บทที่ 13
โปรแกรมระบบปฏิบัติการ

โครงร่างของบทนี้

13.1 วิวัฒนาการของระบบปฏิบัติการ

13.2 องค์ประกอบของโปรแกรมปฏิบัติการ

- 13.2.1 การจ่ายงาน (dispatching) หรือการจัดสรรหน่วยประมวลผล
- 13.2.2 การคิ้วงานและทรัพยากร
- 13.2.3 การจัดการด้าน I/O
- 13.2.3 การจัดการด้านหน่วยความจำ

13.3 ตัวอย่างระบบปฏิบัติการ

- 13.3.1 ระบบปฏิบัติการ DOS
- 13.3.1 ระบบปฏิบัติการ Unix

โปรแกรมระบบปฏิบัติการ (Operating System :OS) คือโปรแกรมที่ช่วยให้ผู้ใช้สามารถติดต่อหรือควบคุมการทำงานของเครื่องคอมพิวเตอร์ให้ง่ายขึ้น และรวมไปถึงการอำนวยความสะดวกในการใช้โปรแกรมต่างๆ และการจัดสรรทรัพยากร (resource) ต่างๆของระบบให้ได้มีประสิทธิภาพมากที่สุด หากจะกล่าวให้เจาะจงมากกว่านี้แล้ว หน้าที่ของระบบปฏิบัติการสามารถแบ่งได้เป็น 2 หน้าที่หลักๆคือ

1. คุมการทำงานของโปรแกรม และอุปกรณ์ต่างๆ ในที่นี้รวมถึงการเอื้ออำนวยให้ผู้ใช้สามารถใช้อุปกรณ์ต่างๆได้สะดวก

2. จัดสรรทรัพยากรต่างๆให้สามารถใช้ร่วมกันได้ ทั้งนี้เนื่องจากอุปกรณ์ต่าง เช่น หน่วยความจำ หน่วยประมวลผลกลาง มีสมรรถนะหรือมีขนาดใหญ่เกิดความจำเป็นของแต่ละงาน จึงได้มีความพยายามที่จะใช้อุปกรณ์ต่างๆเหล่านี้ให้ได้เกิดประโยชน์หรือคุ้มค่าที่สุด โดยให้อุปกรณ์ต่างๆเหล่านี้สามารถใช้งานร่วมกันได้

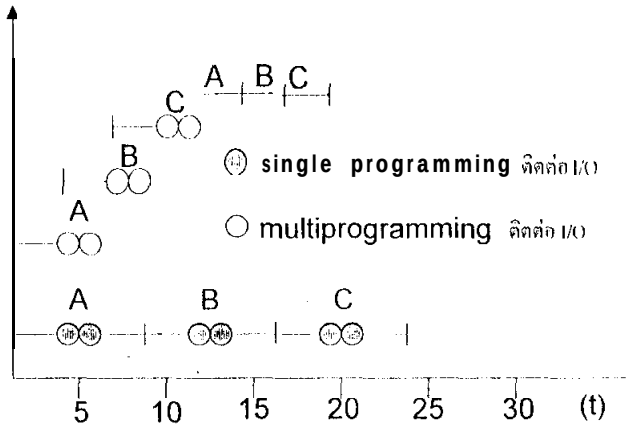
13.1 วิวัฒนาการของระบบปฏิบัติการ

ยุคที่แรกๆของเครื่องคอมพิวเตอร์คือราวปี ค.ศ 1918-1482 คอมพิวเตอร์มีแต่เครื่องมาใส่ๆ ไม่มีระบบปฏิบัติการใดเลย ผู้พัฒนาโปรแกรมต้องเขียนโปรแกรมด้วยภาษาเครื่องทั้งหมด รวมถึงการควบคุม หรือการปฏิบัติงานกับคอมพิวเตอร์ต้องใช้พนักงานเป็นผู้ควบคุม หรือที่เรียกว่า โอเปอร์เรเตอร์ (operater)

การสั่งงานโดยใช้พนักงานควบคุมมีข้อเสียคือ การว่างการทำงานของ CPU เช่น ช่วงเวลาการนำข้อมูลเข้าสู่เครื่อง (ใช้บัตรเจาะรู) หน่วยประมวลกลาง (CPU) ก็จะว่างการทำงานเป็นเวลานานเพราะการทำงานของมนุษย์มีความชักช้ากว่ามากเมื่อเทียบเครื่องจักร จากเหตุผลนี้จึงได้มีการพัฒนาโปรแกรมประมวลแบบกลุ่มโดยอัตโนมัติ (automatic batch) ซึ่งต่อมาได้พัฒนาเป็นโปรแกรมระบบปฏิบัติการ โปรแกรมนี้จะอยู่บนเครื่องตลอดเวลา (resident monitor) โปรแกรมนี้ทำหน้าที่นำข้อมูลจากเครื่องค่านบัตรมาไว้บนหน่วยความจำ หรือนำโปรแกรมมาประมวลผลข้อมูล (load) จากนั้นจะส่งบอกการควบคุมเครื่องคอมพิวเตอร์ให้กับโปรแกรมของผู้ใช้ ซึ่งวิธีนี้ใช้ลดเวลาการทำงานของพนักงานควบคุมเครื่อง

