แผนบริหารการสอนประจำบทที่ 3

หลักการโปรเซส (Process Concept)

วัตถุประสงค์

1. สามารถอธิบายความหมายของโปรเซส
2. สามารถอธิบายลักษณะการใช้หน่วยความจำ
3. สามารถอธิบายสภาวะของโปรเซสได้
4. สามารถอธิบายการสร้างและการยุติโปรโซสได้
5. สามารถอธิบายการบริหารจัดการโปรเซสได้
6. สามารถอธิบายการสลับการทำงานได้
7. สามารถอธิบายการสื่อสารระหว่างโปรเซสได้

เนื้อหา

1. หลักการพื้นฐานและความหมาย (Process Concept)
2. ลักษณะการใช้หน่วยความจำ (Memory Usage)
3. สภาวะของโปรเซส (State of Process)
4. การสร้างและยุติโปรเซส (Process Creation and Termination)
5. การบริหารจัดการโปรเซส (Process Management)
6. การสลับการทำงานของโปรเซส (Process Scheduling)
7. การสื่อสารระหว่างโปรเซส (Interprocess communication: IPC)
8. การสื่อสารในระบบรับ-ให้บริการ (Communication in Client-Server Systems)

กิจกรรมการเรียนการสอน

1. บรรยาย
2. ฝึกปฏิบัติการระบบลีนุกซ์
3. วิดีโออนิเมชั่นการทำงานคอมพิวเตอร์
4. ค้นคว้าด้วยตัวเอง

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน
2. สไลด์การสอน
3. โปรแกรม google class room
4. โปรแกรม facebook
5. โปรแกรม VMware
6. โปรแกรมระบบปฏิบัติการลีนุกซ์ Ubuntu

การวัดผลและการประเมินผล

1. แบบฝึกหัดท้ายบท
2. การถามตอบในชั้นเรียน
3. ใบงาน
4. สอบเก็บคะแนน

บทที่ 3

หลักการโปรเซส (Process Concept)

ในบทนี้จะกล่าวถึงหลักการของโปรเซส (process concept) ถึงแม้คำว่า process จะมีความหมายว่า กระบวนการ การอธิบายตลอดทั้งบทนี้จะใช้คำว่า “โปรเซส” ในการอ้างอิง เนื่องจากการใช้คำนี้ มีนัยมากกว่าคำว่า “กระบวนการ” ดังเหตุผลที่จะอธิบายในลำดับถัดไป

3.1 หลักการพื้นฐานและความหมาย (Process Concept)

ระบบปฏิบัติการในยุคแรกมีหลักการคือให้คอมพิวเตอร์ประมวลผลจากโปรแกรมเพียงโปรแกรมเดียวจนกระทั่งเสร็จสิ้นงาน ซึ่งแตกต่างอย่างสิ้นเชิงจากหลักการในระบบปฏิบัติการยุคใหม่ที่อนุญาต และมีความสามารถจัดการให้หลายโปรแกรมบรรจุเข้าทำงานได้พร้อมกัน โดยมีกระบวนการจัดแบ่งช่วงเวลาในการประมวลผล และแบ่งทรัพยากรระหว่างโปรเซส

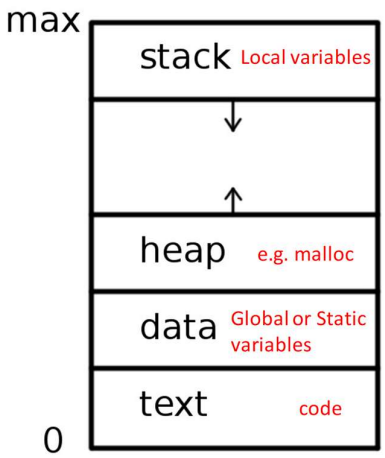
เมื่อมีการอ้างอิงถึงโปรแกรม ก็จะหมายถึงกลุ่มคำสั่งที่ถูกติดตั้งไว้ในหน่วยเก็บข้อมูลของระบบคอมพิวเตอร์ ซึ่งยังไม่ถูกเรียกใช้งานแต่อย่างใด เปรียบเสมือนหุ่นยนต์ที่มีความสมบรูณ์ แต่ไม่ได้เปิดปุ่มทำงาน เมื่อใดที่โปรแกรมถูกเรียกใช้งาน ซึ่งอาจจะเป็นการเรียกใช้โดยระบบปฏิบัติการเอง หรือเป็นการเรียกใช้จากผู้ใช้ โปรแกรมนั้น ๆ จะถูกโหลดเข้าสู่หน่วยความจำหลัก และเข้าครอบครองทรัพยากรบางส่วนที่ระบบปฏิบัติการจัดสรรให้ โปรแกรมนี้จึงถูกเรียกว่า “โปรเซส” (process) ซึ่งพร้อมที่จะถูกลำเลียงคำสั่งเข้าสู่หน่วยประมวลผล ดังนั้นโปรเซสจึงมีนัยของวัตถุพร้อมด้วยทรัพยากรที่กำลังทำงานไปตามคำสั่งที่ถูกกำหนดไว้ เปรียบเสมือนหุ่นยนต์ที่ถูกเรียกใช้งานโดยเปิดปุ่มทำงานนั้นเอง

3.2 ลักษณะการใช้หน่วยความจำ (Memory Usage)

โปรเซสเมื่อถูกโหลดเข้าสู่หน่วยความจำแล้วจะสามารถแบ่งเป็นส่วนดังรูปที่ 3-1 ชุดคำสั่งโปรแกรม (program code) ที่โปรแกรมเมอร์ได้เรียบเรียงไว้ซึ่งได้ผ่านกระบวนแปลภาษาแล้วจนกลายเป็นคำสั่ง (instruction) ที่เครื่องคอมพิวเตอร์เข้าใจ ถูกโหลดเข้าไว้ในส่วนของ text section ตำแหน่งของคำสั่งจะถูกบรรจุให้กับรีจีสเตอร์ชื่อว่า Program Counter (PC) เพื่อจะนำคำสั่งดังกล่าวเข้าสู่การประมวลผล บริเวณหน่วยความจำที่ถูกครอบครองโดยโปรเซสอีกส่วนหนึ่งคือ Stack section ซึ่งเป็นบริเวณเก็บข้อมูลชั่วคราวสำหรับฟังก์ชันต่าง ๆ ที่โปรแกรมเมอร์กำหนดเช่น ค่าพารามิเตอร์ ตัวแปรใช้ภายในฟังก์ชัน (local variables) และค่าตำแหน่งเรียกกลับ (return address) เมื่อออกจากฟังก์ชัน บริเวณหน่วยความจำ stack จะถูกจองไว้

ล่วงหน้าตามความต้องการใช้ของโปรแกรมและจะไม่สามารถขยายในช่วงเวลาดำเนินงาน (run-time) ได้ หากบริเวณหน่วยความจำดังกล่าวถูกใช้จนหมดจะทำให้โปรเซสเกิดทำงานผิดพลาดเรียกว่าเกิด stack overflow

ส่วนถัดไปคือ data section เป็นบริเวณที่ใช้เก็บค่าตัวแปรที่โปรแกรมเมอร์ต้องการประกาศให้ใช้ได้ทั่วไปเรียกว่าตัวแปรสาธารณะ (global variables) และตัวแปรที่ค่าไม่เปลี่ยนแปลง (static variables) ตัวแปรเหล่านี้โปรเซสอื่น ๆ จะสามารถเข้าใช้ได้เช่นกัน ส่วนสำคัญอีกส่วนที่โปรเซสใช้หน่วยความจำคือ heap section เป็นบริเวณที่โปรเซสทำการจองพื้นที่ให้เมื่อมีการร้องขอเกิดขึ้นเช่น คำสั่ง malloc ในภาษา C ซึ่งสามารถทำการร้องขอขณะดำเนินงาน (run-time) ได้ บริเวณ heap สามารถขยายและลดได้แบบพลวัตร ทั้งนี้ต้องทำการจองเมื่อต้องการใช้และคืนเมื่อเลิกใช้งาน หากโปรแกรมเมอร์บริหารหน่วยความจำส่วนดังกล่าวไม่มีประสิทธิภาพ อาจเป็นสาเหตุทำให้หน่วยความจำของระบบคอมพิวเตอร์ไม่เหลือที่ว่าง และทำให้คอมพิวเตอร์หยุดการทำงานได้



รูปที่ 3-1: โปรเซสในหน่วยความจำ

3.3 สภาวะของโปรเซส (State of Process)

