซสพร้อมจะทำงาน หากได้รับการจัดสรรซีพียู

3) สถานะการทำงาน (Running state) หมายถึง สถานะที่โพรเซสได้ครอบครองซีพียู และทำการประมวลผล

4) สถานะรอ (Waiting state) หมายถึง สถานะที่โพรเซสรอเหตุการณ์บางอย่าง

5) สถานะเสร็จสิ้น (Terminate state) หมายถึง สถานะที่โพรเซสเสร็จสิ้นการทำงาน

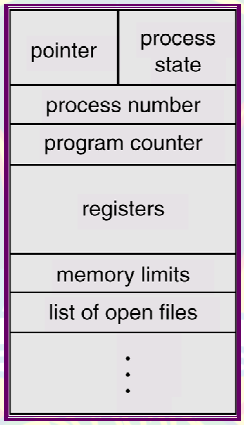


ภาพที่ 2.2 แผนภาพสถานะของโพรเซส

ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.108)

2.4 ตารางข้อมูลการประมวลผล (Process Control Block)

โพรเซส เป็นการทำงานของคำสั่งหรือกลุ่มคำสั่งของโปรแกรม และอาจมีการสลับการทำงานระหว่างโพรเซสในระบบ จึงต้องมีการบันทึกรายละเอียดของโพรเซสแต่ละโพรเซส รวมถึงข้อมูลต่าง ๆ ที่จำเป็นต้องใช้ในการทำงาน จึงใช้โครงสร้างข้อมูลพิเศษที่เรียกว่า Process Control Block (PCB) หรืออาจเรียกว่า Task Control Block โดยมีรายละเอียดของโครงสร้างข้อมูลดังนี้



ภาพที่ 2.3 โครงสร้างข้อมูลของตารางข้อมูลการประมวลผล

1) ตัวชี้ (Pointer) เป็นตัวชี้ไปยังหน่วยความจำหลักที่กันไว้สำหรับกระบวนการนั้น ๆ

2) สถานะของกระบวนการ (Process state) จะเก็บสถานะของโพรเซส ซึ่งจะแสดงสถานะปัจจุบันของโพรเซส

3) ชื่อและหมายเลขประจำตัว (Process number) ของโพรเซสต้องไม่ซ้ำกับโพรเซสอื่น

4) ตัวชี้โปรแกรม (Program counter) เก็บตำแหน่งที่อยู่ของคำสั่งโปรแกรมต่อไปที่จะถูกประมวลผล

5) รีจิสเตอร์ของหน่วยประมวลผล (CPU registers) หน่วยความจำภายในซีพียูทำหน้าที่เก็บสถานะของระบบเมื่อมีอินเทอร์รัพเกิดขึ้น ซึ่งรีจีสเตอร์จะมีข้อมูลที่ถูกเก็บอยู่ในรีจีสเตอร์ภายในโพรเซสเซอร์ ซึ่งถูกนำมาใช้ในขณะที่โพรเซสเซอร์กำลังประมวลผล

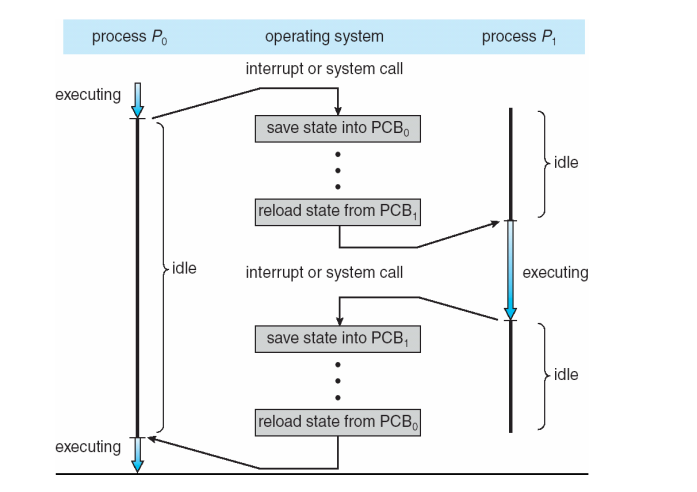
6) ข้อมูลในการจัดตารางทำงานของประมวลผลกลาง (CPU scheduling information) เก็บข้อมูลการจัดตารางเวลาของซีพียู เป็นข้อมูลแสดงลำดับความสำคัญของโพรเซสที่ถูกกำหนดโดยระบบปฏิบัติการ

7) สารสนเทศเกี่ยวกับการจัดการหน่วยความจำ (Memory management information) เก็บข้อมูลการจัดการหน่วยความจำ เป็นข้อมูลเกี่ยวกับหน่วยความจำที่ระบบปฏิบัติการกำหนดไว้ เช่น ขนาดของหน่วยความจำ ค่าของรีจีสเตอร์ ตารางหน่วยความจำเพจเทเบิล (Page table) ตารางเซกเม้นท์ (Segment table)

8) ข้อมูลทางการบัญชี (Accounting information) หมายถึงการเก็บข้อมูลการใช้ทรัพยากร ประกอบด้วย จำนวนเวลาที่ซีพียูใช้ในการทำงาน หมายเลขบัญชี หมายเลขงานหรือหมายเลขโพรเซส และอื่น ๆ

9) ข้อมูลสถานะการรับส่งข้อมูล (I/O status information) เก็บข้อมูลสถานะของ I/O อุปกรณ์ที่โพรเซสปัจจุบันใช้งานอยู่ รายการแฟ้มที่ถูกเปิดอ่าน-เขียนหรือถูกใช้งาน และอื่น ๆ

แต่ละโพรเซสจะมีบล็อกควบคุมเพื่อเก็บสถานะการทำงานของตนเองเรียกว่า PCB โดยโพรเซสแต่ละโพรเซสจะมีหมายเลขเฉพาะเพื่อใช้ในการแสดงตน PCB เป็นพื้นที่หน่วยความจำที่ระบบปฏิบัติการกำหนดไว้เพื่อเก็บข้อมูลที่สำคัญของโพรเซส ดังนั้นเมื่อระบบปฏิบัติการมอบเวลาของซีพียูให้กับโพรเซสอื่นครอบครองเพื่อประมวลผล ถ้าโพรเซสนั้นได้สลับกลับมาครอบครองเวลาของซีพียูใหม่อีกครั้ง โพรเซสจะนำข้อมูลที่อยู่ใน PCB ของตัวมันเองมาใช้เพื่อให้ทราบสถานะการทำงานเดิมของตน



ภาพที่ 2.4 แผนภาพแสดงการสับเปลี่ยนซีพียู จากโพรเซสหนึ่งไปสู่อีกโพรเซสหนึ่ง
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.109)

จากภาพที่ 2.4 สามารถอธิบายการทำงานของโพรเซส P0 และ P1 ที่ต้องการทำงานพร้อมกัน จึงจำเป็นต้องสลับการเข้าใช้ซีพียู สามารถอธิบายลำดับการทำงานได้ดังนี้

1) เมื่อโพรเซส P0 ทำงานได้ระยะเวลาหนึ่ง ระบบปฏิบัติการจะต้องสลับการทำงานจากโพรเซส P0 ไปยังโพรเซส P1 (เนื่องจากเกิดการขัดจังหวะด้วยสาเหตุใด ๆ)

2) ระบบปฏิบัติการจะทำการเก็บสถานะของโพรเซส P0 ไว้ใน PCB0

3) สลับการทำงานของซีพียูไปทำงานในโพรเซส P1 โดยใช้ข้อมูลของโพรเซส P1 ที่เก็บไว้ใน PCB1

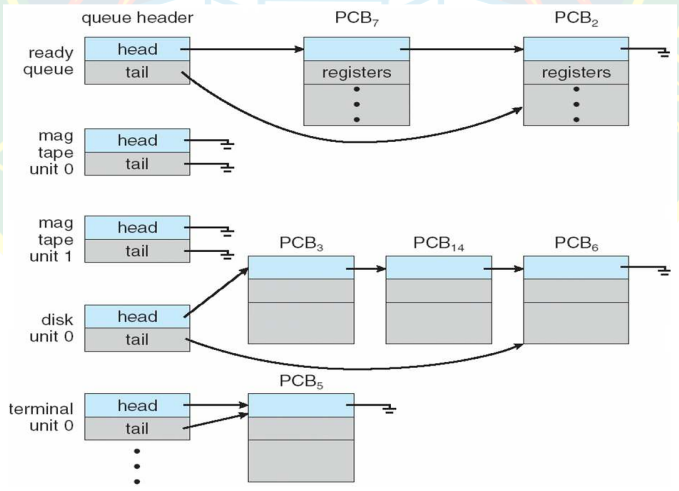
4) เมื่อโพรเซส P1 ทำงานไปแล้วช่วงระยะเวลาหนึ่ง และถูกขัดจังหวะ ระบบปฏิบัติการจะต้องบันทึกข้อมูลของโพรเซส P1 เก็บไว้ใน PCB1 และสลับไปทำงานในโพรเซสอื่นต่อไป

2.5 การจัดตารางของโพรเซส

โดยปกติการทำงานของระบบคอมพิวเตอร์แบบโมโนโปรแกรมมิ่งจะมีเพียง 1 โพรเซสเท่านั้นที่สามารถใช้งานซีพียูได้ ในปัจจุบันการทำงานของระบบคอมพิวเตอร์แบบมัลติโปรแกรมมิ่ง ที่มีหลาย ๆ โพรเซสสามารถทำงานพร้อมกันได้ ถ้ามีซีพียูเพียงตัวเดียว ระบบจำเป็นต้องให้แต่ละโพรเซสสลับกันใช้ซีพียู จุดประสงค์ของการทำงานแบบนี้คือ การพยายามที่จะทำให้หน่วยประมวลผลกลางทำงานอย่างมีประสิทธิผลสูงสุด โดยจัดให้มีโพรเซสเข้าไปทำงานในหน่วยประมวลผลตลอดเวลา แต่สำหรับระบบที่มีหน่วยประมวลผลเดียว จะมีเพียงหนึ่งโพรเซสเท่านั้นที่สามารถเข้าใช้หน่วยประมวลผลกลางได้ และในกรณีที่มีหลายโพรเซสต้องการเข้าใช้หน่วยประมวลผลกลาง งานเหล่านั้นต้องรอจนกว่าหน่วยประมวลผลกลางจะว่างลง แล้วจึงมีการจัดตารางการใช้หน่วยประมวลผลกลางใหม่อีกครั้ง