ปัญหาที่สำคัญอีกประการคือ ความแตกต่างระหว่างความเร็วของการทำงานระหว่างหน่วยประมวลกลาง(Central processing unit :CPU) กับอุปกรณ์รับและแสดงผลข้อมูล ซึ่งความแตกต่างนี้ทำให้ประโยชน์ของการใช้ CPU ต่ำลง เพราะว่าขณะที่เครื่องติดต่อกับอุปกรณ์แสดงผล CPU ของเครื่องจะหยุดรอให้อุปกรณ์นี้ทำงานเสร็จสิ้นก่อนจึงเริ่มกลับไปอ่านข้อมูลเพื่อมาประมวลผลต่อข้อมูล

จากปัญหานี้จึงได้มีแนวคิดจึงได้มีแนวคิดในเรื่องของการทำงานแบบเหลื่อมเวลา ซึ่งทำให้ CPU สามารถประมวลผลได้ครั้งหลายโปรแกรมพร้อมๆกัน ซึ่งเรียกว่า ระบบหลายโปรแกรม (multiprogramming) วิธีเหลื่อมเวลานี้จะใช้ช่วงเวลาที่โปรแกรมติดต่อกับหน่วยนำข้อมูล/ออก นำโปรแกรมอีกตัวหนึ่งมาทำงาน รูปที่ 13.1 แสดงการเปลี่ยนเทียบการประมวลผลแบบแต่ (single programming) และแบบระบบหลายโปรแกรม (multiprogramming)



รูปที่ 13.1 แสดงการเปลี่ยนเทียบประมวลแบบโปรแกรมเดียวและแบบระบบหลายโปรแกรม เมื่อรันโปรแกรม A B และ C

การที่ โปรแกรมระบบปฏิบัติการสามารถประมวลผลข้อมูลในลักษณะระบบหลายโปรแกรมได้ ระบบจะต้องมีองค์ประกอบต่างๆเข้าไป องค์ประกอบของโปรแกรมระบบปฏิบัติการ

1. การจ่ายงาน (dispatching)
2. การจัดการเรื่อง Interrupt handling
3. การจัดสรรทรัพยากรของเครื่องคอมพิวเตอร์ ได้แก่ หน่วยความจำ ,I/O
4. การป้องกันการใช้ทรัพยากร
5. การอำนวยความสะดวกให้กับผู้ใช้ในการควบคุมเครื่องคอมพิวเตอร์
6. การจัดการระบบแฟ้มข้อมูล

13.2 องค์ประกอบของโปรแกรมปฏิบัติการ

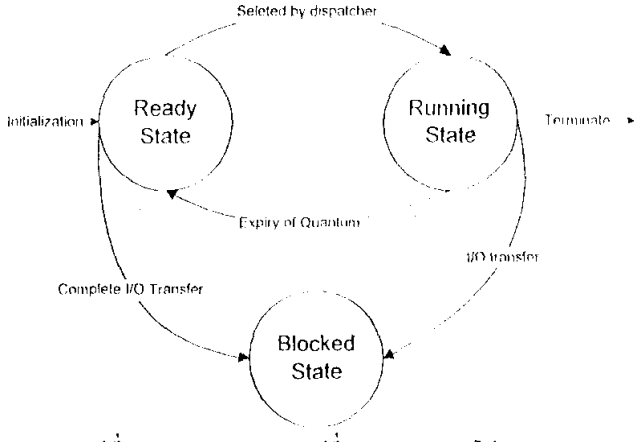
13.2.1 การจ่ายงาน (dispatching) หรือการจัดสรรหน่วยประมวลผล

หน้าที่ประการหนึ่งของ OS คือ การจัดสรรทรัพยากรของระบบที่มีอยู่จำกัดในเครื่องคอมพิวเตอร์ให้กับผู้ต้องการใช้ทรัพยากรเหล่านั้น ซึ่งผู้ที่นำทรัพยากรไปใช้นั้นก็คือ โปรเซส (process) คำว่าโปรเซสถูกนำมาใช้ครั้งแรกโดยผู้ออกแบบระบบปฏิบัติการมัลติกส์ (Multics) โดย

คำว่า "โปรเซส" ยังไม่ได้มีการกำหนดคำนิยามที่แน่นอนลงไป แต่ความหมายว่า *โปรเซส คือ โปรแกรมที่กำลังถูกเอ็กซีคิ้ว* ถูกนำมาใช้บ่อยมาก