สภาวะของโปรเซส (State of process) คือสภาพของโปรเซสที่เป็นอยู่ในขณะหนึ่ง โดยโปรเซสสามารถเปลี่ยนแปลงสภาวะได้เป็นสภาวะหนึ่งสถาวะใดจากทั้งหมด 5 สภาวะ คือ

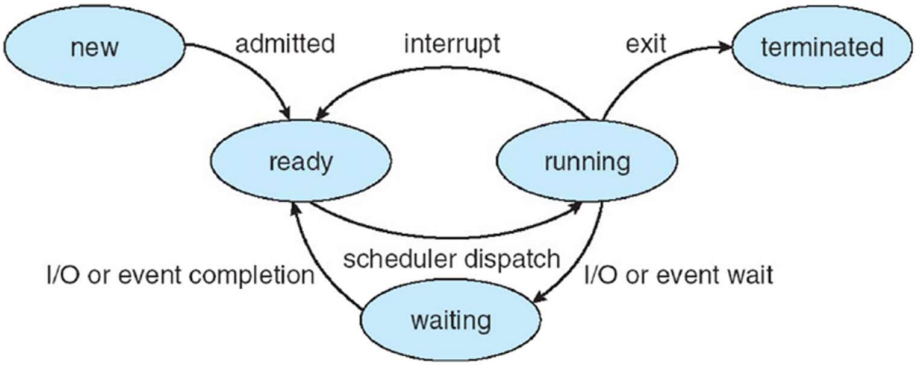
1) New: โปรเซสอยู่ในสภาวะถูกสร้าง ซึ่งเป็นสภาวะที่โปรแกรมในหน่วยเก็บข้อมูลถูกเรียกใช้และโหลดเข้าสู่หน่วยความจำหลัก

2) Ready: โปรเซสอยู่ในสภาวะพร้อม และรอให้คำสั่งถูกนำเข้าสู่หน่วยประมวลผล (Processor)

3) Running: โปรเซสอยู่ในสภาวะประมวลผลชุดคำสั่ง

4) Waiting: โปรเซสอยู่ในสภาวะรอให้มีเหตุการณ์อื่นเกิดขึ้นเช่น รอสัญญาณจากอุปกรณ์อินพุทเอาท์พุท เมื่อทำงานเสร็จสิ้น

5) Terminated: โปรเซสอยู่ในสภาวะเสร็จสิ้นการประมวลผล



รูปที่ 3-2: ไดอะแกรมสภาวะของโปรเซส

ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

ไดอะแกรมในรูปที่ 3-2 แสดงความเชื่อมโยงจากสภาวะหนึ่งไปสู่สภาวะหนึ่งของโปรเซส เมื่อโปรเซสถูกสร้าง และโหลดเข้าสู่หน่วยความจำเรียบร้อยแล้ว จึงเข้าสู่สภาวะพร้อม (ready) และรอให้ระบบปฏิบัตินำเข้าสู่หน่วยประมวลผล เมื่อโปรเซสเข้าสู่การประมวลผลสภาวะของโปรเซสจึงเปลี่ยนเป็นสภาวะกำลังทำงาน (running) ในช่วงเวลานี้โปรเซสอาจถูกขัดจังหวะ ดังนั้นโปรเซสจึงย้อนกลับมีสภาวะพร้อมเพื่อรอการประมวลต่อไป หรือในขณะทำงานโปรเซสอาจทำการเรียกใช้งานอุปกรณ์อินพุทเอาท์พุท ซึ่งมีการทำงานช้ากว่าซีพียู อยู่มาก ดังนั้นโปรเซสจึงต้องเข้าสู่สภาวะคอยเวลาให้อุปกรณ์ดังกล่าวทำงานเสร็จสิ้น ในสภาวะสุดท้ายคือ สภาวะสิ้นสุดการทำงาน (terminated) ซึ่งโปรเซสจะถูกหยุดการทำงานอย่างสมบูรณ์และคืนค่าหน่วยความจำให้ระบบ

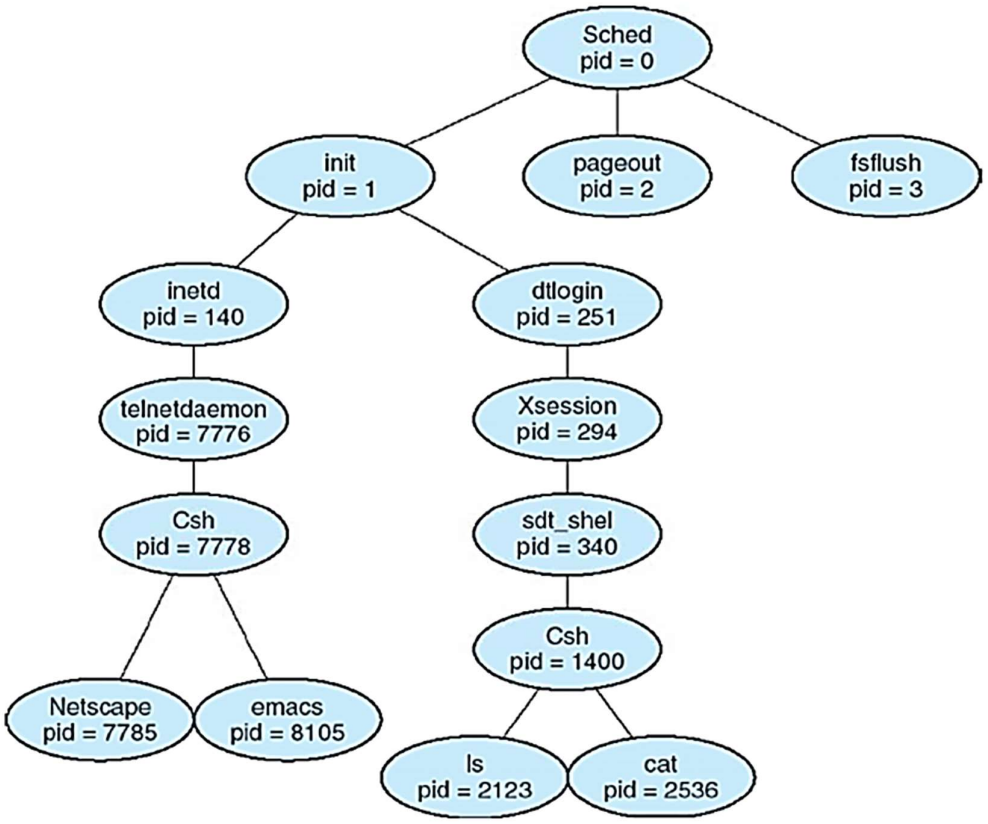
3.4 การสร้างและยุติโปรเซส (Process Creation and Termination)

ระบบปฏิบัติการในยุคปัจจุบันสามารถให้โปรเซสหลายโปรเซสทำงานในเวลาพร้อมกันได้ ดังนั้นโปรเซสสามารถถูกสร้างและยุติได้ตลอดเวลา ในส่วนนี้อธิบายวิธีการสร้างและยุติโปรเซสในรายละเอียด

3.4.1 กระบวนการสร้างโปรเซส (Process creation)

ระบบปฏิบัติการมักใช้รหัสประจำตัวโปรเซส (process identifier: pid) เจาะจงโปรเซสหนึ่ง ๆ ซึ่งเป็นตัวเลขจำนวนเต็มเพียงตัวเลขเดียวในระบบปฏิบัติการ โปรเซสหนึ่งที่อยู่ในระบบแล้วยังสามารถสร้าง

โปรเซสใหม่ให้เกิดขึ้นได้ตามที่ต้องการ ผ่านการเรียกใช้ระบบ (system call) ชื่อ CreateProcess() โดยโปรเซสผู้สร้างจะเรียกว่าโปรเซสแม่ (parent process) และโปรเซสที่ถูกสร้างเรียกว่าโปรเซสลูก (child process) นอกจากนี้โปรเซสลูกก็ยังสามารถสร้างโปรเซสอื่น ๆ ต่อไปได้ จึงเกิดเป็นโครงสร้างต้นไม้ (tree structure) ดังรูปที่ 3-3 จะเห็นว่าแต่ละโปรเซสมีรหัส (pid) เป็นตัวกำกับไว้ และโปรเซสแม่ก็อาจจะมีโปรเซสลูกได้หลายโปรเซส ลำดับการประมวลผลอาจเป็นได้สองกรณีคือ โปรเซสแม่รอจนกระทั่งโปรเซสลูกประมวลเสร็จสิ้นและยุติการทำงาน หรือการประมวลผลคู่ขนานกันไป