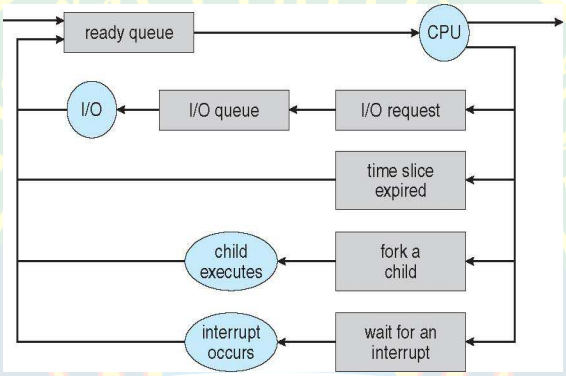
2.5.1 การจัดตารางแถวคอย (Scheduling Queue)

(พิเชษฐ์ ศิริรัตนไพศาลกุล, 2548: 75) เมื่อโพรเซสเข้าสู่ระบบจะถูกจัดให้อยู่ในแถวคอย (Job queue) ในหน่วยเก็บข้อมูลขนาดใหญ่ เมื่อโพรเซสได้เข้าสู่หน่วยความจำหลักแล้ว โพรเซสนั้นจะถูกเก็บในรายการแบบเชื่องโยง (Link list) ที่เรียกว่าแถวพร้อม (Ready queue) ซึ่งโพรเซสภายในแถวพร้อมนี้เป็นโพรเซสที่อยู่ในหน่วยความจำหลัก และพร้อมที่จะทำงาน ทั้งนี้เพียงแค่กำลังรอคอยที่จะเข้าใช้หน่วยประมวลผลกลางเพื่อทำงานต่อไป โดยที่แถวพร้อมจะเก็บตัวชี้ที่ชี้ไปยังตารางข้อมูลของโพรเซสแรกและโพรเซสสุดท้ายในแถวพร้อม และในแต่ละ PCB ก็จะมีตัวชี้ไปยัง PCB ที่อยู่ถัดไปด้วย เมื่อโพรเซสได้เข้าใช้หน่วยประมวลผลกลางและทำงานไปได้ระยะเวลาหนึ่ง โพรเซสนั้นอาจต้องหยุดทำงาน เป็นเพราะงานเสร็จหรือหยุดรอเหตุการณ์บางอย่างให้เกิดขึ้น เช่น รออุปกรณ์รับ-ส่งข้อมูล เนื่องจากมีหลายโพรเซสทำงานอยู่ในระบบ ดังนั้นโพรเซสทั้งหลายอาจต้องเข้าแถวคอยเพื่อที่จะใช้อุปกรณ์รับ-ส่งข้อมูลต่าง ๆ ซึ่งแถวคอยอุปกรณ์ดังกล่าว เรียกว่า “แถวอุปกรณ์” (Device queue) โดยอุปกรณ์แต่ละตัวจะมีแถวคอยเป็นของตัวเอง ดังภาพที่ 2.5



ภาพที่ 2.5 แสดงลำดับงานที่พร้อมทำงานและลำดับงานของอุปกรณ์ประเภทต่าง ๆ
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.111)

ในกรณีที่อุปกรณ์ตัวนั้นเป็นอุปกรณ์ที่ใช้เฉพาะงานใดงานหนึ่ง เช่น เครื่องขับเทป แถวคอยของอุปกรณ์ตัวนั้นก็จะมีเพียง 1 โพรเซสเท่านั้น แต่ถ้าเป็นอุปกรณ์ที่สามารถใช้ร่วมกันได้ เช่น จานบันทึก ภายในแถวคอยของอุปกรณ์ตัวนั้นก็อาจมีหลายโพรเซสคอยอยู่ได้ ภาพถัดไป แสดง “แผนภาพของแถวคอย” (Queuing diagram) รูปสี่เหลี่ยมผืนผ้าแทนแถวคอย 2 แบบ คือ แถวพร้อมและแถวอุปกรณ์ วงกลมแทนทรัพยากรซึ่งเป็นเจ้าของแถวคอยนั้น ๆ เส้นลูกศรแสดงทิศทางการไหลของโพรเซสในระบบ



ภาพที่ 2.6 แผนภาพแสดงแถวลำดับที่คอยใช้แทนการจัดลำดับโพรเซส
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.112)

เมื่อมีโพรเซสใดโพรเซสหนึ่งเข้าสู่แถวพร้อม โพรเซสนั้นจะรออยู่ในแถวพร้อม จนกระทั่งถูกเลือกให้เป็นผู้ใช้หน่วยประมวลผลกลาง และหลังจากที่โพรเซสได้ถูกจัดให้เข้าใช้หน่วยประมวลผลกลางแล้ว ขณะที่โพรเซสกำลังทำงานอยู่ อาจมีเหตุการณ์หนึ่งเหตุการณ์ใดเกิดขึ้น ดังนี้คือ

1) โพรเซสร้องขออุปกรณ์รับ-ส่งข้อมูล ดังนั้นจึงถูกจัดให้รอในแถวอุปกรณ์

2) โพรเซสสร้างโพรเซสย่อยขึ้นมา และรอจนโพรเซสย่อยทำงานเสร็จ

3) โพรเซสถูกขัดจังหวะโดยระบบ ทำให้ต้องหยุดการทำงาน และถูกย้ายไปไว้ในแถวพร้อมอีกครั้ง แม้ใน 2 กรณีแรก เมื่ออุปกรณ์ทำงานเสร็จหรือโพรเซสย่อยทำงานเสร็จ โพรเซสก็สามารถเปลี่ยนจากสถานะรอคอยไปเป็นสถานะพร้อม และถูกย้ายไปไว้ในแถวพร้อมในที่สุด โพรเซสหนึ่ง ๆ จะทำงานสลับกันไปเช่นนี้เรื่อย ๆ จนกว่าจะสิ้นสุดการทำงานและออกไปจากระบบ

2.5.2 การจัดตารางการทำงาน (Schedulers)

การนำโพรเซสไปต่อท้ายคิว หรือนำเอาโพรเซสออกจากคิว จะต้องมีโปรแกรมที่ช่วยกำหนดและจัดลำดับของโพรเซส เรียกว่า กำหนดการ (Scheduler) โดยแบ่งออกได้เป็น 3 ประเภท คือ

2.5.2.1 กำหนดการระยะยาว (Long-term Scheduler)

หรือเรียกอีกชื่อหนึ่งว่า Job Scheduler ทำหน้าที่เลือกโพรเซสที่รออยู่ที่หน่วยความจำสำรอง และต้องการเข้าไปทำงานที่ซีพียู ให้เข้าไปอยู่ในหน่วยความจำหลักที่คิวพร้อมเพื่อรอจนกว่าซีพียูว่าง

2.5.2.2 กำหนดการระยะสั้น (Short-term Scheduler)

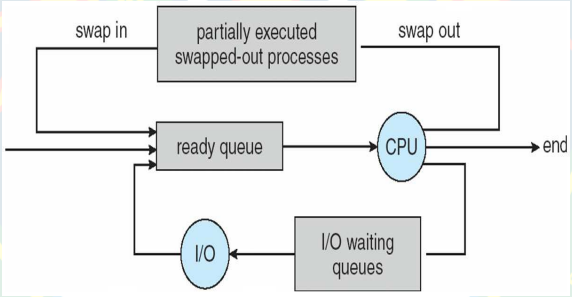
หรือเรียกอีกชื่อหนึ่งว่า CPU Scheduler ทำหน้าที่เลือกโพรเซสจากคิวพร้อมในหน่วยความจำหลัก และมอบหมายซีพียูให้กับโพรเซสแต่ละโพรเซส เป็นผลให้โพรเซสเปลี่ยนสถานะจากพร้อมเป็นทำงาน

2.5.2.3 กำหนดการระยะกลาง (Medium-term Scheduler)

สามารถถูกนำมาใช้เพิ่มเติม ในกรณีถ้ามีความจำเป็นใช้โพรเซสในระบบมัลติโปรแกรมมิ่ง เพื่อลดโพรเซสที่อยู่ในหน่วยความจำ โดยการย้ายโพรเซสออกจากหน่วยความจำและ Swap ลงดิสก์ และนำโพรเซสนั้นกลับมาใช้หน่วยความจำอีกครั้งเมื่อต้องการให้โพรเซสนั้นทำงานต่อไป

ข้อแตกต่างระหว่างกำหนดการระยะยาวและกำหนดการระยะสั้น คือ ความถี่ของการถูกเรียกใช้งาน กำหนดการระยะสั้นจะถูกเรียกใช้งานมากกว่ากำหนดการระยะยาว เนื่องจากซีพียูทำงานด้วยความเร็วสูง ทำให้โพรเซสถูกทำงานในระยะเวลาสั้นมาก (2-3 มิลลิวินาที) จากนั้นโพรเซสจะถูกขัดจังหวะเพื่อไปทำงานด้านการรับส่งข้อมูล หรืองานอาจจะเสร็จเรียบร้อยทำให้ซีพียูว่าง กำหนดการระยะสั้น จะเลือกโพรเซสต่อไปจากคิวพร้อม และมอบหมายซีพียูให้โพรเซสที่เลือกเข้ามา