การที่จะเอ็กซีคิ้วโปรแกรมหลายๆโปรแกรมโดยที่มีจำนวน CPU เท่ากับจำนวนโปรแกรม หรือมากกว่า ก็จะมีปัญหาในเรื่องของการจัดสรร CPU ให้กับแต่ละโปรเซส แต่ในทางปฏิบัติแล้วเป็นไปไม่ได้ที่เครื่องคอมพิวเตอร์จะมีการเตรียม CPU จำนวนมากพอสำหรับการประมวลผลในระบบหลายโปรแกรม

เนื่องจากในทางปฏิบัติจำนวน CPU มีจำนวนน้อยกว่างานหรือโปรแกรมที่นำมาเอ็กซีคิ้ว ดังนั้นการประมวลในระบบหลายโปรแกรมจึงใช้วิธีการสลับการประมวลผล โดยใช้หลักการของการกำหนดสถานะ (state) ของโปรเซส ซึ่งมี 4 สถานะ



รูปที่ 13.2 แสดงวงจรการเปลี่ยนสถานะของโปรเซส

สถานะพร้อม (Ready State) : สถานะที่โปรเซสพร้อมที่จะใช้ CPU ทันทีที่ OS มอบหมายให้ โดยสถานะนี้ถือเป็นสถานะเริ่มแรกก่อนการประมวลผล

สถานะรัน (Run State) : สถานะที่โปรเซสกำลังครอบครอง CPU ที่สถานะโปรเซสมีการประมวลผลจริงๆบนหน่วยความจำ

สถานะติดขัด (Blocked State) : เป็นสถานะที่โปรเซสหยุดรอเหตุการณ์ใดเหตุการหนึ่ง เช่น การรับข้อมูลจากผู้ใช้ การพิมพ์งานออกสู่เครื่องพิมพ์ ซึ่งโปรเซสไม่จำเป็นต่อครอบครองการทำงานของ CPU

เนื่องจาก CPU ต้องเอ็กซีคิ้วหลายโปรแกรมหรือหลายโปรเซส ดังจึงต้อง "ตัวจ่ายงาน (dispatcher)" ทำหน้าที่ในการสลับการทำงานให้กับแต่ละโปรเซสเมื่อโปรเซสมีการเปลี่ยนสถานะการทำงาน

การเปลี่ยนสถานะของแต่ละโปรเซส เมื่อโปรเซสเริ่มเข้ามาเอ็กซีคิ้วสถานะของโปรเซสอยู่ที่*สถานะพร้อม* จากนั้นเมื่อ CPU ว่างโปรเซสก็จะเปลี่ยนสถานะมาเป็น*สถานะรัน* โดยตัว dispather

ทำหน้าที่สลับการทำงานให้ ที่สถานะนี้ถ้าการประมวลผลของโปรเซสสิ้นสุดลงก็จะจากการทำงานไป (termination) และตัว dispather ก็จะนำโปรเซสที่อยู่ในสถานะพร้อมมาทำการเอ็กซีคิ้ว หรือถ้าโปรเซสต้องการติดต่อกับอุปกรณ์รอบนอก เช่น keyboard เครื่องพิมพ์ สถานะของโปรเซสก็จะเป็น *สถานะติดขัด* โดยที่สถานะติดขัดนี้เมื่อโปรเซสติดต่อกับ I/O เสร็จสิ้นก็จะเป็นสถานะพร้อม ที่สถานะพร้อมถ้ามีหลายโปรเซสอยู่ที่สถานะนี้การเปลี่ยนสถานะไปเป็นสถานะรัน จะใช้วิธีเลือกจากโปรเซสที่มีความสำคัญที่สุด

นอกจากนี้แล้วที่สถานะรันถ้าหากมีโปรเซสใดใช้ CPU นานโดยไม่มีการติดต่อกับ I/O เลย ก็จะทำให้โปรเซสอื่นที่อยู่สถานะพร้อมไม่ได้ใช้ CPU ดังนั้นจึงได้ใช้วิธีกำหนดช่วงเวลาการเอ็กซีคิ้วในสถานะพร้อม โดยที่ถ้าโปรเซสอยู่ในสถานะรันเกินช่วงเวลาที่กำหนด ก็จะถูกเปลี่ยนมาเป็นสถานะพร้อม แล้วนำโปรเซสอื่นที่มีสถานะพร้อมไปทำการเอ็กซีคิ้ว

13.2.2 การจัดลำดับความสำคัญของโปรเซสและทรัพยากร

เนื่องจากมีหลายโปรเซสต้องการใช้ CPU ดังนั้นจึงต้องมีการจัดกิจวรรมเพื่อเลือกโปรเซสที่อยู่สถานะพร้อมมาเปลี่ยนเป็นสถานะรัน โดยตัวจัดกิจวรรมเรียกว่า scheuler ซึ่งทำหน้าที่เลือกว่าโปรเซสใดควรถูกประมวลเมื่อ CPU ว่าง