รูปที่ 3-3: โครงสร้างต้นไม้ของโปรเซสในระบบปฏิบัติการ Solaris

ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

3.4.2 กระบวนการยุติการทำงานโปรเซส (Process termination)

การยุติการทำงานของและออกจากระบบของโปรเซสเกิดขึ้นเมื่อโปรเซสทำงานเสร็จสิ้นแล้ว จากนั้นระบบปฏิบัติการจะทำการลบโปรเซสนั้นด้วย system call ชื่อ exit() เพื่อคืนทรัพยากรต่าง ๆ ให้กับระบบส่วนกลางอีกครั้ง การเสร็จสิ้นการประมวลผลของโปรเซสมักจะมีการส่งค่าผลลัพธ์กลับไป (ปกติเป็นค่า

จำนวนเต็ม) โดยเฉพาะโปรเซสลูกจะส่งค่าผลลัพธ์กลับไปให้โปรเซสแม่ อย่างไรก็ตามการยุติการทำงานของโปรเซสอาจเกิดขึ้นจากเหตุอื่นได้เช่น การสั่งโปรเซสลูกให้หยุดการทำงานโดยโปรเซสแม่ หรือการสั่งหยุดการทำงานโดยผู้ใช้โดยตรง โดยที่การสั่งโปรเซสหยุดการทำงานกระทันหันเช่นนี้จะกระทำผ่าน system call ชื่อ TerminateProcess()

3.5 การบริหารจัดการโปรเซส (Process Management)

เนื่องระบบปฏิบัติการยินยอมให้มีโปรเซสอยู่ในระบบหลายโปรเซสในเวลาเดียวกัน ดังนั้นระบบปฏิบัติการจึงมีมาตรการในการควบคุมโปรเซสเหล่านี้ เครื่องมือที่ใช้ในระบบปฏิบัติการคือ แต่ละโปรเซสจะถูกควบคุมในรูปของ Process Control Block (PCB) ซึ่งแสดงไว้ในรูปที่ 3-4 โปรเซสแต่ละโปรเซสจะมี PCB เป็นของตัวเอง โดย PCB เป็นส่วนในหน่วยความจำที่เก็บค่าต่าง ๆ ที่สำคัญของโปรเซสเอาไว้ โดยมีรายละเอียดดังนี้

1) Process id: รหัสโปรเซส

2) Process state: สภาวะของโปรเซสในขณะนั้น

3) Priority: สถานะความสำคัญของโปรเซส

4) Program counter: ระบุตำแหน่งของคำสั่งถัดไปที่จะประมวลผลของโปรเซส

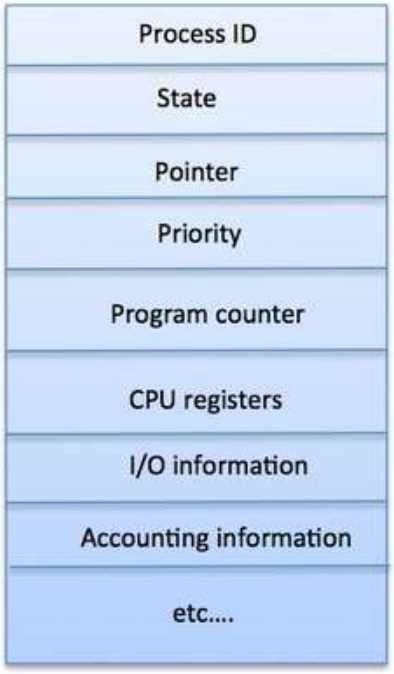
5) CPU registers: ค่าต่าง ๆ ในรีจีสเตอร์ที่ใช้โดย ซีพียู ซึ่งจะมีบทบาทเมื่อโปรเซสถูกขัดจังหวะ เพื่อคืนค่าได้อย่างถูกต้อง

6) I/O status info: เป็นรายการของทรัพยากรที่จัดสรรให้โปรเซส เช่น ไฟล์

7) CPU-scheduling info: เป็นข้อมูลเพื่อช่วยในการจัดการสลับงาน (scheduling)

8) Memory-management info: เป็นข้อมูลเพื่อช่วยในการจัดการหน่วยความจำ

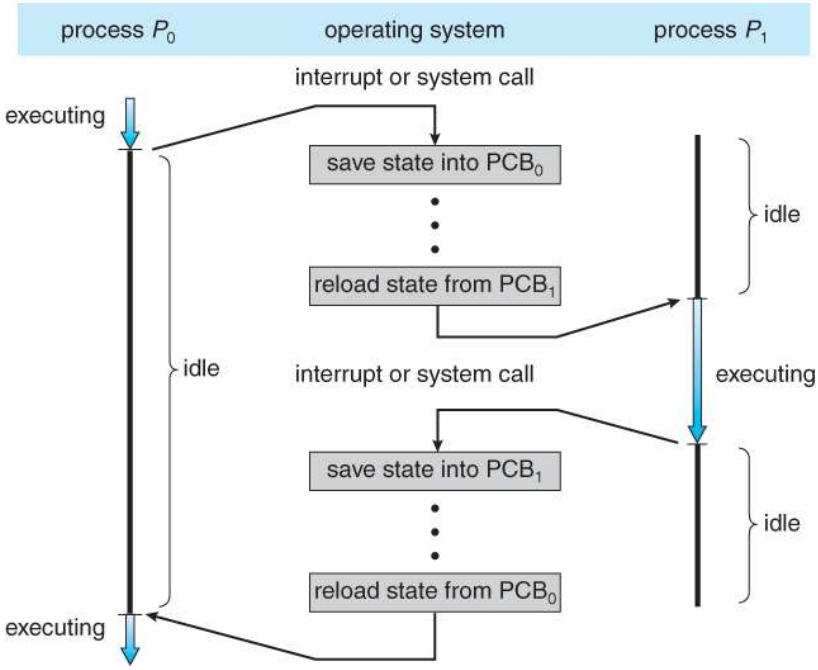
9) Accounting info: เป็นข้อมูลบันทึกการใช้ทรัพยากรเช่น ปริมาณการใช้ ซีพียู และอินเตอร์เน็ตเพื่อประโยชน์ในการจัดทำบัญชี



รูปที่ 3-4: Process Control Block (PCB)

ที่มา: (Operating System Tutorial, 2017)

พิจารณารูปที่ 3-5 จะเห็นการนำ PCB มาใช้ควบคุมการทำงานของโปรเซส P0 และ P1 ซึ่งแทนโปรเซสใด ๆ ที่กำลังทำงานอยู่ในระบบ เริ่มต้น P0 กำลังดำเนินงานจนกระทั่งถูกขัดจังหวะโดย P1 ระบบปฏิบัติการจะทำการบันทึกค่าต่าง ๆ ณ ขณะนั้นลงใน PCB_0 ค่าที่ถูกบันทึกลงใน PCB_0 ซึ่งเป็นค่าจากการคำนวน ตำแหน่งของคำสั่งที่ และข้อมูลที่จำเป็น ค่าเหล่านี้จำเป็นต่อการกลับมาทำงานต่อในคราวถัดไป เมื่อบันทึก PCB_0 จนเสร็จ จึงทำการโหลดค่าต่าง ๆ จาก PCB_1 เข้าสู่กระบวนการประมวลผลของ ซีพียู ต่อไป จากนี้ P1 จะถูกประมวลผลจนกระทั่งถูกขัดจังหวะ ระบบปฏิบัติการก็จะคืนค่า PCB_0 กลับสู่หน่วยประมวลผลเพื่อให้ P0 ได้รับการประมวลผลต่อไป เป็นเช่นนี้สลับกันระหว่างโปรเซส ดังนั้นสรุปการใช้งานของ PCB ได้ว่าเป็นส่วนเก็บสภาพแวดล้อมของโปรเซสพักไว้ขณะหนึ่ง เพื่อระบบปฏิบัติการสามารถกระทำการสลับเปลี่ยนการทำงานของโปรเซส