โดยปกติกำหนดการระยะสั้นจะทำงานอย่างน้อยทุก ๆ 10 มิลลิวินาที ส่วนกำหนดการระยะยาวมีความถี่ในการทำงานน้อยมาก เนื่องจากการสร้างโพรเซสใหม่จำเป็นต้องใช้เวลาเป็นนาที ในการเลือกโพรเซสของกำหนดการระยะยาวจำเป็นต้องพิจารณาว่าเป็นโพรเซสที่เน้นการรับส่งข้อมูลหรือโพรเซสที่เน้นซีพียู เพื่อให้ระบบมีสัดส่วนการทำงานที่พอดี ถ้าเลือกโพรเซสที่เน้นการรับส่งข้อมูล จะทำให้คิวพร้อมว่างและกำหนดการระยะสั้นถูกเรียกใช้งานน้อยมาก ถ้าเลือกโพรเซสที่เน้นงานด้านซีพียู จะทำให้คิวรอว่าง อุปกรณ์รับส่งข้อมูลไม่ถูกใช้งาน



ภาพที่ 2.7 การเพิ่มตัวจัดลำดับระยะกลาง
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.114)

ในปัจจุบันระบบมัลติทาสกิ้ง (Multitasking) หมายถึง ระบบที่สมรรถภาพของเครื่องคอมพิวเตอร์สามารถทำงานสองโปรแกรมขึ้นไปได้ในเวลาเดียวกัน ตอนเริ่มต้นของระบบมัลติทาสกิ้งบนมือถือ จะยอมให้ทำงานเพียงหนึ่งโพรเซสเท่านั้น นอกนั้นจะให้หยุดรอไว้ก่อนโดยตัวอย่างเช่น

1) ในระบบปฏิบัติการไอโอเอส ในส่วนของการดำเนินการโพรเซสส่วนหน้า (Single foreground processes) ยอมให้ทำงานเพียงหนึ่งโพรเซส โดยควบคุมผ่านส่วนที่ติดต่อผู้ใช้งาน ในส่วน

ของโพรเซสที่ทำงานอยู่เบื้องหลัง (Multiple background processes) จะทำงานอยู่บนหน่วยความจำโดยไม่แสดงออกมาและมีข้อจำกัดในการทำงาน

2) ระบบปฏิบัติการแอนดรอยด์ โพรเซสที่ทำงานอยู่เบื้องหลังถูกใช้เพื่อปฏิบัติงานด้านต่าง ๆ การบริการยังคงดำเนินการได้ แม้ว่าโพรเซสที่ทำงานเบื้องหลังจะถูกระงับการทำงาน โพรเซสในการบริการไม่มีในส่วนของการติดต่อกับผู้ใช้ จึงใช้หน่วยความจำน้อย

2.5.3 การสลับการทำงานของซีพียู (Context Switch)

การสลับการทำงานของซีพียูหมายถึง การที่ซีพียูสลับไปประมวลผลโพรเซสอื่น โดยจะทำการบันทึกสถานะของโพรเซสปัจจุบันเอาไว้ จากนั้นจะเรียกสถานะของโพรเซสที่ถูกบันทึกไว้กลับขึ้นมาประมวลผลต่ออีกครั้ง เมื่อย้อนกลับมาทำงานเดิมซีพียูจะทำการประมวลผลโพรเซสจนเสร็จ จากนั้นก็จะสลับไปยังโพรเซสอื่นถัดไปที่อยู่ในคิว ระหว่างที่ซีพียูสลับการประมวลผลนั้นซีพียูจะว่างและไม่มีการทำงาน เวลาระหว่างนี้จะสูญเปล่า เวลาที่ใช้ในการสลับจะเปลี่ยนไปตามเครื่อง ซึ่งขึ้นอยู่กับความเร็วของหน่วยความจำ จำนวนรีจีสเตอร์ที่ถูกคัดลอก และคำสั่งพิเศษที่มีอยู่ในระบบ

ดังนั้นการสลับการทำงานของซีพียูจึงขึ้นอยู่กับการสนับสนุนของฮาร์ดแวร์ด้วย เช่น ซีพียูบางตัวมีรีจีสเตอร์อยู่หลายชุดก็จะเปลี่ยนตัวชี้ไปยังชุดของรีจีสเตอร์ แต่ถ้าหากมีจำนวนโพรเซสมากกว่า ชุดของรีจีสเตอร์ ระบบจะคัดลอกข้อมูลระหว่างรีจีสเตอร์กับหน่วยความจำ

การสลับการทำงานนี้จะเข้ามาแก้ปัญหาคอขวดของระบบ (Performance bottleneck) ซึ่งการใช้ Context switch กลายเป็นสิ่งที่เพิ่มประสิทธิภาพให้กับระบบเกิดเป็นโครงสร้างใหม่ที่เรียกว่า เธรด (Threads) ที่นำมาใช้เพื่อหลีกเลี่ยงปัญหาคอขวด ในเรื่องของเธรดจะอธิบายในบทที่ 4 ต่อไป

2.6 การดำเนินการของโพรเซส

ในระบบปฏิบัติการมีโพรเซสมากมาย สามารถทำงานได้พร้อมกัน และระบบปฏิบัติการจะมีวิธีการในการสร้างหรือทำลายโพรเซสเป็นปกติอยู่เสมอ กลไกในการสร้างและทำลายโพรเซสสามารถอธิบายลำดับชั้นของโพรเซส (Process hierarchy) ได้ดังนี้

1) เมื่อผู้ใช้ส่งงานให้กับระบบเพื่อทำงาน ระบบปฏิบัติการจะทำการสร้างโพรเซสสำหรับงานนั้นขึ้นมา

2) การทำงานของระบบปฏิบัติการก็ถือว่าเป็นงานของระบบ ดังนั้นจะมีการสร้างโพรเซสขึ้นเหมือนกัน

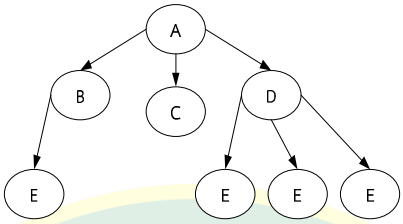
3) นอกจากนั้นโพรเซสที่ถูกสร้างขึ้นก็สามารถสร้างโพรเซสย่อยได้

4) โพรเซสที่ให้กำเนิด เราเรียกว่า โพรเซสแม่ (Parent process)

5) โพรเซสย่อยที่เกิดขึ้น เราเรียกว่า โพรเซสลูก (Child process)

6) โดยทั่วไป เมื่อโพรเซสแม่จบลง โพรเซสต่าง ๆ ที่อยู่ภายใต้ตัวมันก็จะจบลงตามไปด้วย

7) ระบบปฏิบัติการบางตัวยอมให้โพรเซสแม่จบลง โดยที่โพรเซสลูกไม่ต้องจบลงตามไปด้วย ในกรณีนี้โพรเซสลูกก็จะไม่มีโพรเซสแม่



ภาพที่ 2.8 แสดงลำดับชั้นของโพรเซส

จากตัวอย่างในภาพที่ 2.8 โพรเซส A จะมีโพรเซสลูก 3 โพรเซส คือ B, C และ D ถึงแม้ว่าโพรเซส A เป็นโพรเซสแม่ของโพรเซส B, C และ D แต่โพรเซส A ไม่ได้เป็นผู้ที่สร้างโพรเซส B, C และ D ผู้ที่สร้างโพรเซสทั้งหมดได้แก่ระบบปฏิบัติการ ซึ่งระบบปฏิบัติการจะมีโพรเซสหนึ่งทำหน้าที่สร้างและยุติโพรเซส คือ ตัวจัดคิวระยะยาว

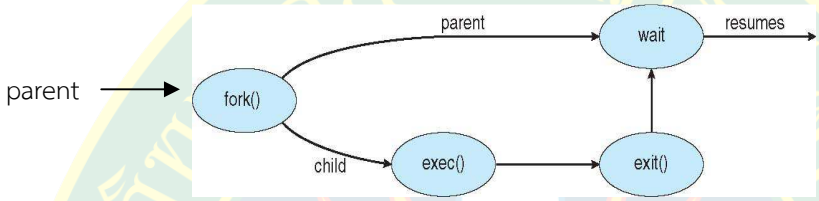
2.6.1 การสร้างโพรเซส (Process Creation)

โพรเซสใด ๆ สามารถสร้างโพรเซสใหม่ได้ด้วยการเรียกใช้คำสั่งระบบ (System command) ของระบบปฏิบัติการ หรือผ่านทาง System call ที่ชื่อ Fork (ในระบบปฏิบัติการ Unix) โพรเซสที่สร้างโพรเซสอื่นเรียกว่า โพรเซสแม่ เมื่อสร้างโพรเซสลูกแล้วสามารถทำงานต่อไปพร้อมกับโพรเซสลูก หรือหยุดรอจนกว่าโพรเซสลูกจะทำงานเสร็จ โพรเซสที่ถูกสร้างเรียกว่า โพรเซสลูก สามารถร้องขอทรัพยากรจากระบบปฏิบัติการ หรือถูกจำกัดให้ใช้ได้เฉพาะทรัพยากรของโพรเซสแม่เท่านั้น ทั้งนี้โพรเซสแม่จำเป็นต้องทราบหมายเลขโพรเซสลูกทั้งหมด

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
 pid_t pid;