การพิจารณาการเลือกโปรเซสมาทำการเอ็กซีคิ้ว

1. ปริมาณทรัพยากรที่ใช้ในการประมวลผลของโปรเซส ได้แก่ memory ,ระยะเวลาที่ใช้ในการติดต่อกับ I/O
2. ปริมาณทรัพยากรที่เลือกอยู่ในระบบ
3. ความเร่งด่วนในการทำงานของแต่ละโปรเซส

การจัดสรรทรัพยากรให้กับระบบมี 2 วิธี คือ การจัดสรรแบบสถิต (static) ซึ่งวิธีนี้ทรัพยากรที่โปรเซสต้องการทั้งหมดจะถูกมอบให้กับโปรเซสตั้งแต่เริ่มจนกระทั่งงานสิ้นสุดลง วิธีนี้มีข้อเสียการใช้ทรัพยากรต่างๆอาจไม่คุ้ม เช่น ทรัพยากรบางอย่างอาจถูกมีการใช้เพียงครั้งเดียวแล้วคืนให้กับระบบเลยกับต้องรอให้โปรเซสเสร็จการทำงานก่อน หรืออาจต้องรอให้โปรเซสประมวลผลซักระยะหนึ่งก่อนจึงจะมีการใช้ทรัพยากรที่จัดสรรให้ ซึ่งแทนที่จะนำทรัพยากรที่ยังไม่ได้ประมวลผลหรือประมวลผลเสร็จสิ้นแล้วคืนให้ระบบ เพื่อที่ระบบจะได้จัดสรรทรัพยากรเหล่านี้ให้กับโปรเซสอื่น

แบบที่สองคือแบบจลน์ (dynamic) วิธีนี้จะระบบจัดสรรทรัพยากรให้กับโปรเซสก็ต่อเมื่อโปรเซสมีความต้องการใช้ และคืนเมื่อโปรเซสใช้เสร็จสิ้นแล้ว แต่วิธีนี้อาจเกิดปัญหาที่เรียกว่า dead lock ได้ คือ เมื่อมีโปรเซสตั้งแต่ 2 ตัวขึ้นไปใช้ต้องการใช้ทรัพยากรที่เป็นที่ต้องการของอีกฝ่ายหนึ่ง จึงเกิดปัญหาการรอปล่อยทรัพยากรซึ่งกันและกัน

13.2.3 การจัดการ I/O

สาเหตุที่ OS ที่ต้องมาทำหน้าที่งานเรื่องการติดต่อกับอุปกรณ์รอบนอก ได้แก่

1. I/O ของคอมพิวเตอร์แต่ละยี่ห้อมีการทำงานไม่เหมือนกัน ดังนั้นถ้าหากโปรแกรมประยุกต์ต้องการติดต่อกับอุปกรณ์เหล่านี้ต้องเขียนโปรแกรมควบคุมอุปกรณ์ ซึ่งเป็นการยุ่งยากมาก เพราะวิธีการติดต่อระหว่างอุปกรณ์พวกนี้แต่ละยี่ห้อมีวิธีต่างกัน ด้วยเหตุนี้จึงมอบหน้าที่นี้ให้อยู่ในการควบคุมของ OS เพื่อที่ผู้พัฒนาโปรแกรมต่างๆจะได้ไม่ต้องยุ่งยากเกี่ยวกับการติดต่อกับอุปกรณ์แต่ละยี่ห้อ ดังนั้นโพรเซสหรือโปรแกรมของผู้ใช้จะไม่ทราบถึงขั้นตอนการทำงานของอุปกรณ์ต่างๆ

2 อุปกรณ์ I/O บางตัวสามารถที่ใช้งานร่วมกันได้ เช่น จานแม่เหล็ก ดังนั้นเพื่อให้อุปกรณ์พวกนี้สามารถใช้งานร่วมกันได้หลายโปรแกรม

3. เพื่อต้องการทราบการเกิดขัดจังหวะ (interrupt) เพื่อที่ OS จะได้ทำการเปลี่ยนสถานะของโปรเซสได้ (การติดต่อระหว่าง I/O กับ CPU ใช้วิธีการส่งสัญญาณการขัดจังหวะ เพื่อแจ้งให้ CPU ทราบว่ามีการติดต่อจากอุปกรณ์รอบนอก)