รูปที่ 3-5: ไดอะแกรมแสดงการใช้ PCB ในการควบคุมโปรเซส

ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

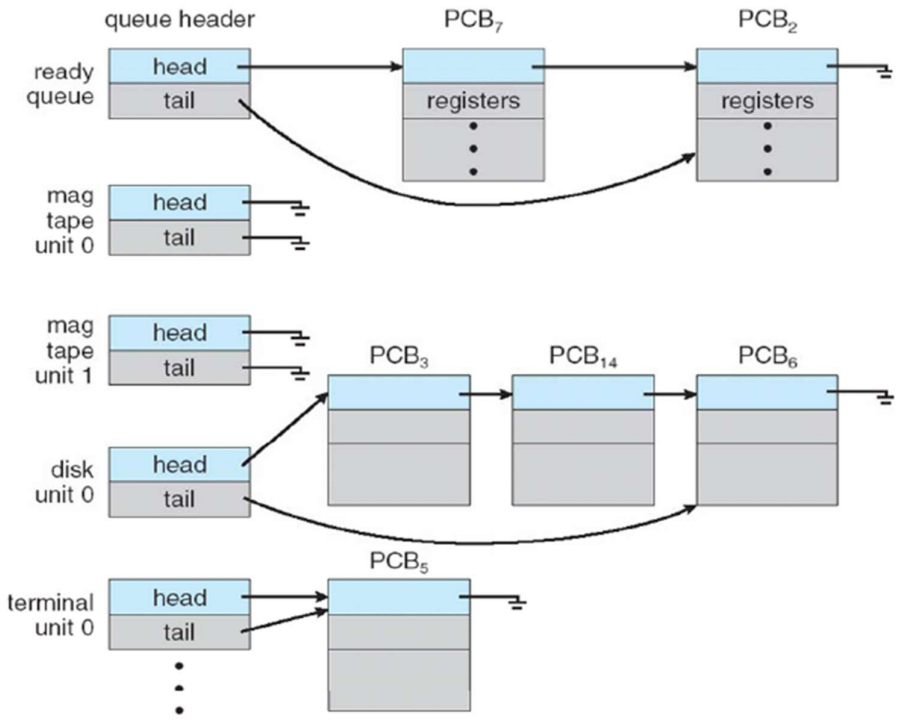
3.6 การสลับการทำงานของโปรเซส (Process Scheduling)

การสลับโปรเซสมีเป้าหมายเพื่อให้ผู้ใช้งานสามารถทำงานได้หลาย ๆ โปรแกรมในเวลาเดียวกัน โดยการบริหารการทำงานของ ซีพียู ให้เต็มประสิทธิภาพ ข้อเท็จจริงคือหน่วยประมวลผล ซีพียู 1 หน่วยสามารถประมวลผลได้ครั้งละ 1 คำสั่งในเวลาหนึ่ง ๆ แต่เมื่อมีการสลับงานอื่น ๆ เข้ามาประมวลในช่วงที่ ซีพียู ยังว่างจากการรอส่วนอื่น ๆ ทำงานนั้นโปรเซสปัจจุบัน จะทำให้ ซีพียู สามารถสลับการประมวลผลให้แก่โปรเซสอื่นได้ด้วยความสามารถของ ซีพียู มีการทำงานที่รวดเร็ว และสามารถสลับโปรเซสได้หลายโปรเซส ดังนั้นผู้ใช้จึงมีความรู้สึกถึงการทำงานได้หลาย ๆ งานพร้อมกัน ทั้งนี้การสลับการทำงานของโปรเซสจะอยู่ภายใต้คิว

3.6.1 คิวสลับโปรเซส (Scheduling queue)

โปรเซสทุกตัวที่พร้อมสำหรับการประมวลผล ต้องถูกบรรจุสู่คิวซึ่งมีชื่อเรียกว่า ready queue เพื่อรอให้ ซีพียู ประมวลผล สภาวะของโปรเซสเหล่านี้จะถูกแสดงเป็นสภาพ “ready” โดยโปรเซสที่รออยู่ใน ready queue เป็นลักษณะของ PCB ของโปรเซสนั้น ๆ เชื่อมโยงต่อกันเป็น link-list โดยมีต้นทางเป็นหัวคิว (queue header) ซึ่งประกอบด้วยส่วน head เก็บค่าตำแหน่งชี้ไปยัง PCB ของโปรเซสลำดับแรกของคิวซึ่งเป็นต้นขบวน และส่วน tail เก็บค่าตำแหน่งชี้ไปยัง PCB ของโปรเซสสุดท้ายในคิว ดังแสดงในรูปที่ 3-6

นอกจาก ready queue สำหรับโปรเซสกำลังรอรับการประมวลผลโดย ซีพียู แล้ว ยังมีคิวสำหรับอุปกรณ์อินพุทเอาท์พุท (I/O queue) สำหรับโปรเซสที่ต้องการเข้าใช้อุปกรณ์ดังกล่าว เช่น mag-tape queue คิวสำหรับอุปกรณ์ชนิดเทป disk queue คิวสำหรับโปรเซสเข้าใช้อุปกรณ์ดิสก์ และ terminal queue คิวสำหรับโปรเซสเข้าใช้จอคอมพิวเตอร์



รูปที่ 3-6: ไดอะแกรม Ready Queue และคิวสำหรับอุปกรณ์อินพุทเอาท์พุต แบบ Link-List

ที่มา: (Silberschatz, Galvin, & Gagne, 2010)

3.6.2 การจัดการคิวสลับโปรเซส (queueing)

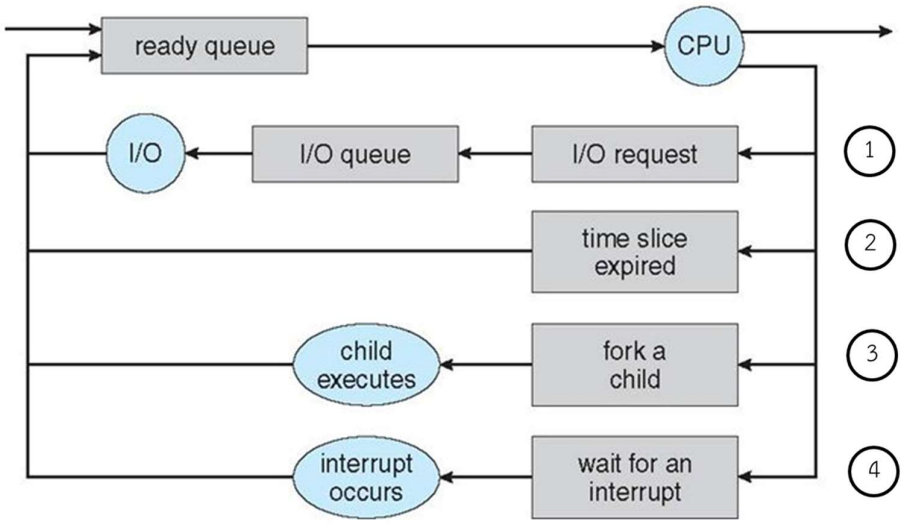
กระบวนการจัดการคิวใน ready queue เพื่อนำสู่หน่วยประมวลผล ซีพียู และการสลับนำโปรเซสพลัดเปลี่ยนกันประมวลผลแสดงไว้ในรูปที่ 3-7 จากรูปแบ่งออกเป็น 4 เหตุการณ์ โดยสมมุติให้มีโปรเซส 4 โปรเซสรออยู่ใน ready queue คือ P0, P1, P2 และ P3 ซึ่งเหตุการณ์อาจเกิดขึ้นได้ดังนี้

เหตุการณ์ที่ (1) โปรเซสที่ 0 (P0) ได้ถูกนำเข้าสู่หน่วยประมวลผล ซีพียู เป็นอันดับแรก จากคำสั่งในโปรแกรมของ P0 มีการเรียกใช้อุปกรณ์อินพุทเอาท์พุท ระบบปฏิบัติการจึงนำ P0 เข้าสู่ I/O queue จะเห็นว่า P0 เข้าสู่คิวของอุปกรณ์ และทำงานกับอุปกรณ์ดังกล่าวจนกระทั่งบรรลุผลโดยไม่ยุ่งเกี่ยวกับ ซีพียู แต่อย่างใด เมื่อ P0 ทำงานกับอุปกรณ์จนเสร็จสิ้นจึงย้อนกลับเข้าสู่ ready queue อีกครั้งเพื่อรอ ซีพียูประมวลผล

เหตุการณ์ที่ (2) โปรเซสที่ 1 (P1) ถูกระบบปฏิบัติการนำเข้าสู่หน่วยประมวลผล ซีพียู ภายหลังจาก P0 เข้าสู่คิวการใช้อุปกรณ์ (I/O queue) โดย P1 ได้รับการประมวลผลจนกระทั่งหมดเวลา ซึ่งแต่ละโปรเซสได้รับอนุญาตประมวลผลช่วงระยะเวลาหนึ่งในแต่ละรอบ (time lice) ดังนั้น P1 จึงถูกขัดจังหวะและนำไปต่อคิว ณ ready queue เพื่อรอประมวลผลรอบถัดไป