  /* fork a child process */
 pid = fork();

 if (pid < 0) { /* child process */
  fprintf(stderr, “Fork Failed”);
  return 1;
}
  else if (pid ==0) { /*child process */
   execlp (“/bin/ls”, “ls, NULL);
}
else {  /* parent process */
 /* parent will wait for the child to complete */
wait (NULL);
printf(“Child Complete”);
}
   return 0;
}
```

ภาพที่ 2.9 ตัวอย่างโค้ดภาษาซีในการสร้างโพรเซส

โค้ดภาษาซีที่แสดงในภาพที่ 2.9 เป็นตัวอย่างของขั้นตอนการสร้างโพรเซสและการดำเนินการของโพรเซส จะเห็นว่ามีโปสเซสที่แตกต่างกันรันอยู่บนโปรแกรมเดียวกัน (ในบรรทัด pid = fork()) เมื่อค่า pid เป็นศูนย์ ระบบจะมีคำสั่ง System call คือคำสั่ง execlp() ซึ่งในส่วนของโปรแกรมจะเป็นการแสดงรายการไดเรกทอรี (ls เป็นคำสั่งในระบบ Unix) โพรเซสแม่จะรอจนโพรเซสลูกทำงานจนเสร็จ (คำสั่ง wait()) และระบบจะคืนข้อมูลหรือผลลัพธ์ของโพรเซสลูกไปให้โพรเซสแม่ผ่านทาง System call ที่ชื่อ Wait (Unix) และเมื่อโพรเซสลูกทำงานจนเสร็จสิ้น โพรเซสแม่จะกลับมาทำงานต่อไป



ภาพที่ 2.10 การสร้างโพรเซสโดยระบบเรียกฟังก์ชั่น fork( )
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.119)

2.6.2 การสิ้นสุดของโพรเซส (Process Termination)

ในการทำลายโพรเซส โพรเซสจะสิ้นสุดลงเมื่อสิ้นสุดการทำงานในคำสั่งสุดท้าย และแจ้งให้ระบบปฏิบัติการลบมันออกไปโดยใช้ System call ที่ชื่อ Exit (ในระบบปฏิบัติการ Unix) หรือยกเลิกโพรเซสเมื่อทำงานเสร็จสิ้น เมื่อโพรเซสแม่ทำงานเสร็จสิ้น โพรเซสลูกและโพรเซสที่ถูกสร้างโดยโพรเซสลูกจะต้องสิ้นสุดตามไปด้วย เมื่อโพรเซสแม่สิ้นสุดไปแล้วโพรเซสลูกทั้งหมดก็จะสิ้นสุดไปด้วย สิ่งที่เกิดขึ้นในลักษณะนี้เรียกว่า การสิ้นสุดเป็นขั้น ๆ (Cascading termination) นอกจากนี้โพรเซสใดโพรเซสหนึ่งสามารถเป็นสาเหตุให้โพรเซสอื่นสิ้นสุดการทำงาน โดยผ่านทาง System call ที่เหมาะสมเช่น Abort (ใช้ในระบบปฏิบัติการ Unix) โดยโพรเซสแม่สามารถยกเลิกโพรเซสลูกได้ ในกรณีที่โพรเซสลูกใช้ทรัพยากรของโพรเซสแม่จนหมด ทำให้ทรัพยากรไม่พอใช้ โพรเซสแม่ไม่ต้องการใช้โพรเซสลูกอีกต่อไป โพรเซสแม่ทำงานเสร็จสิ้นแล้ว และระบบปฏิบัติการไม่ต้องการให้โพรเซสลูกทำงานต่อไป

2.7 การสื่อสารระหว่างโพรเซส

โพรเซสที่ทำงานในระบบปฏิบัติการอาจจะเป็นโพรเซสอิสระ (Independent processes) หรือ โพรเซสที่ต้องทำงานร่วมกัน (Cooperation processes) โพรเซสอิสระ คือ โพรเซสที่ไม่มีผลกระทบต่อโพรเซสอื่นในระบบเช่น โพรเซสซึ่งไม่แบ่งข้อมูล (ชั่วคราวหรือถาวร) ให้กับโพรเซสอื่น โพรเซสที่ต้องทำงานร่วมกัน คือ โพรเซสที่มีผลกระทบต่อโพรเซสอื่นในระบบ เช่น โพรเซสที่ต้องแบ่งข้อมูลให้กับโพรเซสอื่นที่เป็นโพรเซสร่วม มีเหตุผลต่าง ๆ มากมายที่ทำให้ต้องจัดเตรียมสิ่งแวดล้อมให้กับโพรเซสที่ต้องทำงานร่วมกันดังนี้

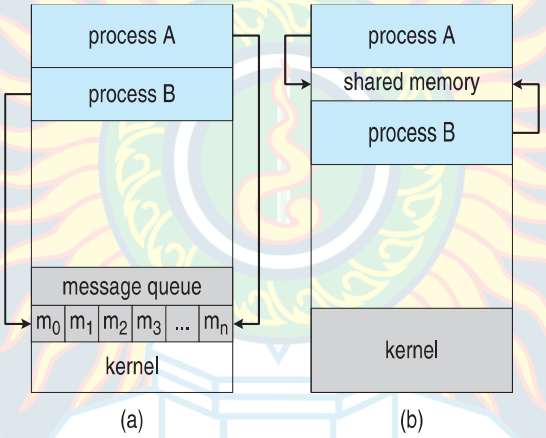
1) การร่วมกันใช้ข้อมูลข่าวสาร (Information Sharing) เมื่อผู้ใช้หลายคนสนใจข้อมูลข่าวสารชิ้นเดียวกัน (ตัวอย่างเช่น แฟ้มข้อมูลที่ถูกแชร์) จำเป็นต้องจัดเตรียมสิ่งแวดล้อม โดยอนุญาตให้เข้าถึงทรัพยากรเหล่านี้ร่วมกันได้

2) การคำนวณรวดเร็วขึ้น (Computation Speedup) ถ้าต้องการให้งานปกติสามารถทำงานได้เร็วขึ้น จำเป็นจะต้องแตกงานเหล่านั้นเป็นส่วนย่อย ๆ แล้วให้แต่ละส่วนทำงานขนานกันไป แต่ความเร็วในการคำนวณจะสูงขึ้นได้ก็ต่อเมื่อ ระบบมีอุปกรณ์ที่ใช้คำนวณหลาย ๆ ตัว เช่น มีซีพียูหลาย ๆ ตัว หรือมีหน่วยคำนวณหลาย ๆ ตัว

3) ระบบย่อย (Modularity) บางทีระบบอาจต้องการที่จะสร้างระบบให้อยู่ในรูปแบบของระบบย่อยหรือโมดูล โดยอาจแบ่งหน้าที่งานต่าง ๆ ของระบบไปเป็นหน้าที่ละหนึ่งโพรเซส

4) ความสะดวกสบาย (Convenience) ผู้ใช้แต่ละคนอาจจะมีงานหลาย ๆ งานที่ต้องทำงานในเวลาเดียวกัน เช่น ต้องการแก้ไขข้อมูล พิมพ์ข้อมูล และแปลภาษาไปพร้อม ๆ กัน ซึ่งสามารถกระทำได้ในเวลาเดียวกัน

กลไกที่สนับสนุนให้โพรเซสสามารถประสานกันได้ คือ การสื่อสารระหว่างโพรเซส (Inter Process Communication : IPC) และการประสานโพรเซส (Synchronize process) การทำงานของโพรเซสที่มีการประสานกับโพรเซสอื่น จำเป็นต้องใช้บัฟเฟอร์ โดยระบบปฏิบัติการจะต้องทำการแชร์หน่วยความจำไว้ใช้งาน และจะต้องมีกลไกที่สนับสนุนให้สามารถประสานได้ กลไกที่ว่าคือการติดต่อระหว่างโพรเซส ดังนั้นวิธีการสื่อสารระหว่างโพรเซสที่ยอมโพรเซสเหล่านั้นให้มีการแลกเปลี่ยนข้อมูลหรือสารสนเทศ พื้นฐานของ IPC มีอยู่ 2 แบบคือ แบบ Share Memory และแบบ Message Passing



ภาพที่ 2.11 แสดงรูปแบบการสื่อสาร (a) Message passing. (b) Shared memory
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.124)

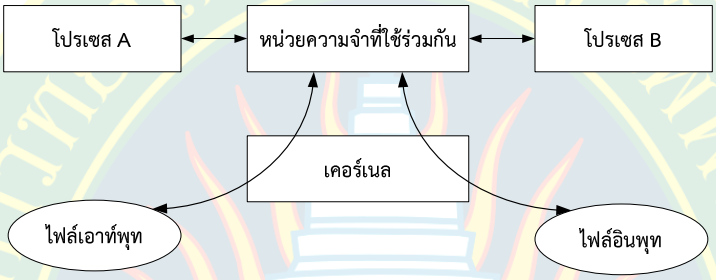
2.7.1 Shared-Memory Systems

ระบบ Share memory เป็นเทคนิคการสื่อสารข้อมูลในกระบวนการที่มีหลาย ๆ โพรเซสเข้ามาใช้หน่วยความจำที่เดียวกัน โดยจะสามารถทำได้ในระบบปฏิบัติการแบบ Multitasking การใช้งานหน่วยความจำร่วมกันเป็นปัจจัยหนึ่งที่จะทำให้โพรเซส 2 โพรเซส หรือมากกว่าใช้ข้อมูลในหน่วยความจำร่วมกันได้โดยตรง ไม่ผ่านเคอร์เนล (ออบเจ็ก IPC ที่ผ่านมาเช่น PIPE ,FIFO จะทำงานผ่านเคอร์เนล) โดยโพรเซสเหล่านั้นจะต้องจัดการการทำงานร่วมกันด้วยตัวเอง

หลักการทำงานของระบบ Shared Memory หน่วยความจำที่ถูกกำหนดให้ใช้งานร่วมกัน ปกติแล้วก็คือ ตำแหน่งแอดเดรสที่ว่างของหน่วยความจำ เพื่อกำหนดเป็นพื้นที่ใช้หน่วยความจำร่วมกัน โพรเซส

สามารถเข้ามาใช้ข้อมูลในหน่วยความจำ เพื่อการสื่อสารข้อมูลหรือแลกเปลี่ยนข้อมูลเหล่านั้น เสมือนกับว่าพื้นที่ของโพรเซสเอง การเปลี่ยนแปลงใด ๆ ที่เกิดขึ้นไม่ว่ามาจากโพรเซสใด จะมีผลต่อโพรเซสทั้งหมดที่กำลังใช้หน่วยความจำในขณะนั้น

หน่วยความจำที่ถูกใช้งานร่วมกันไม่มีหน้าที่กำหนดจังหวะการทำงานร่วมกันของโพรเซสต่าง ๆ เช่น การอ่านข้อมูลขณะที่โพรเซสอื่นกำลังเขียนข้อมูลอยู่ รวมทั้งไม่มีฟังก์ชันอื่นใดที่จะมาทำหน้าที่นี้ ดังนั้นโปรแกรมเมอร์จะต้องการทำงานนี้ด้วยตนเอง



ภาพที่ 2.12 แสดงการส่งผ่านข้อมูลระหว่างไคลเอนท์และเซิร์ฟเวอร์เมื่อใช้งานหน่วยความจำที่ใช้ร่วมกัน

ดังนั้นจึงต้องมีการประสานงานระหว่างโพรเซสเพื่อป้องกันไม่ให้มีโพรเซสใดใส่ข้อมูลลงในบัฟเฟอร์ที่เต็มแล้ว หรือโพรเซสใดพยายามดึงข้อมูลจากบัฟเฟอร์ทั้ง ๆ ที่ยังไม่มีข้อมูลอยู่ ตัวอย่างของแนวทาง การทำงานร่วมกันของโพรเซสนี้ สามารถพิจารณาจากกรณีปัญหาผู้ผลิตและผู้บริโภค (Producer Consumer Problem) ซึ่งเป็นหลักการพื้นฐานของการใช้หน่วยความจำร่วมกันระหว่างโพรเซส เช่น มีการแชร์บัฟเฟอร์เป็นส่วนความจำที่คั่นระหว่างงานสองงาน ซึ่งต้องส่งผ่านข้อมูลระหว่างกัน แต่มีความเร็วในการรับส่งข้อมูลไม่เท่ากัน งานทั้งสองจะต้องทำงานร่วมกันในลักษณะผู้ผลิตและผู้บริโภค (Producer-consumer) การทำงานร่วมกันคือ ผู้ผลิตจะต้องสามารถผลิตข้อมูลเข้าสู่บัฟเฟอร์ได้และผู้บริโภคก็สามารถบริโภคข้อมูลต่าง ๆ เหล่านั้นได้ ดังนั้นผู้ผลิตจะมีหน้าที่ใส่ข้อมูลลงในส่วนของบัฟเฟอร์ และผู้บริโภคจะนำข้อมูลออกจากบัฟเฟอร์ โดยผู้ผลิตจะไม่มีการใส่ข้อมูลทับข้อมูลเก่า และผู้บริโภคจะไม่พยายามนำข้อมูลที่ยังผลิตไม่เสร็จออกไปใช้ บัฟเฟอร์นี้จะว่างก็ต่อเมื่อไม่มีการผลิตและผู้บริโภคได้บริโภคข้อมูลจนหมดแล้ว

ประเภทของบัฟเฟอร์ที่สามารถถูกนำมาใช้มี 2 ประเภทคือ

1) บัฟเฟอร์ข้อมูลมีขนาดไม่จำกัด (Unbounded-buffer) โพรเซสที่ต้องการบริโภคข้อมูลอาจต้องคอยจากผู้ผลิต แต่โพรเซสผู้ผลิตข้อมูลสามารถผลิตข้อมูลได้ตลอดเวลา (ไม่มีวันเต็ม)

2) บัฟเฟอร์มีขนาดจำกัด (Bounded-buffer) โพรเซสผู้ผลิตอาจต้องรอถ้าบัฟเฟอร์เต็ม และโพรเซสที่ต้องการบริโภคข้อมูลอาจต้องรอถ้าบัฟเฟอร์ว่าง

เพื่อให้เกิดความเข้าใจมีตัวอย่างของโค้ดโปรแกรมภาษาซี ในการใช้ลักษณะของบัฟเฟอร์ที่มีขนาดจำกัด ในรูปแบบ Circular buffer หรือเรียกเทคนิคนี้ว่า Circular queue โดยประกาศตัวแปรที่อยู่ใน หน่วยความจำเพื่อใช้งานร่วมกัน (ตัวแปรแบบ Structure) ทั้งผู้ผลิตและผู้บริโภคดังนี้