ขั้นตอนการทำงานของ OS เมื่อโปรเซสต้องการติดต่อกับ I/O

1. ตรวจสอบดูว่าคำขอบริการของโปรเซสสามารถจัดสรรอุปกรณ์ที่ต้องการให้กับโปรเซสได้หรือไม่
2. เริ่มทำการติดต่อกับอุปกรณ์ภายนอก
3. ทำการเปลี่ยนสถานะของโปรเซสจาก*สถานะรัน*เป็น*สถานะติดขัด*
4. เรียกตัวจ่ายงาน (dispatcher) เพื่อทำการสลับโพรเซส

ขั้นตอนการทำงานของ OS เมื่อโปรเซสติดต่อกับ I/O เสร็จ

1. ค้นหาว่าโปรเซสใดเป็นผู้ใช้ I/O จากนั้นทำการถ่ายข้อมูลให้กับโพรเซสนั้น
2. เปลี่ยนสถานะของโปรเซสจาก*สถานะติดขัด*เป็น*สถานะพร้อม*
3. กลับไปทำโปรเซสที่เกิดก่อนการขัดจังหวะ

บางครั้งอาจมีโปรเซสจำนวนมากที่มีสถานะติดขัด แล้วรอการใช้อุปกรณ์ที่ไม่สามารถใช้ร่วมกันได้ เช่น เครื่องพิมพ์ จึงทำให้เกิดเทคนิคที่เรียกว่า spooler ซึ่งวิธีนี้จะนำผลลัพธ์ลงสู่จานแม่เหล็กก่อนเพราะเป็นอุปกรณ์ที่ทำงานเร็วกว่าเครื่องพิมพ์ จากนั้นจึงค่อยๆนำข้อมูลจากจานแม่เหล็กมาพิมพ์ที่หลัง ข้อดีของวิธีนี้คือโปรเซสไม่ต้องรอการเปลี่ยนสถานะจากสถานะติดขัดเป็นสถานะพร้อม

13.2.3 การจัดการด้านหน่วยความจำ

หน่วยความจำเป็นทรัพยากรที่สำคัญอย่างของระบบคอมพิวเตอร์ โดย OS ทำหน้าที่ดูแลในเรื่องของการจัดสรรพื้นที่หน่วยความจำให้กับโปรแกรมที่ต้องการใช้ หรือการจัดสรรให้โปรแกรมหลายๆตัวสามารถอยู่บนหน่วยความจำได้พร้อมๆกัน

หน้าที่ของหน่วยจัดสรรหน่วยความจำให้กับระบบ

1. จัดสรรพื้นที่หน่วยความจำให้โปรเซสใช้พื้นที่ในการปฏิบัติงานกับข้อมูลได้อย่างเพียงพอ
2. ป้องกันไม่ให้โปรเซสใช้พื้นที่หน่วยความจำที่ไม่ใช่ของตนเอง
3. จัดสรรพื้นที่หน่วยความจำให้มีพื้นที่ว่างมากพอสำหรับการใช้งานได้ตลอดเวลา

วิธีการจัดสรรพื้นที่หน่วยความจำของ OS มี 2 ระบบ คือ ระบบโปรแกรมเดี่ยว (single program) และระบบหลายโปรแกรม (multiprograming) ซึ่งวิธีในการจัดสรรพื้นที่หน่วยความจำให้กับทั้งสองระบบมีหลายวิธีด้วยกัน แต่เนื้อหาในวิชานี้จะยกแต่หลักการที่ไม่สลับซับซ้อนเป็นตัวอย่างเพื่อให้ผู้อ่านเกิดภาพของการบริหารหน่วยความจำของ OS

ระบบโปรแกรมเดี่ยว (Sigle programming)

ระบบนี้แบ่งพื้นที่หน่วยความจำออกเป็น 2 ส่วน คือ ส่วนของ OS และส่วนของผู้ใช้ (user area) ซึ่งส่วนนี้เป็นส่วนที่ผู้ใช้สามารถนำโปรแกรมเข้ามาทำการประมวลผลได้ เนื่องจากเพื่อที่โปรแกรมถูกแบ่งออกเป็น 2 ส่วน ตัว OS จึงจำเป็นต้องมีการป้องกันเพื่อที่หน่วยความจำไม่ให้ผู้ใช้เข้ามารุกล้ำส่วนของ OS วิธีหนึ่งที่นิยมกันก็คือการสร้างรีจีสเตอร์ขึ้นมาตัวตัวหนึ่งซึ่งเรียกว่า รีจีสเตอร์ขอบเขต (boundary register) ซึ่งรีจีสเตอร์นี้จะเก็บค่าขอบเขตที่เป็นรอยต่อระหว่างส่วนของ OS และส่วนของผู้ใช้ ซึ่งเมื่อมีการประมวลคำสั่ง CPU ก็จะทำการตรวจสอบว่ามีการรุกล้ำผ่านค่าที่เก็บไว้ในรีจีสเตอร์ขอบเขตหรือไม่

ระบบโปรแกรมหลายโปรแกรม (Multiprogramming)