เหตุการณ์ที่ (3) โปรเซสที่ 2 (P2) เข้าสู่การประมวลผลเมื่อ P1 ออกจาก ซีพียู แล้ว เมื่อได้รับการประมวลผล P2 จำเป็นต้องสั่งการให้สร้างโปรเซสลูก และ P2 ต้องรอให้โปรเซสลูกทำงานเสร็จสิ้นเสียก่อน ดังนั้น ซีพียู จึงปล่อย P2 เข้าสู่ waiting queue เมื่อโปรเซสลูกทำงานเสร็จ P2 จะกลับไปยัง ready queue เพื่อรอประมวลผลอีกครั้ง

เหตุการณ์ที่ (4) โปรเซสที่ 3 (P3) เข้าสู่การประมวลผลหลังจาก P2 ออกไปยัง waiting queue P3 ได้รับการประมวลผลจนกระทั่งเกิดการขัดจังหวะ ซึ่งอาจจะเป็นการขัดจังหวะจากโปรเซสที่มี priority ที่สูงกว่า ดังนั้น P3 จะเข้าสู่ waiting queue เพื่อรอให้มีสัญญาณให้ทำงานต่อไปได้ P3 จึงจะได้นำเข้าสู่ ready queue เพื่อรับการประมวลผลในรอบถัดไป



รูปที่ 3-7: ไดอะแกรมการจัดการคิวสลับโปรเซส

ที่มา: (Silberschatz, Galvin, & Gagne, 2010, p. 108)

3.6.3 ผู้จัดการสลับโปรเซส (schedulers)

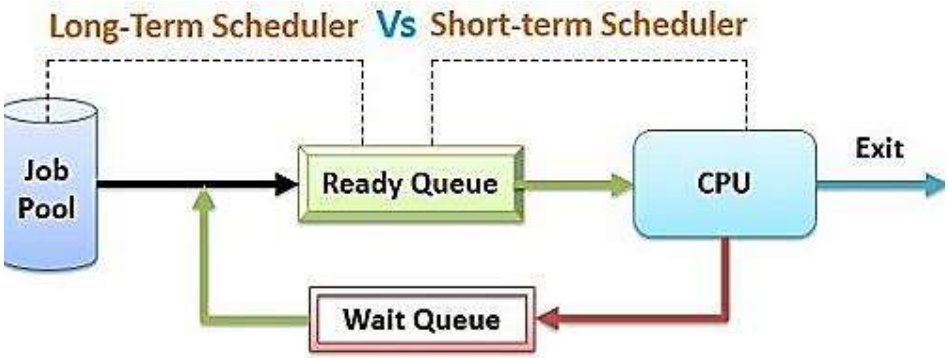
ผู้จัดการสลับโปรเซส (schedulers) มีภาระกิจในการเลือกโปรเซสภายในคิวเข้าสู่หน่วยประมวลผล ซีพียู อย่างเหมาะสม วิธีการเลือกโปรเซสมาประมวลผลไม่จำเป็นต้องเป็นแบบเรียงลำดับตามคิว

หรือ First in First out (FIFO) เสมอไป ในทางตรงข้ามผู้จัดการสามารถลือกโปรเซสได้ตามเงื่อนไขอื่น ๆ เช่น ระดับความสำคัญของโปรเซส (Priority) เป็นต้น ผู้จัดการสลับโปรเซสมีด้วยกัน 3 ประเภทคือ

1) Long-term scheduler หรือ job scheduler: ทำการเลือก processes จาก job pool และโหลดเข้าสู่ ready queue ควบคุมการทำงานหลาย ๆ งาน (Multi-programing) ให้มีประสิทธิภาพ มีหน้าที่จัดการเวลาในการประมวลผลแต่ละ process โดยหากมี processes เข้าสู่ ready queue มากจนเกินไปจะทำให้สัดส่วนของเวลาในการประมวลแต่ละ process (time slice) น้อยลง ดังรูปที่ 3-8

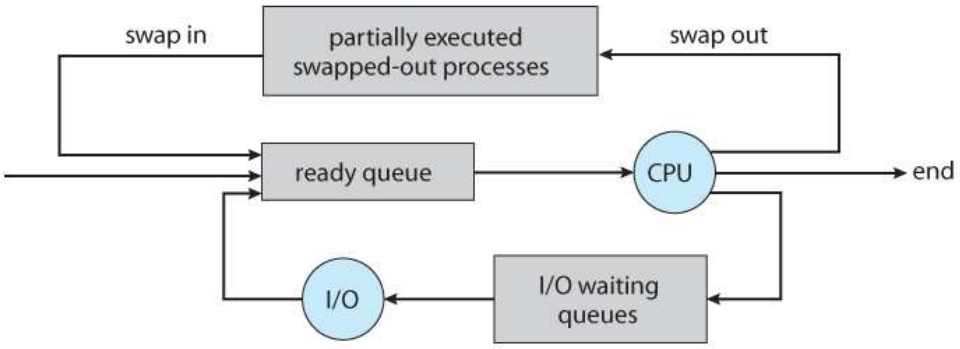
2) Short-term scheduler หรือ CPU scheduler: มีหน้าที่เลือก process จาก ready queue นำเข้าสู่หน่วยประมวลผล ซีพียู ได้อย่างมีประสิทธิภาพ โดยพิจารณาจากการใช้ ซีพียู ให้เต็มประสิทธิภาพ (CPU utilisation) หรือเวลารอคอยใน ready queue (waiting time) นอกจากนั้นยังมีหน้าที่จัดการเมื่อมีการขัดจังหวะจาก process อื่นที่มี priority สูงกว่า หรือได้รับสัญญาณจากอุปกรณ์อินพุดเอาท์พุท (รูปที่ 3-8)

3) Medium-term scheduler: มีหน้าที่ลดภาระการประมวลผลหลายโปรเซสโดยไม่มีความจำเป็น โปรเซสที่อยู่ในสภาวะรอคอย (waiting) ซึ่งอาจจะรอสัญญาณจากฮาร์ดแวร์หรือผู้ใช้เป็นเวลานาน ผู้จัดการนี้จะทำการรย้ายโปรเซสออกจากหน่วยความจำหลัก ไปเก็บไว้ในหน่วยความจำสำรองชั่วคราว เพื่อเพิ่มพื้นที่ให้กับหน่วยความจำหลัก จากนั้นรอจังหวะนำโปรเซสดังกล่าวกลับมาประมวลอีกครั้งอย่างเหมาะสม เรียกกระบวนการนี้ว่า swap out – swap in ดังในรูปที่ 3-9



รูปที่ 3-8: Long-term และ Short-term Schedulers

ที่มา : (Difference Between Long-Term and Short-Term Scheduler in OS, 2016)



รูปที่ 3-9: Medium-term Scheduler

ที่มา: (Silberschatz, Galvin, & Gagne, 2010, p. 109)

3.6.4 การปรับเปลี่ยนบริบท (Context Switch)

เมื่อการทำงานของคอมพิวเตอร์ ต้องรองรับการทำงานหลาย ๆ งานในเวลาเดียวกัน จึงเกิดการสลับการประมวลผลของโปรเซส การสลับงานหนึ่งสู่อีกงานหนึ่งมีความจำเป็นต้องเปลี่ยนบริบท (context) จากโปรเซสที่กำลังประมวลผลสู่โปรเซสใหม่ที่เข้ามาทดแทน บริบทดังกล่าวเช่นค่าต่าง ๆ ในรีจีสเตอร์ของหน่วยประมวลผลซึ่งเก็บค่าปัจจุบันในระหว่างการประมวลผล สภาวะของโปรเซส และตำแหน่งของคำสั่งถัดไปที่ต้องประมวลซึ่งบันทึกไว้ในรีจีสเตอร์ PC บริบทเหล่านี้มีความสำคัญจำเป็นต้องถูกเก็บไว้ใน PCB ประจำโปรเซสเพื่อสามารถนำกลับคืนมาใช้เมื่อถึงรอบประมวลผลครั้งถัดไป ดังนั้นก่อนที่โปรเซสใหม่จะถูกประมวลผล บริบท ของโปรเซสปัจจุบันจะถูกบันทึกลงใน PCB ของโปรเซสปัจจุบัน จากนั้นจึงนำบริบทจาก PCB ของโปรเซสใหม่เข้าสู่หน่วยประมวลผล ลักษณะดังกล่าวนี้เรียกการปรับเปลี่ยนบริบท Context switch