```
#define BUFFER_SIZE 10
typedef struct {
        . . .
} item;
item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

ภาพที่ 2.13 แสดงโค้ดโปรแกรมการประกาศโครงสร้างข้อมูล

การแชร์บัฟเฟอร์ คือ ส่วนความจำที่คั่นระหว่างงานสองงาน ซึ่งต้องส่งผ่านข้อมูลระหว่างกัน ถูกประกาศเป็นอาร์เรย์แบบวงกลม และตัวแปรที่กำหนดเป็นตัวชี้ตำแหน่งของอาร์เรย์คือ in และ out โดยบัฟเฟอร์ว่างก็ต่อเมื่อค่า in ชี้ในตำแหน่งอาร์เรย์เดียวกันกับ out และ บัฟเฟอร์เต็มเมื่อตรวจสอบพบว่า ((in + 1) % BUFFER SIZE) == out หมายความว่าเมื่อข้อมูลในช่องถัดไปของค่าที่ชี้ตำแหน่งข้อมูลเข้า in มีค่าเท่ากับค่าชี้ตำแหน่งออก out

```
while (true) {
        /* produce an item in next produced */
        while (((in + 1) % BUFFER_SIZE) == out)
                ; /* do nothing */
        buffer[in] = next_produced;
        in = (in + 1) % BUFFER_SIZE;
}
```

ภาพที่ 2.14 แสดงโค้ดโปรแกรมของโพรเซสผู้ผลิต

```
item next_consumed;
while (true) {
        while (in == out)
                ; /* do nothing */
        next_consumed = buffer[out];
        out = (out + 1) % BUFFER_SIZE;
        /* consume the item in next consumed */
}
```

ภาพที่ 2.15 แสดงโค้ดโปรแกรมของโพรเซสผู้บริโภค

ตัวแปรเป็นโลคอล next_produced เป็นการสร้างข้อมูลเพื่อนำไปใส่ให้กับอาร์เรย์ และ next_consumed เป็นการนำข้อมูลจากอาร์เรย์ไปใช้

2.7.2 Message-Passing Systems

ฟังก์ชั่นระบบข่าวสารจะอนุญาตให้โพรเซสสามารถสื่อสารกับกระบวนการอื่นได้ โดยไม่จำเป็นต้องมีการใช้ทรัพยากรหรือข้อมูลร่วมกัน โดยไอพีซีจะจัดเตรียมการดำเนินงานอย่างน้อย 2 ขั้นตอน คือ Send / Receive Message สำหรับการสื่อสาร โดยอาศัยช่องทางที่เรียกว่า การเชื่อมโยงการสื่อสาร Communication Link กระบวนการที่ทำการส่งจะต้องสร้างการเชื่อมโยงกับโพรเซสที่ทำหน้าที่รับเสียก่อน ส่วนการเชื่อมโยงการสื่อสารนั้นสามารถทำได้หลายวิธี โดยอาจเป็นแบบกายภาพ (Physical) เช่น ใช้หน่วยความจำร่วมกัน ระบบฮาร์ดแวร์บัส ระบบเครือข่ายร่วมกัน หรือแบบตรรกะ Logical ก็ได้

ถ้าใช้แบบตรรกะก็สามารถทำได้หลายวิธี โดยอาศัยการเชื่อมโยงและการดำเนินการแบบ Send / Receive เข้ามาช่วย ได้แก่

1) การสื่อสารทางตรงและทางอ้อม (Direct / Indirect communication)
2) การสื่อสารแบบสมมาตรและแบบอสมมาตร (Symmetric / Asymmetric)
3) การปรับอัตรา (Buffering) แบบอัตโนมัติหรือแบบชัดเจนข่าวสารแบบขนาดคงที่หรือแปรผัน

ในการใช้งานการเชื่อมโยง (Link) เส้นทางการสื่อสารระหว่างโพรเซสนั้นจำเป็นต้องพิจารณาประเด็นคำถามดังต่อไปนี้

1) จะทำการติดตั้งสายการเชื่อมโยงได้อย่างไร
2) สายการเชื่อมโยงหนึ่งเส้น สามารถใช้สื่อสารได้มากกว่าหนึ่งโพรเซสหรือไม่
3) ต้องมีจำนวนสายการเชื่อมโยงกระบวนการกี่เส้น
4) ความจุของสายควรมีขนาดเท่าใด
5) ขนาดของข่าวสารที่ใช้สื่อสารกันจะมีรูปแบบคงที่หรือแปรผัน
6) สายการเชื่อมโยงจะเป็นแบบเดี่ยว (Unidirectional) หรือแบบคู่ (Bi-directional)

การตั้งชื่อ (Naming) กระบวนการที่ต้องการสื่อสารกับโพรเซสอื่นจะต้องมีวิธีในการอ้างอิงถึง ซึ่งสามารถสื่อสารกันได้ทั้งทางตรงและทางอ้อม โดยการสื่อสารทางตรง (Direct communication) โพรเซสที่ต้องการสื่อจะต้องใช้ชื่อที่ชัดเจนทั้งฝั่งรับ และฝั่งส่งในรูปแบบดังนี้

send (P, message) ส่งข่าวสารไปยังโพรเซส P
receive (Q, message) รับข่าวสารจากโพรเซส Q

การเชื่อมโยงการสื่อสารจะต้องมีคุณสมบัติ ดังนี้

1) ทั้งฝั่งรับและฝั่งส่งจะต้องติดตั้งการเชื่อมโยงระหว่างกันโดยอัตโนมัติ
2) การเชื่อมโยงจะทำเฉพาะคู่ของฝั่งรับและฝั่งส่งเท่านั้น
3) ฝั่งรับและฝั่งส่งจะมีการเชื่อมโยงเพียงเส้นเดียวเท่านั้น
4) สายการเชื่อมโยงสามารถใช้ได้ทั้งแบบทางเดี่ยวและทางคู่ แต่โดยทั่วไปใช้แบบทางคู่