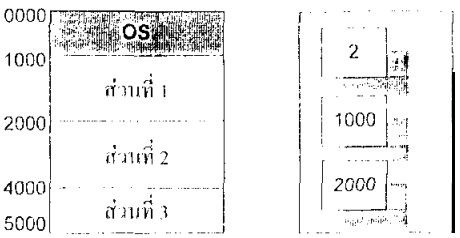
เนื่องจากระบบนี้โปรแกรมหลายๆตัวต้องสามารถทำงานได้พร้อมกัน ดังนั้นในการจัดสรรหน่วยความจำจึงใช้วิธีการแบ่ง (partition) หน่วยความจำออกเป็นส่วนๆ เพื่อไม่ให้โปรแกรมแต่ละตัวใช้หน่วยความจำปะปนกัน ซึ่งการทำงานในลักษณะนี้ OS นอกจากจะต้องป้องกันตนเองจากโปรแกรมตัวอื่นแล้ว ยังต้องป้องกันโปรแกรมที่กำลังประมวลอยู่ไม่ให้มีการอ้างถึงหน่วยความจำของกันและกัน

ในที่นี้ขอยกตัวอย่างการแบ่งหน่วยความจำชนิดคงที่ (fixed partition multiprogramming) ซึ่งวิธีนี้หน่วยความจำจะถูกแบ่งออกเป็นส่วนๆโดยที่แต่ละส่วนมีขนาดคงที่

รูปที่ 13.3 แสดงตัวอย่างการแบ่งหน่วยความจำออกเป็นส่วน

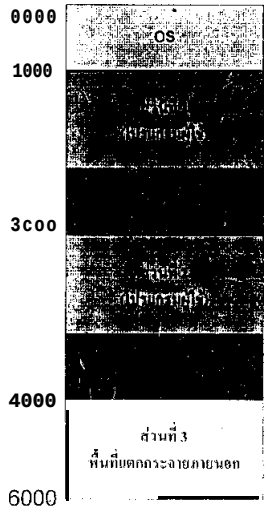
รูปที่ 13.3 แสดงตัวอย่างการแบ่งหน่วยความจำออกเป็นส่วนๆ ซึ่งจากการแบ่งในลักษณะนี้จำนวนโปรแกรมที่สามารถรันได้พร้อมกันมากที่สุดคือ 3 โปรแกรม

ซึ่งการป้องกัน OS การอ้างหน่วยความจำผิดในระบบหลายโปรแกรมจะใช้รีจิสเตอร์ขอบเขต 2 ตัวในการแยกขอบเขต คือ รีจิสเตอร์ขอบเขตบน/ล่าง (high/low boundary register) และมีรีจิสเตอร์อีกตัวเพื่อบอกว่า OS กำลังนำหน่วยความจำส่วนไหนมาทำงาน รูปที่ 13.4 แสดงตัวอย่างการป้องกันของ OS ในระบบหลายโปรแกรม



รูปที่ 13.4 แสดงตัวอย่างการป้องกันของ OS ในระบบหลายโปรแกรม

ปัญหาในการแบ่งพื้นที่หน่วยความจำก็คือ ปัญหาในเรื่องของ*การเกิดการแตกกระจาย (framentation)* ซึ่งการเกิดการแตกกระจายมี 2 ลักษณะคือ การแตกกระจายภายใน (internal framentation) ซึ่งปัญหานี้เกิดจากการที่โปรแกรมของผู้ใช้ใช้พื้นที่จัดสรรให้ไม่หมด และการแตกกระจายภายนอก (external framentation) คือพื้นที่ว่างที่ไม่ได้ใช้ แสดงรูปที่ 13.5 ตัวอย่างการเกิดการแตกกระจาย ซึ่งจากพื้นที่ว่างที่เหลือนี้หากเรานำโปรแกรมที่มีขนาดโดยรวมเท่ากับพื้นที่เหลือมารัน (run) ก็ไม่สามารถเอ็กซีคิ้วได้เนื่องจากพื้นที่หน่วยความจำไม่เพียงพอ



รูปที่ 13.5 แสดงการเกิดปัญหาแตกกระจาย

13.2.3 การจัดการระบบแฟ้มข้อมูล (file system)

โดยทั่วไป "แฟ้ม (file)" มีความหมายคือ กลุ่มของสารสนเทศที่มีความสัมพันธ์กัน ความสัมพันธ์นี้ถูกกำหนดโดยผู้สร้าง ซึ่งอาจใช้เก็บอะไรก็ได้ ไม่ว่าจะเป็นข้อมูลหรือโปรแกรม โดยทั่วไปเราได้แบ่งไฟล์ออกเป็นชนิดต่างโดยใช้นามสกุลหรือส่วนขยาย (file extension) เป็นตัวแยกชนิดต่างๆ เช่น