3.7 การสื่อสารระหว่างโปรเซส (Interprocess communication: IPC)

การสื่อสารระหว่างโปรเซสที่กำลังทำการประมวลผลอยู่ในระบบนั้นอาจเป็นได้ ทั้งโปรเซสอิสระ (independent process) และโปรเซสไม่อิสระ (dependent process) โปรเซสอิสระนั้นคือโปรเซสที่ไม่มีเกี่ยวข้องกับโปรเซสอื่นเลย เช่น ไม่มีผลกระทบ ไม่มีการใช้ข้อมูลร่วมกัน และไม่ใช้ผลการประมวลผลจากโปรเซสอื่น ส่วนประเภทโปรเซสที่ไม่อิสระคือโปรเซสที่มีความเกี่ยวข้องกับโปรเซสอื่น ๆ ตัวอย่างเช่น ผลลัพธ์มีผลกับโปรเซสอื่น หรือข้อมูลต้องใช้ร่วมกับโปรเซสอื่นเป็นต้น

สาเหตุที่โปรเซสในระบบมีความจำเป็นในการสื่อสารกันนั้นมีรายประการดังต่อไปนี้

1) Information sharing: เพื่อการแบ่งข้อมูล เช่นการเข้าใช้ไฟล์เดียวกันจากหลายผู้ใช้

2) Computation speedup: เพื่อเพิ่มความเร็วการทำงาน การเพิ่มความเร็วจำเป็นต้องแบ่งงานออกเป็นงานย่อยหลาย ๆ ชุด เพื่อให้งานบางอย่างทำงานคู่ขนานกัน อย่างไรก็ตามการเพิ่มความเร็วนั้นฮาร์ดแวร์ต้องรองรับด้วยเช่นต้องมี ซีพียู หลายชุด

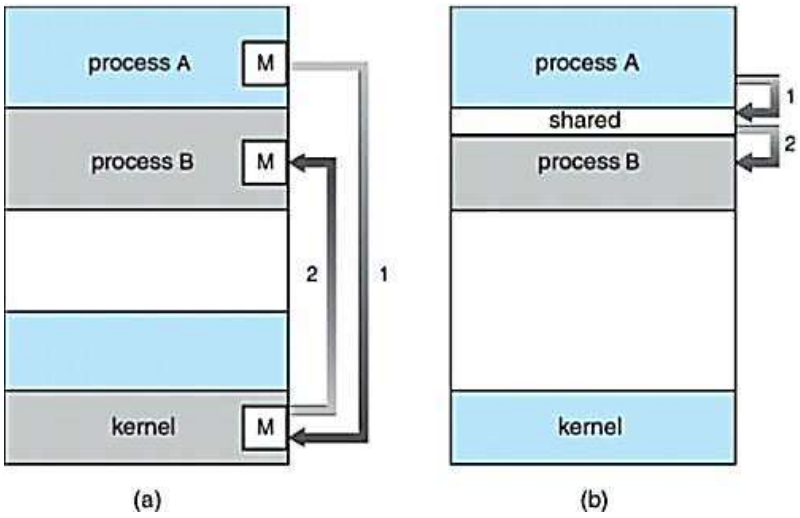
3) Modularity: เพื่อจัดโครงสร้างการประมวลผลออกเป็นชุด (modular fashion) เช่น process สามารถสร้าง thread ออกเป็นหลายชุด

4) Convenience: เพื่อรองรับการทำงานของผู้ใช้หลาย ๆ งาน เช่นผู้ใช้หนึ่งคนอาจทำการแก้ไขเอกสาร พิมพ์งาน ในขณะเดียวกัน

กลไกของการสื่อสารระหว่างโปรเซส (IPC mechanism) มีด้วยกัน 2 วิธี คือ shared memory และ message passing

1) Shared memory model ระบบจะจัดสรรพื้นที่ในหน่วยความจำเรียกว่า shared region (รูปที่ 3-10(b)) เพื่อให้โปรเซสที่เกี่ยวข้องเข้ามาแลกเปลี่ยนข้อมูลซึ่งกันและกันผ่านการอ่านและเขียนได้ โปรเซสเหล่านี้จะมีการตกลงเพื่อใช้บริเวณหน่วยความจำที่จัดสรรไว้ร่วมกัน ดังนั้นเมื่อโปรเซสหนึ่งกระทำการสิ่งใดบนพื้นที่นี้โปรเซสอื่นจะเห็นสิ่งเดียวกัน เช่น P0 สร้างและเขียนไฟล์ a.txt โปรเซส P1 ก็จะเห็นและเข้าถึงข้อมูลไฟล์ดังกล่าวได้เช่นเดียวกัน อย่างไรก็ตามวิธีการแบบนี้ เป็นวิธีการที่มีความซับซ้อน แต่สามารถทำงานได้รวดเร็ว และเหมาะกับการแลกเปลี่ยนข้อมูลจำนวนมาก

2) Message passing model: เป็นการสื่อสารอาศัยรูปแบบการส่งข้อความเพื่อแลกเปลี่ยนข้อมูลระหว่างโปรเซสโดยไม่ได้มีการใช้พื้นในหน่วยความจำร่วมกันแต่อย่างใด การรับส่งข้อความจำเป็นต้องกระทำผ่านฟังก์ชันอย่างสองชนิดคือ send() และ receive() นอกจากนี้การส่งต้องระบุโปรเซสปลายซึ่งมักใช้ pid เป็นตัวกำหนดผู้รับ การใช้วิธีการนี้นอกจากจะสื่อสารกันระหว่างโปรเซสภายในระบบคอมพิวเตอร์เดียวกันแล้ว ยังสามารถสื่อสารระหว่างโปรเซสที่อยู่ต่างเครื่องได้ผ่านระบบเครือข่าย เมื่อการสื่อสารลักษณะนี้มีการส่งและรับข้อมูลจึงจำเป็นต้องมีสื่อกลางนำพาข้อความที่บรรจุข้อมูลนั้นเดินทางจากต้นทางไปยังปลายทางได้ การสื่อสารภายในคอมพิวเตอร์เดียวกัน สื่อกลางคือเคอร์เนลของระบบคอมพิวเตอร์ (รูปที่ 3-10(a)) แต่หากการสื่อสารเกิดขึ้นระหว่างคอมพิวเตอร์สื่อกลางก็คือเครือข่ายนั้นเอง



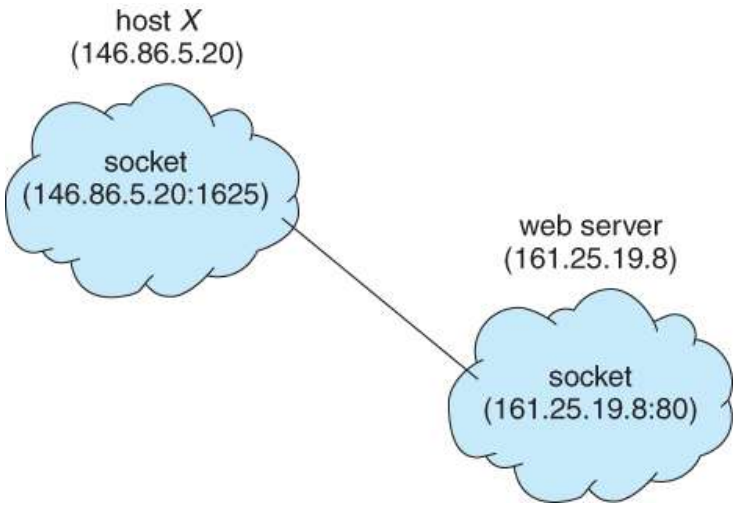
รูปที่ 3-10: รูปแบบการสื่อสารระหว่างโปรเซส (a) Massage passing (b) Shared memory

3.8 การสื่อสารในระบบรับ-ให้บริการ (Communication in Client-Server Systems)

การสื่อสารกันระหว่างโปรเซสไม่ได้เกิดขึ้นเฉพาะภายในคอมพิวเตอร์เดียวกันเท่านั้น แต่ยังมีการสื่อสารของโปรเซสระหว่างคอมพิวเตอร์ที่เชื่อมต่อกันในเครือข่ายดังที่กล่าวไว้ข้างต้น ซึ่งการสื่อสารเป็นรูปแบบรับ-ให้บริการ (Client-Server) การสื่อสารลักษณะนี้มี 3 รูปแบบคือ Sockets, Remote Procedure Calls (RPCs) และ Pipes