ในการกำหนดตำแหน่งที่อยู่แบบสมมาตร ทั้งผู้รับและผู้ส่งจะต้องมีชื่อสำหรับการสื่อสารกัน ในขณะที่ตำแหน่งที่อยู่แบบอสมมาตรนั้น ผู้ส่งระบุเพียงชื่อของผู้รับเท่านั้น แต่ผู้รับไม่จำเป็นต้องระบุชื่อผู้ส่งตามรูปแบบข้างล่าง ดังนี้

send (P, message) ส่งข่าวสารไปยังกระบวนการ P

receive (id, message) รับข่าวสารที่ถูกส่งมาจากกระบวนการใด ๆ ก็ได้ โดยตัวแปร id จะแทนกลุ่มของชื่อกระบวนการที่สื่อสารกัน

ข้อด้อยของวิธีแบบสมมาตรและแบบอสมมาตรก็คือ มีข้อจำกัดเกี่ยวกับชื่อมากเกินไป ถ้ามีการเปลี่ยนชื่อของกระบวนการจะต้องแจ้งไปยังสมาชิกทั้งหมด เวลาจะแก้ไขชื่อใหม่จะต้องหาชื่อเดิมที่อ้างอิงเสียก่อน ซึ่งถือว่าไม่ยืดหยุ่น

2.7.3 การติดต่อทางตรง (Direct Communication)

การติดต่อแบบนี้แต่ละโพรเซสที่ต้องการติดต่อกันจะต้องกำหนดชื่อเฉพาะที่ใช้ในการติดต่อทั้งผู้รับและผู้ส่ง ยกตัวอย่างเช่น ถ้าต้องการส่งเมสเสจจาก A ไป B จะมีการกำหนดรูปแบบคือ

send (B, message) จะเป็นการส่งเมสเสจไปยังโพรเซส B

receive (A, message) จะเป็นการรับเมสเสจจากโพรเซส A

ลิงค์แบบนี้จะมีคุณสมบัติดังนี้

1) การสร้างลิงค์จะเป็นแบบอัตโนมัติระหว่างคู่ของโพรเซสที่ต้องการติดต่อ (ในที่นี้คือ A กับ B) โพรเซสจะทราบหมายเลขโพรเซสที่จะติดต่อด้วย

2) ลิงค์หนึ่ง ๆ จะมีความสัมพันธ์เฉพาะโพรเซสสองโพรเซสเท่านั้น

3) ระหว่างโพรเซสแต่ละคู่นั้นจะมีเพียงลิงค์เดียวเท่านั้น

4) ลิงค์นี้เป็นได้ทั้งทิศทางเดียว และสองทิศทาง แต่ปกติจะเป็นแบบสองทิศทาง

วิธีการรับ-ส่งเมสเสจแบบนี้ ระบบปฏิบัติการจะไม่ดำเนินการติดต่อให้กับโพรเซสทั้งสอง โพรเซสทั้งสองจะต้องจัดการเอง ตัวอย่างการส่งเมสเสจจากโพรเซส A ไปโพรเซส B หลังจากสร้างลิงค์เรียบร้อยแล้ว ทั้งโพรเซส A และโพรเซส B จะจองหน่วยความจำที่มีการกำหนดให้สามารถใช้ร่วมกัน (โพรเซสทั้งสองจะทราบว่าหน่วยความจำนั้นอยู่ตำแหน่งใด) โพรเซส A จะนำข้อมูลไปวางบนตำแหน่งหน่วยความจำที่จองไว้ โพรเซส B จะต้องคอยตรวจสอบว่าโพรเซส A วางข้อมูลหรือยัง ถ้ายังไม่มีการวางก็จะยังไม่ดึงข้อมูล รอจนกระทั่งโพรเซส A วางเรียบร้อยแล้วจึงจะดึงข้อมูลนั้นไปใช้งาน ส่วนโพรเซส A ก็จะต้องตรวจสอบเช่นกันว่าข้อมูลที่ตนเองวางนั้น โพรเซส B ดึงไปใช้งานหรือยัง ถ้ายังไม่ดึง โพรเซส A จะต้องรอก่อน ยังไม่มีการวางข้อมูลใหม่ เพื่อป้องกันการวางข้อมูลใหม่ทับข้อมูลเดิมซึ่งจะมีผลให้การทำงานไม่สมบูรณ์ (ข้อมูลหายไป) รอจนกระทั่งการดึงข้อมูลเรียบร้อยจึงวางข้อมูลใหม่ได้ ทำเช่นนี้เรื่อยไปจนกว่าข้อมูลจะหมด กลไกที่ใช้ในการตรวจสอบเวลาที่เหมาะสมในการรับ-ส่งข้อมูลนี้เรียกว่า Process synchronization

2.7.4 การสื่อสารทางอ้อม (Indirect Communication)

ด้วยการสื่อสารทางอ้อมข่าวสารจะถูกส่งและรับผ่านทางตู้ไปรษณีย์ หรือเรียกทับศัพท์ว่าเมลล์บ็อกซ์ (Mail box) หรือพอร์ต (Port) โดยเมลล์บ็อกซ์จะมีเลขที่ซึ่งไม่ซ้ำกันกำกับไว้ ทุกโพรเซสสามารถติดต่อสื่อสารถึงกันโดยอาศัยเมลล์บ็อกซ์ดังกล่าวนี้ ในบางกรณีอาจมีมากกว่าหนึ่งโพรเซสที่สามารถใช้เมลล์บ็อกซ์ร่วมกันได้ การปฏิบัติงานเกี่ยวกับเมลล์บ็อกซ์ที่สามารถทำได้มีดังนี้

1) การสร้างเมลล์บ็อกซ์ใหม่
2) การส่งข่าวสารผ่านเมลล์บ็อกซ์
3) การลบเมลล์บ็อกซ์

การส่ง / รับ ที่สามารถทำได้มีดังนี้

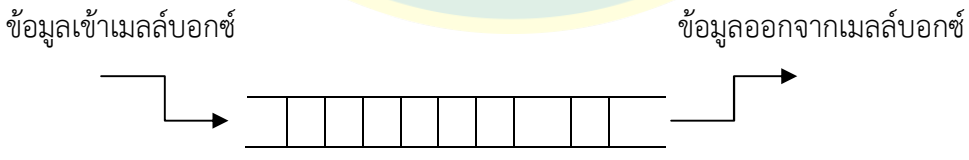
send (A, message) ส่งข่าวสารไปยังเมลล์บ็อกซ์ A
receive (A, message) รับข่าวสารจากเมลล์บ็อกซ์ A

การเชื่อมโยงการสื่อสารด้วยวิธีดังกล่าว จะต้องมีการสร้างความเชื่อมโยงไว้ระหว่างกันทั้งสองโพรเซสที่ใช้เมลล์บ็อกซ์ร่วมกัน สายเชื่อมโยงหนึ่งเส้นสามารถรองรับการเชื่อมโยงของโพรเซสได้มากกว่าสองโพรเซส โพรเซสแต่ละคู่สามารถใช้สายเชื่อมโยงเพื่อติดต่อกันได้หลายเส้นทาง สายการเชื่อมโยงอาจใช้ได้ทั้งแบบทางเดี่ยวและทางคู่

แต่เนื่องจากเมลล์บ็อกซ์มีการใช้งานร่วมกันมากกว่าหนึ่งโพรเซส จึงอาจเกิดปัญหาในการสื่อสารขึ้นมาได้ สมมติว่ากระบวนการ P1, P2 และ P3 ต่างใช้ตู้ไปรษณีย์ A ร่วมกัน เมื่อ P1 ส่งข่าวสารไปยังตู้ไปรษณีย์ A | P2 หรือ P3 จะเป็นผู้ได้รับข่าวสารจาก P1 เพื่อแก้ปัญหาดังกล่าว ระบบจะต้องมีข้อกำหนดต่าง ๆ ดังนี้

1) อนุญาตให้สายการเชื่อมโยงหนึ่งเส้นรองรับกระบวนการได้มากที่สุดเพียงสองโพรเซสเท่านั้น
2) อนุญาตให้มีเพียงกระบวนการเดียวที่สามารถรับข่าวสารจากเมลล์บ็อกซ์ ณ ขณะใดขณะหนึ่ง ให้ระบบเป็นผูตัดสินใจชี้ขาดว่าจะเลือกให้กระบวนการใดเป็นผู้รับข่าวสารนั้น และแจ้งว่าใครเป็นผู้รับไปให้ผู้ส่งทราบ
3) อนุญาตให้ระบบเลือกว่าโพรเซสใดที่จะเข้ารับเมสเสจ (ในที่นี้อาจจะเป็นโพรเซส 2 หรือโพรเซส 3 ก็ได้ แต่ไม่ใช้ครั้งละสองโพรเซสพร้อมกัน) โดยระบบจะกำหนดผู้รับไปยังผู้ส่งก่อนการส่ง

โครงสร้างของเมลล์บ็อกซ์แบบคิว เป็นโครงสร้างที่ดึงข้อมูลออกจากเมลล์บ็อกซ์ตามลำดับก่อน-หลังของข้อมูลที่ส่งเข้ามา นั่นคือข้อมูลใดส่งเข้ามาในเมลล์บ็อกซ์ก่อนก็จะถูกดึงออกไปก่อน ส่วนข้อมูลใดส่งเข้ามาภายหลังก็จะถูกดึงออกไปภายหลัง อาจเรียกการทำงานแบบนี้ว่า FIFO (First In First Out) ลักษณะโครงสร้างเมลล์บ็อกซ์แบบคิวเป็นดังภาพที่ 2.16