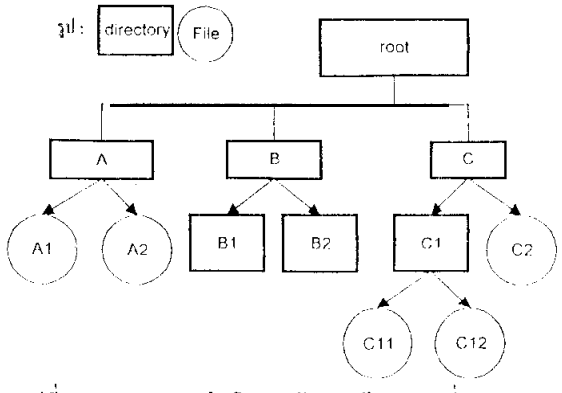
FILENAME.C : แฟ้มโปรแกรมภาษา C

FILENAME.DAT : แฟ้มข้อมูลทั่วไป

FILENAME.EXE : แฟ้มโปรแกรมที่สามารถเอ็กซีคิ้วได้

ใน OS บางตัวส่วนขยายส่วนขยายไม่มีความหมายกับตัวระบบ OS แต่เมื่อเพื่อให้ผู้ใช้สามารถแบบกลุ่มข้อมูลได้ง่ายขึ้น เช่น ระบบ Unix จะไม่สนใจนามสกุลของแฟ้มข้อมูล ส่วน DOS แฟ้มข้อมูลที่นามสกุล EXE หรือ COM สามารถที่จะเอ็กซีคิ้วได้

ไดเร็กทอรี่ (direcotory) คือ สารบัญหรือที่รวาเรวมรายชื่อแฟ้มข้อมลในระบบ และข้อมูลบางส่วนที่สำคัญของระบบ เช่น ชื่อ ส่วนขยาย เจ้าของแฟ้มข้อมูล เป็นต้น ลักษณะของไดเร็กทอรี่มีลักษณะเหมือนกับต้นหมายหัวกลับ ซึงข้อมูลในหนึ่งไดเร็กทอรี่อาจประกอบด้วยแฟ้มข้อมูลหรือไดเร็กทอรี่ย่อย รูปที่ 13.6 แสดงตัวอย่างขอระบบไดเร็กทอรี่ ในระบบไดเร็กทอรี่ส่วนที่อยู่บนสุดเรียกว่า root



รูปที่ 13.6 แสดงตัวอย่างโครงสร้างของไดเรอทอรี่

ระบบแฟ้ม (file sytem) คือ ส่วนที่ทำหน้าที่ดูแลเกี่ยวกับงานที่เกี่ยวข้องกับแฟ้ม ซึ่งหน้าที่ของระบบแฟ้ม เราสามารถจำแนกได้เป็น

1. สร้างและลบ

2. การเรียกใช้ข้อมูล

3. จัดสรรเนื้อที่ของหน่วยความจำสำรองเพื่อเก็บข้อมูล

4. ในระบบ multiuser มีส่วนป้องกันไม่ให้ผู้มีสิทธิในการเข้าถึงข้อมูล

5. ป้องกันการเสียหายจากการจัดเก็บแฟ้มข้อมูล

เนื่องจากในการจัดเก็บข้อมูลต้องมีการสำรองข้อมูล (Back up) เพื่อป้องกันการเสียหายจากข้อมูล ซึ่งมี 2 วิธี

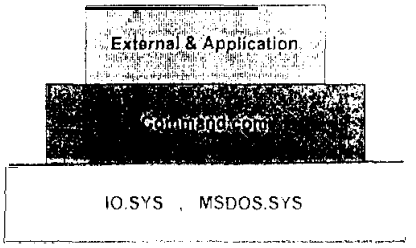
1. periodically backup : วิธีทำการสำรองข้อมูลเป็นระยะๆ โดยอาจใช้วิธีกำหนดไว้ว่า ทุกๆ 1 สัปดาห์ทำการสำรองข้อมูลหนึ่งครั้ง โดยการสำรองลักษณะนี้จะทำการสำรองข้อมูลทั้งหมด

2. incrementally backup : วิธีนี้การสำรองข้อมูลจะมีแต่เฉพาะข้อมูลที่ถูกแก้ไข และถูกสร้างใหม่เท่านั้น ซึ่งวิธีนี้มีข้อดีคือ เร็วเพราะในกรณีที่มีข้อมูลที่ถูกจัดเก็บปริมาณมากๆถ้าหากใช้วิธีสำรองข้อมูลทั้งหมดก็จะเสียเวลานาน แต่ถ้าหากเลือกสำรองเฉพาะข้อมูลที่ถูกแก้ไข หรือถูกสร้างใหม่ก็จะทำได้รวดเร็วเพราะมีปริมาณข้อมูลที่น้อยกว่า