3.8.1 ซ็อกเก็ต (Sockets)

Socket ถูกนิยามให้เปรียบเสมือนช่องรับส่งข้อมูลสำหรับการสื่อสาร ซึ่งช่องทางการสื่อสารนี้จะถูกกำหนดขึ้นให้เป็นคู่กันระหว่างฝั่งผู้รับ (client) และฝั่งผู้ให้ (server) โดย client ถูกนิยามให้เป็นผู้ขอใช้บริการจาก server ซึ่งเป็นผู้ให้บริการ เช่นให้บริการข้อมูลเวปไซต์ เป็นต้น Socket ประกอบด้วย IP address และ port number ต่อเรียงกันโดยมีสัญลักษณ์ “:” ขั้นกลาง IP address จะเป็นสิ่งบ่งบอกถึงเครื่องคอมพิวเตอร์ในเครือข่ายซึ่งไม่ซ้ำกัน ในส่วน port number จะเป็นการระบุช่องทางเข้าถึงบริการการ การสื่อสารในรูปแบบ client-server นี้ ฝั่ง server จะคอยรับการติดต่อขอใช้บริการจากฝั่ง client ที่ช่องทางที่รู้จักกันโดยทั่วไป (well known port) โดย well known ports เป็นตัวเลขอยู่ในช่วง 1- 1024 เช่น FTP (21) , HTTP (80) และ Telnet (23) ส่วนในฝั่ง client สามารถกำหนดช่องทางขึ้นโดยมีเลขที่มากกว่า 1024 ขึ้นไป ทั้งนี้ client ต้องแจ้งตัวเลข port ให้แก่ฝั่ง server ทราบด้วย เพื่อตอบกลับมาได้อย่างถูกต้อง



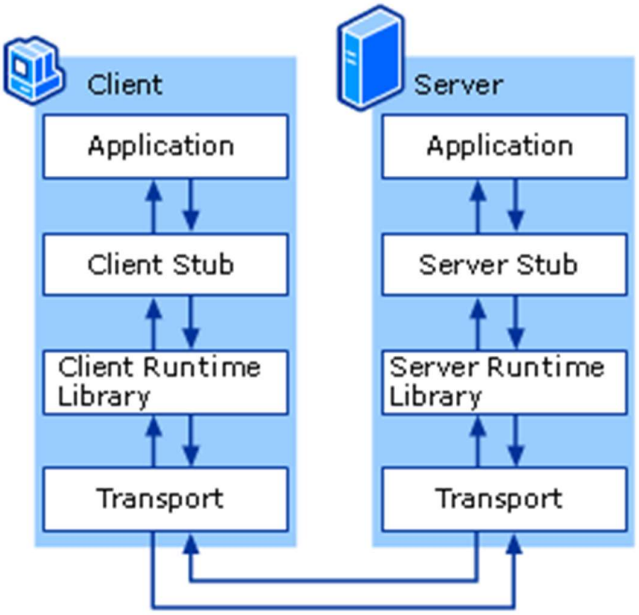
รูปที่ 3-11: การสื่อสารโดยใช้ socket

ที่มา: (Silberschatz, Galvin, & Gagne, 2010, p. 129)

ตัวอย่างจากรูปที่ 3-11 แสดงการสื่อสารโดยใช้ socket Host X ซึ่งเป็น client มีหมายเลข IP address 146.86.5.20 ต้องการข้อใช้บริการดูข้อมูลเวปไซต์จากเครื่อง server หมายเลข IP address 161.25.19.8 โปรเซสของ host X จึงกำหนดหมายเลข port ขึ้นมาหนึ่งหมายเลขคือ 1625 ไว้เป็นช่องทางสื่อสาร และร้องขอใช้บริการไปยังผู้ให้บริการเวปไซต์ web server ที่ช่องทาง port หมายเลข 80 ซึ่งเป็น well-known port ผู้ให้บริการ web server ส่งข้อมูลกลับไปตามที่ร้องขอมาไปยังเครื่องปลายด้วย port ที่แจ้งมา (1625) ก็จะทำให้โปรเซสของทั้งฝั่งสื่อสารกันได้

3.8.2 การเรียกการทำงานระยะไกล (Remote Procedure Calls : RPCs)

RPC มีความคล้างคลึงกับการสื่อสารระหว่างโปรเซสภายในเครื่องคอมพิวเตอร์เดียวกัน (IPC) ในรูปแบบ message-passing model เพียงแต่เป็นการกระทำส่งข้อความข้ามเครือข่ายไปยังคอมพิวเตอร์เครื่องอื่น ข้อความถูกจัดให้อยู่ในรูปแบบที่กำหนดที่สามารถส่งผ่านเครือข่ายได้ และส่งผ่านไปยัง port ซึ่งเป็นเลข เพื่อใช้สำหรับรับส่งข้อมูลระหว่างกัน เมื่อ client ต้องการติดต่อไปยัง server จะทำการเรียกใช้ RPC system call โดยมี stub เป็นผู้จัดการรูปแบบต่าง ๆ ของข้อมูล และ port ที่ต้องใช้ จากนั้นจะส่งข้อความไปยัง server ดังรูปที่ 3-12



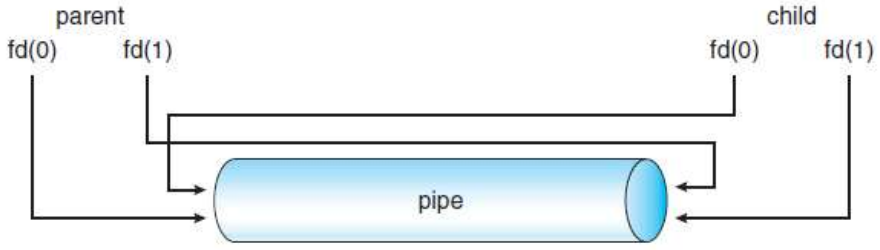
รูปที่ 3-12: การสื่อสารโดยใช้ RPC

ที่มา: (Microsoft, 2003)

3.8.3 ท่อ (Pipe)

Pipe ทำงานเปรียบเสมือนท่อที่เชื่อมต่อระหว่างโปรเซสด้าน client กับโปรเซสด้าน server ข้อมูลถูกรับส่งภายในท่อ ทั้งนี้ Pipe มี 2 ชนิดคือ ordinary pipe และ named pipe

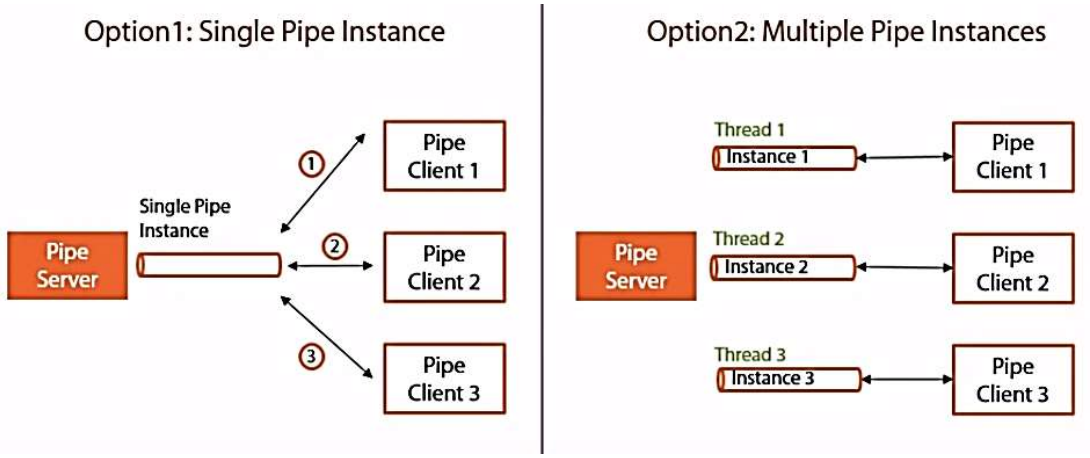
1) Ordinary pipe มีรูปแบบ parent-child กำหนดด้านหนึ่งสำหรับอ่าน และอีกด้านสำหรับเขียน (รูปที่ 3-13) Pipe ประเภทนี้เป็นชนิดการสื่อสารทางเดียวซึ่งจะเห็นจากรูป โดยที่ fd{0} เป็นด้านการอ่าน และ fd(1) เป็นด้านการเขียน หากต้องการสื่อสารสองทิศทางก็ต้องสร้าง Pipe ไว้สองชุด การอ่านและเขียนกระทำผ่านการเรียกใช้ระบบ Read() และ Write()