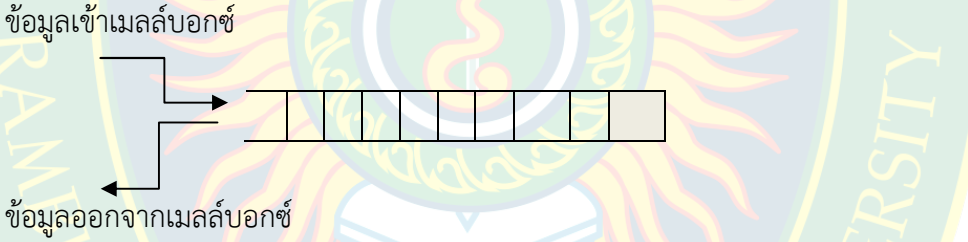


ภาพที่ 2.16 เมลล์บ็อกซ์แบบคิว

เมลล์บ็อกซ์แบบไปป์ โครงสร้างของเมลล์บ็อกซ์แบบนี้เป็นโครงสร้างที่คล้ายกับโครงสร้างแบบคิว คือ การดึงข้อมูลจะเป็นในลักษณะที่ข้อมูลใดส่งเข้ามาก่อนก็จะถูกดึงออกไปก่อน ข้อมูลใดส่งเข้ามาภายหลังจะถูกดึงออกไปใช้งานภายหลัง ข้อแตกต่างระหว่างเมลล์บ็อกซ์แบบคิวกับเมลล์บ็อกซ์แบบไปป์ คือ เมลล์บ็อกซ์แบบคิวจะมีขนาดบ็อกซ์คงที่ ถ้าใส่ข้อมูลมากเกินไปเมลล์บ็อกซ์จะเต็ม แต่ถ้าเป็นเมลล์บ็อกซ์แบบไปป์ขนาดของบ็อกซ์จะยืดหยุ่นได้ ทำให้สามารถใส่ข้อมูลในเมลล์บ็อกซ์ได้มากเท่าที่ต้องการ เมลล์บ็อกซ์จะขยายตัวโดยอัตโนมัติ ลักษณะโครงสร้างเมลล์บ็อกซ์แบบไปป์เป็นดังภาพที่ 2.17

ภาพที่ 2.17 เมลล์บ็อกซ์แบบไปป์

เมลล์บ็อกซ์แบบสแต็ก โครงสร้างของเมลล์บ็อกซ์แบบนี้เป็นโครงสร้างตรงข้ามกับเมลล์บ็อกซ์แบบคิวในการดึงข้อมูล นั่นก็คือข้อมูลใดส่งเข้าเมลล์บ็อกซ์ก่อนจะถูกดึงออกไปใช้งานภายหลัง โดยจะนำข้อมูลที่ส่งเข้ามาภายหลังออกไปใช้ก่อน อาจเรียกการทำงานแบบนี้ว่า LIFO (Last In First Out) ลักษณะโครงสร้างเมลล์บ็อกซ์แบบสแต็กเป็นภาพที่ 2.18



ภาพที่ 2.18 เมลล์บ็อกซ์แบบสแตก์

2.8 การปรับอัตรา

การจัดบัฟเฟอร์ในการสร้างลิงค์ นอกจากจะเป็นการกำหนดเส้นทางข้อมูลแล้ว ลิงค์ยังมีความจุที่เป็นตัวเลขแสดงจำนวนเมสเสจที่สามารถเก็บไว้ชั่วคราวได้ คุณสมบัตินี้อาจจะมองว่าเป็นคิวของเมสเสจที่ผูกติดลิงค์ก็ได้ การสื่อสารระหว่างโพรเซสนั้นอาจเป็นแบบทางตรงหรือทางอ้อมก็ได้ ข่าวสารจะถูกแลกเปลี่ยนโดยกระบวนการสื่อสาร ซึ่งอยู่ในกองซ้อนชั่วคราว (Temporary queue) โดยพื้นฐานแล้ว เราจะมีวิธีใช้งานกองซ้อนชั่วคราวได้ วิธีใดวิธีหนึ่งสามวิธีดังนี้

ความจุค่าศูนย์ (Zero capacity) ขนาดสูงสุดของกองซ้อนมีค่าเป็นศูนย์ หมายความว่า จะมีข่าวสารอยู่ในกองซ้อนได้เพียงชุดเดียวเท่านั้น ในกรณีนี้ผู้ส่งจะต้องบล็อกรอจนกระทั่งผู้รับได้รับข่าวสารเรียบร้อยแล้วจึงจะทำอย่างอื่นได้

ความจุค่าจำกัด (Bounded capacity) ขนาดความจุของกองซ้อนมีค่าจำกัดเท่ากับ n ดังนั้นจึงรองรับข่าวสารได้มากถึง n จำนวนเท่าความจุถ้ามีข่าวสารใหม่เข้ามาและกองซ้อนยังไม่เต็ม ก็จะเก็บไว้ในกองซ้อน ฝั่งที่ส่งก็ไม่จำเป็นต้องหยุดรอ แต่ถ้ากองซ้อนหรือสายการเชื่อมโยงเต็ม ที่ส่งจำเป็นต้องหยุดรอจนกระทั่งมีพื้นที่ว่างในกองซ้อนจึงจะสามารถวางข่าวสารได้

ความจุค่าไม่จำกัด (Unbounded capacity) ขนาดของกองซ้อนมีค่าเป็นอนันต์ ดังนั้นจึงสามารถรองรับข่าวสารได้ทั้งหมดโดยผู้ส่งไม่จำเป็นต้องหยุดรอ

ในกรณีของความจุค่าศูนย์ บางทีหมายถึง ระบบข่าวสารที่ไม่มีการปรับอัตราส่วนอีก 2 กรณีหลังเรียกว่า การปรับอัตราแบบอัตโนมัติ อาจกล่าวได้ว่าความจุศูนย์เป็นระบบที่ไม่มีบัฟเฟอร์ก็ได้ ส่วนกรณีอื่นมีการจัดบัฟเฟอร์ให้อัตโนมัติ สำหรับในกรณีที่ไม่ใช่ศูนย์ โพรเซสจะไม่รู้เลยว่าเมสเสจถึงปลายทางหรือไม่หลังจากการส่งเสร็จสิ้นไปแล้ว ถ้าข้อมูลนี้มีความสำคัญในการคำนวณ ผู้ส่งจะต้องติดต่อกับผู้รับเพื่อหาเมสเสจที่ส่งมาครั้งหลังสุด ตัวอย่างเช่น ถ้าโพรเซส P ส่งเมสเสจไปยังโพรเซส Q และสามารถทำงานได้ต่อไปเฉพาะหลังจากที่เมสเสจได้รับไปแล้ว โพรเซส P จะมีขั้นตอนดังนี้

```
send (Q, message);
receive (Q, message);

โพรเซส Q จะเอ็กซิคิวต์คำสั่ง
  receive (P, message);
  send (P, “acknowledgment”);
```

การรับส่งเมสเสจระหว่างโพรเซส P และโพรเซส Q ในลักษณะนี้เรียกว่า Asynchronous อย่างไรก็ตามยังมี 2-3 กรณีที่ไม่เข้ากลุ่มใดตามที่กล่าวมาแล้วนี้ โดยสามารถอธิบายได้ดังต่อไปนี้

1) การส่งเมสเสจของโพรเซสได้โดยไม่ต้องคอย กรณีนี้ถ้าผู้รับยังได้รับเมสเสจก่อนที่ผู้ส่งจะส่งเมสเสจอื่น เมสเสจแรกจะหายไป ข้อได้เปรียบของกรณีนี้ก็คือ เมสเสจขนาดใหญ่ไม่จำเป็นต้องสำเนาไว้หลายครั้งส่วนข้อเสียเปรียบ คือ การเขียนโปรแกรมของงานจะมีความยุ่งยาก โพรเซสเหล่านี้จำเป็นต้องมีการซินโครไนซ์แบบพิเศษเพื่อให้มั่นใจว่าเมสเสจไม่สูญหายไปไหน ทั้งผู้รับและผู้ส่งไม่ต้องคำนวณบัฟเฟอร์ของเมสเสจ

2) การส่งเมสเสจของโพรเซสจะล่าช้าออกไปจนกว่าโพรเซสจะได้รับคำตอบ วิธีการนี้นำมาใช้ในระบบปฏิบัติการที่ชื่อว่า Thoth โดยในระบบนี้เมสเสจจะมีขนาดแน่นอน (8 word) โพรเซส P ที่เมสเสจจะถูกบล็อกจนกว่าโพรเซสที่ทำหน้าที่รับเมสเสจแล้วส่งคำตอบกลับมา 8 word ในรูปแบบคำสั่ง reply(P,message) เมสเสจที่ตอบมานี้จะเขียนทับบัฟเฟอร์ดั้งเดิมของเมสเสจ ข้อแตกต่างระหว่าง send คือ คำสั่ง send จะทำให้โพรเซสที่ส่งเมสเสจเกิดการบล็อก ในขณะที่มีการตอบกลับจะยอมให้ทั้งโพรเซสที่ส่งเมสเสจ และโพรเซสที่รับเมสเสจสามารถทำงานต่อไปได้เลยโดยไม่มีการบล็อก

2.9 เงื่อนไขข้อยกเว้น

ระบบเมสเสจมีประโยชน์บนสภาพแวดล้อมระบบแบบกระจาย ซึ่งโพรเซสอาจจะอยู่บนเครื่องอื่น ๆ ในสภาพแวดล้อมที่กล่าวถึง ความน่าจะเป็นที่จะเกิดข้อผิดพลาดระหว่างการติดต่อสื่อสารระหว่าง

โพรเซสจะมากกว่าในสภาพแวดล้อมที่เป็นเครื่องเดียว สำหรับสภาพแวดล้อมที่เป็นเครื่องเดียว เมสเสจจะอยู่ในหน่วยความจำที่ใช้แชร์ข้อมูลร่วมกัน ถ้าเกิดข้อผิดพลาดขึ้นระบบโดยรวมจะล่ม แต่สำหรับในระบบแบบกระจายเมสเสจจะถูกถ่ายโอนไปตามสาย การเกิดข้อผิดพลาดที่เครื่องใดเครื่องหนึ่งไม่จำเป็นที่ระบบโดยรวมจะล่ม ทั้งระบบที่เป็นศูนย์กลางหรือระบบแบบกระจายข้อผิดพลาดอาจจะได้รับการแก้ไข ตอนนี้มาพิจารณาเงื่อนไขยกเว้นที่ระบบต้องดูแลเมสเสจ

2.9.1 การสิ้นสุดของโพรเซส

ถ้าผู้รับหรือผู้ส่งเมสเสจสิ้นสุดก่อนเมสเสจจะเอกซิคิวต์ ในสภาวะแบบนี้ทำให้เมสเสจจะถูกกำจัด ทำให้ผู้รับไม่ได้รับเมสเสจ หรือผู้ส่งไม่ได้ส่งเมสเสจ ลองพิจารณา 2 กรณีดังนี้