รูปที่ 3-13: Ordinary Pipe

ที่มา: (Naik, 2014)

2) Named pipe ยืดหยุ่นกว่า ordinary pipe คือเป็นการสื่อสารแบบไม่มีทิศทาง และไม่เป็นรูปแบบ parent-child ดังนั้น หลาย ๆ processes จึงสามารถเข้าร่วมใช้ได้ (รูปที่ 3-14) เมื่อไหร่ที่ named pipe ถูกสร้างขึ้น โปรเซสก็สามารถเข้าใช้ได้ และเมื่อโปรเซสใช้งานช่องทางสื่อสารเสร็จสิ้นก็จะออกจากการใช้งานเพื่อเปิดโอกาสให้โปรเซสอื่น ๆ ใช้



รูปที่ 3-14: Named Pipe

ที่มา : (Named Pipe Server, 2014)

บทสรุป

บทนี้ได้เริ่มต้นด้วยกับการให้ความหมายของโปรเซสเปรียบเสมือนวัตถุที่มีความเคลื่อนไหวอยู่ในระบบคอมพิวเตอร์ ซึ่งมีการใช้งานทรัพยกรของระบบโดยเฉพาะหน่วยความจำ การเคลื่อนไหวของโปรเซสทำให้เกิดสภาวะต่าง ๆ เช่น สถาวะเตรียมพร้อม (ready) สภาวะกำลังประมวลผล (running) และสภาวะรอ (waiting) นอกจากโปรเซสจะถูกสร้างจากการเรียกใช้ของผู้ใช้งาน โปรเซสยังสามารถสร้างโปรเซสอื่นได้ซึ่งกลายเป็นความสัมพันธ์ระหว่างโปรเซสแม่และโปรเซสลูก การเกิดขึ้นใหม่และการยุติการทำงานชองโปรเซสในขณะระบบกำลังทำงานอยู่นั้นมีลักษณะเป็นพลวัต

การทำงานของระบบปฏิบัติการในปัจจุบันรองรับการทำงานหลายโปรแกรมพร้อมกัน ดังนั้นโปรเซสจึงต้องได้รับการจัดการที่มีประสิทธิภาพ สภาพแวดล้อมต่าง ๆ ของโปรเซสจะถูกเก็บรักษาไว้ในชุดควบคุมเรียกว่า Process Control Block (PCB) ซึ่งบันทึกสภาพปัจจุบันของโปรเซสไว้ เมื่อโปรเซสหยุดการประมวลผลชั่วคราวเพื่อปล่อยให้โปรเซสอื่น ๆ ได้รับการประมวลผล สภาพแวดล้อมของโปรเซสในขณะนั้นจะถูกบันทึก

เก็บไว้ และจะถูกนำกลับมาใช้อีกครั้งในการประมวลผลรอบถัดไป การจัดการลักษณะนี้จะทำให้โปรเซสสามารถสลับสับเปลี่ยนการประมวลผลได้

นอกจากโปรเซสจะปฏิบัติภาระกิจของตัวเองแล้ว ในบางครั้งยังมีความต้องการสื่อสารกับโปรเซสอื่นด้วยเช่นกัน การแลกเปลี่ยนข้อมูล การรอรับผลการประมวลผล และการรอให้โปรเซสลูกทำงานจนเสร็จสิ้น สิ่งเหล่านี้เป็นเหตุของการสื่อสารระหว่างโปรเซส รูปแบบของการสื่อสารสามารถแบ่งออกเป็นสองลักษณะคือ รูปแบบใช้พื้นที่ในหน่วยความจำร่วมกัน เมื่อเกิดการอ่านเขียนทุกโปรเซสที่เกี่ยวข้องจะรับทราบทั่วถึงกัน ในอีกด้านเป็นการสื่อสารด้วยการรับส่งข้อความเพื่อให้ได้ข้อมูลที่ต้องการ ซึ่งต้องอาศัยการระบุเจาะจงโปรเซสผู้ส่งและผู้รับในการส่งข้อความในแต่ละครั้ง ทั้งนี้การสื่อสารระหว่างโปรเซสยังกระทำระหว่างคอมพิวเตอร์ด้วย

แบบฝึกหัดท้ายบท

3.1. จงอธิบายความแตกต่างระหว่าง program และ process

3.2. จงบอกส่วนประกอบของ process

3.3. Process มีกี่สภาวะ อะไรบ้าง

3.4. จงอธิบาย Parent process และ Child process มาให้เข้าใจ

3.5. จงอธิบายความสัมพันธ์ระหว่าง Process Control Block (PCB) และ Context Switch

3.6. จงอธิบายการสื่อสารระหว่าง 2 processes ในแบบ shared memory model และ message passing model พร้อมเปรียบเทียบข้อดีข้อเสีย มาให้เข้าใจ

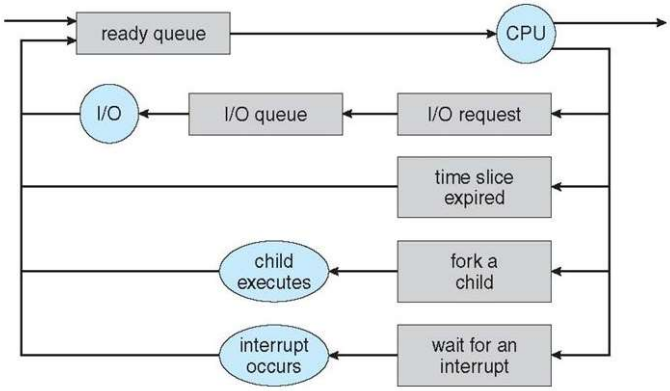
3.7. จงเขียนโปรแกรมภาษา C คำนวนหาค่าพื้นที่สี่เหลี่ยมผืนผ้า พร้อมวาดรูปแสดงหน่วยความจำใน stack โดยที่ :

1) ฟังก์ชัน main() ให้รับค่าความกว้าง และยาว มาเก็บไว้ในตัวแปร a และ b จากผู้ใช้ตามลำดับ

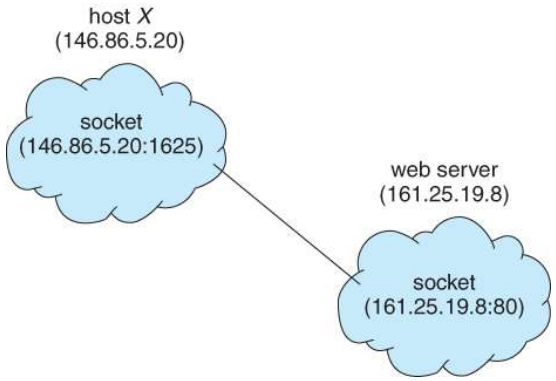
2) เรียกใช้ฟังก์ชัน square() เพื่อคำนวนค่าพื้นที่จะตัวแปรที่รับค่าได้

3) พิมพ์ค่าพื้นที่ออกทางหน้าจอ

3.8. จงอธิบายการสลับการประมวลผลตามไดอะแกรมด้านล่างมาอย่างละเอียด



3.9. จงอธิบายการสื่อสารระหว่าง processes 2 ชุดในลักษณะ client-server ตามไดอะแกรมด้านล่างมาอย่างละเอียด



บรรณานุกรม

Difference Between Long-Term and Short-Term Scheduler in OS. (2016). Retrieved from Techdifferences.com: http://techdifferences.com/difference-between-long-term-and-short-term-scheduler.html

Microsoft. (2003, March 28). *How RPC Works*. Retrieved September 2, 2017, from Microsoft: https://technet.microsoft.com/en-us/library/cc738291(v=ws.10).aspx

Naik, R. (2014). *Pipes in Operating System*. Retrieved September 5, 2017, from Computer Science Geek Zone: https://cdynamicprogramming.blogspot.com/p/pipes-in-operating-system.html

Named Pipe Server. (2014). Retrieved from YouTube Tutorial: https://i.ytimg.com/vi/dLKTbGiogJo/maxresdefault.jpg

Operating System Tutorial. (2017). Retrieved from Tutorial Point: https://www.tutorialspoint.com/operating_system/os_processes.htm

Silberschatz, A., Galvin, P. B., & Gagne, G. (2010). *Operating System Concepts* (8th ed.). Asia: John Wiley & Sons.