1) ผู้รับโพรเซส P อาจจะรอเมสเสจจากโพรเซส Q ที่สิ้นสุดไปแล้ว ถ้าไม่มีการกระทำใด ๆ เกิดขึ้นโพรเซส P จะถูกบล็อกตลอดไป ในกรณีนี้ระบบอาจจะสั่งให้โพรเซส P สิ้นสุด หรืออาจจะแจ้งให้โพรเซส P ทราบว่าโพรเซส Q สิ้นสุดไปแล้วก็ได้

2) โพรเซส P อาจจะส่งเมสเสจไปยังโพรเซส Q ที่สิ้นสุดไปแล้ว รูปแบบการจัดบัฟเฟอร์แบบอัตโนมัติจะไม่เกิดอันตรายใด ๆ โดย P ยังคงเอ็กซิคิวต์ต่อไป ถ้าโพรเซส P ต้องการทราบว่าเมสเสจของตนนั้นโพรเซส Q เอ็กซิคิวต์หรือไม่ จะต้องมีโปรแกรมพิเศษสำหรับการแจ้งให้ทราบ แต่ในกรณีที่ไม่มีบัฟเฟอร์โพรเซส P จะถูกบล็อกตลอดไปเช่นเดียวกับข้อ 1 ระบบอาจจะสั่งให้ P สิ้นสุด หรืออาจจะแจ้งให้ P ทราบว่า Q สิ้นสุดไปแล้วก็ได้

2.9.2 การสูญหายของเมสเสจ

เมสเสจจากโพรเซส P ที่ส่งไปยังโพรเซส Q อาจจะสูญหายระหว่างทางของการสื่อสารก็ได้ ซึ่งอาจจะเป็นข้อผิดพลาดด้านฮาร์ดแวร์หรือสายสื่อสาร มี 3 วิธีพื้นฐานในการจัดการเหตุการณ์นี้

1) ระบบปฏิบัติการมีหน้าที่รับผิดชอบในการตรวจสอบการสูญหายนี้ เพื่อส่งเมสเสจไปใหม่

2) โพรเซสที่ทำหน้าที่ส่งเมสเสจ มีหน้าที่รับผิดชอบในการตรวจสอบการสูญหายเพื่อส่งเมสเสจใหม่ถ้าต้องการอีก

3) ระบบปฏิบัติการมีหน้าที่รับผิดชอบในการตรวจสอบการสูญหายนี้ หลังจากนั้นจะแจ้งให้โพรเซสที่ทำหน้าที่ส่งเมสเสจที่เกิดการสูญหาย เพื่อให้ส่งเมสเสจไปใหม่

จะทำการตรวจสอบได้อย่างไรว่าเมสเสจเกิดการสูญหายไปจากระบบ วิธีการตรวจสอบที่ง่ายที่สุดคือใช้การกำหนดช่วงเวลา Timeout เมื่อเมสเสจถูกส่งออกไปจะมีสัญญาณตอบกลับมาระบบปฏิบัติการหรือโพรเซสอาจจะกำหนดช่วงเวลาที่แน่นอน โดยคาดการณ์จากสัญญาณการตอบรับที่มาถึง ถ้าช่วงเวลาเกิดการเหลื่อมก่อนที่สัญญาณการตอบรับจะมาถึง ระบบปฏิบัติการอาจจะกำหนดว่าเมสเสจนั้นสูญหายไปและจะต้องส่งเมสเสจใหม่ อย่างไรก็ตามถ้าเมสเสจไม่ได้สูญหายไปจากระบบจริงแต่เกิดการใช้เวลาในการเดินทางในเครือข่ายมาก ในกรณีนี้ระบบอาจจะต้องทำสำเนาเมสเสจที่ส่งไปยังเครือข่ายและจะต้องมีกลไกเพื่อแบ่งแยกประเภทของเมสเสจเหล่านั้นด้วย

2.10 สรุป

ปัญหาหนึ่งในการอธิบายเกี่ยวกับระบบปฏิบัติการ คือ เราจะเรียกกิจกรรมของซีพียูว่าอย่างไรดี ในระบบการทำงานแบบกลุ่มเรียกว่า Job ในระบบแบ่งปันส่วนเรียกว่า โปรแกรมผู้ใช้หรือ Task แม้แต่ระบบผู้ใช้คนเดียวอย่าง MS-DOS และ Macintosh ผู้ใช้เครื่องอาจให้หลาย ๆ โปรแกรมทำงานพร้อมกันได้โดยโปรแกรมหนึ่งเป็นโปรแกรมโต้ตอบและที่เหลือเป็นแบบกลุ่ม

เราอาจเรียกโปรแกรมที่กำลังทำงานอยู่ว่าโพรเซส การทำงานของโพรเซสต้องเป็นแบบลำดับ หรืออีกนัยหนึ่งคือ ณ เวลาใดเวลาหนึ่ง จะต้องมีหนึ่งคำสั่งที่กำลังดำเนินการอยู่ในนามของโพรเซสนี้ โพรเซสไม่ได้หมายความเพียงโปรแกรมและการทำงานเท่านั้น แต่รวมถึง Program counter (พวกรีจิสเตอร์ในซีพียู), Stack (ข้อมูลชั่วคราว), Data section (ใช้เก็บตัวแปรแบบ Global) โปรแกรมที่เป็นกระบวนการเหมือนสิ่งไม่มีชีวิต (Passive entity) เช่น คำสั่งที่เก็บไว้ในจานบันทึก ส่วนกระบวนการเสมือนสิ่งมีชีวิต (Active entity) มี Program counter ทำหน้าที่ชี้บรรทัดคำสั่งที่จะทำงานต่อไป และกลุ่มของทรัพยากรที่เกี่ยวข้อง

ถึงแม้ว่าอาจมีโพรเซสสองโพรเซสทำงานอยู่บนโปรแกรมเดียวกัน ต้องนับว่าเป็นการทำงานตามลำดับแยกจากกัน เป็นเรื่องปกติที่โพรเซสหลักจะสร้างโพรเซสย่อยหลาย ๆ โพรเซสในขณะทำงาน การสื่อสารระหว่างโพรเซสจะอยู่ในรูปแบบที่เรียกว่า การส่งและการรับ ซึ่งสามารถออกแบบในการรับ-ส่งข่าวสารเป็นแบบบล็อกเรียกว่า การประสานเวลา หรือแบบ Non-Blocking บางทีเรียกว่าแบบไม่ประสานเวลาก็ได้

การส่งแบบบล็อกฝั่งผู้ส่งจะต้องหยุดรอจนกว่าข่าวสารที่ถูกส่งไปถึงที่รับแล้ว ซึ่งรับโดยโพรเซสหรือตู้ไปรษณีย์ก็ได้ แล้วมีการตอบรับจากผู้รับ การส่งแบบ Non-Blocking send ผู้ส่งไม่จำเป็นต้องการการตอบรับจากผู้รับ การรับแบบ Blocking receive ฝั่งผู้รับจะต้องหยุดรอจนกว่าจะได้รับข่าวสารอย่างครบถ้วน การรับแบบ Non-Blocking receive ฝั่งผู้รับไม่ต้องรอจนกระทั่งส่งเสร็จ สามารถเรียกใช้ข่าวสาร ซึ่งอาจได้ข่าวสารที่ถูกต้องหรือเป็นค่าว่าง (null) ก็ได้ การประสานเวลาของโพรเซสจะได้ศึกษาในบทต่อไป

แบบฝึกหัดท้ายบทที่ 2

จงตอบคำถามต่อไปนี้

1) จงอธิบายและเขียนแผนภาพสถานะของโพรเซสในแต่ละสถานะมาโดยสังเขป

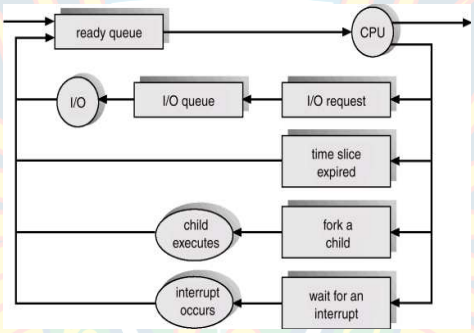
2) จงอธิบายแต่ละองค์ประกอบของ PCB (Process Control Block) มาอย่างน้อย 5 องค์ประกอบ พร้อมยกตัวอย่างการทำงาน

3) จงอธิบายการสลับการทำงานระหว่างโพรเซส

4) จงบอกความแตกต่างของ Job queue, Ready queue และ Device queue

5) จงอธิบายเหตุผลอะไรบ้างโพรเซสที่ต้องแบ่งข้อมูลให้กับโพรเซสอื่นที่เป็นโพรเซสร่วม ที่ทำให้ต้องจัดเตรียมสิ่งแวดล้อมให้กับโพรเซสที่ต้องทำงานร่วมกันดังนี้

6) จงใช้แผนภาพนี้อธิบายการทำงานของโพรเซสร่วมกับสถานะของโพรเซส



7) จงเขียนแผนภาพพร้อมอธิบายการสื่อสารกันของโพรเซสแบบส่งผ่านข้อความ และการใช้หน่วยความจำร่วมกัน

8) ในการปรับอัตราในการสื่อสารระหว่างโพรเซส จงอธิบายว่ามีประโยชน์อย่างไร

9) จงอธิบายการพิจารณาเงื่อนไขข้อยกเว้นที่ระบบต้องดูแลเมสเสจ มีเหตุผลอะไรบ้าง

เอกสารอ้างอิง

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

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

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

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

Andrew S. Tanenbaum, Maarten van Steen. (2002). **Distributed Systems Principles and Paradigms.** Prentice Hall, Pearson Education International.

M.Sc TU. Blog. Retrieved April 2, 2014 from http://msctu.blogspot.com/2010/03/