13.3 ตัวอย่างระบบปฏิบัติการ

13.3.1 ระบบปฏิบัติการ DOS

ระบบปฏิบัติการนี้ถูกพัฒนาจากบริษัท Mircrosoft เป็นระบบแบบผู้ใช้คนเดียว (Single User) รูปที่ 13.7 แสดงโครงสร้างของ DOS ซึ่งประกอบด้วย 3 ส่วน คือ



รูปที่ 13.7 แสดงตัวอย่างโครงสร้างของ DOS

1. *IO.SYS และ MS-DOS.SYS* : ซึ่งในส่วนนี้ทำหน้าที่ควบคุมการทำงานอุปกรณ์ต่างของระบบโดยตรง

2. *Command.com* : เป็นแฟ้มคำสั่งธรรมดาซึ่งเราสามารถดูชื่อแฟ้มข้อมูลได้โดยใช้คำสั่งดูรายชื่อข้อมูล โดยหน้าที่ของส่วนนี้ทำหน้าที่จัดการเรื่องโปรเซสต่างๆในระบบ และเป็นตัวแปรคำสั่งที่ใช้ติดต่อกับผู้ใช้ โดยในส่วนนี้มีคำสั่งที่เรียกว่า Internal Command อยู่ (คำสั่งเป็นนี้จัดเก็บในรูปของแฟ้มข้อมูล)

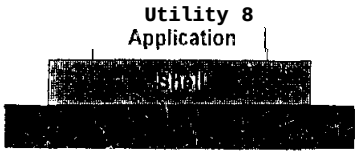
3. *External command & Application* : External Command คือ คำสั่งภายนอกที่จะถูกเก็บอยู่รูปแฟ้มข้อมูล เมื่อเราใช้คำสั่งดูรายชื่อแฟ้มข้อมูล (dir) จะเห็นรายชื่อแฟ้มข้อมูลของคำสั่ง ภายนอกที่ต้องนำมาไว้ภายนอกเพราะขนาดของคำสั่งที่ใช้ในการจัดเก็บมีขนาดใหญ่ ส่วน Application ได้แก่ โปรแกรมทั่วไป เมื่อโปรแกรมในระดับชั้นต้องการติดต่อกับเครื่องคอมพิวเตอร์จะติดต่อผ่านทาง command.com

การจัดเก็บข้อมูลของ DOS เป็นมีลักษณะเป็นไดเรกทอรี โดยที่ระบบแฟ้มของ DOS เป็นแบบหลายไดเรกทอรีระบบคล้ายคลึ หนึ่งไดร์ฟก็คือหนึ่งระบบแฟ้มข้อมูล รูปที่ 13.8 แสดงตัวอย่างระบบแฟ้มของ DOS

รูปที่ 13.8 ระบบแฟ้มของไดรฟ์ A และ C ของระบบปฏิบัติการ DOS

13.3.1 ระบบปฏิบัติการ Unix

ระบบปฏิบัติการนี้ถูกพัฒนาจากห้องปฏิบัติการเบลล์แลป (bell lab) ของบริษัท AT&T โดย Ken Tompson และ Dennis Ritchie เป็นระบบแบบผู้ใช้หลายคน (Multi User) รูปที่ 13.8 แสดงโครงสร้างของ Unix ซึ่งประกอบด้วย 3 ส่วน คือ



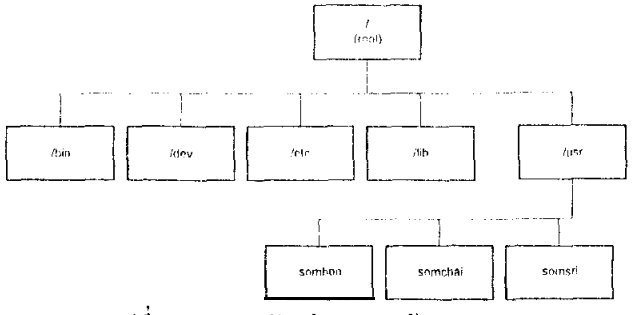
รูปที่ 13.9 แสดงโครงสร้างของระบบ Unix

1. Kernal : ถือว่าเป็นส่วนหัวใจของระบบเพราะเป็นส่วนที่ทำหน้าที่ควบคุมการทำงานของฮาร์ดแวร์ เช่น ควบคุมการสลับโปรเซสไปมาระหว่างงานหลายๆงาน และควบคุมการทำงานของจานแม่เหล็ก เป็นต้น

2. Shell : ทำหน้าที่เป็นสื่อกลางระหว่างผู้ใช้และระบบ ซึ่งถ้าเทียบกับ DOS ก็เป็นเสมือน Command.com โดยหน้าที่หลักของส่วนนี้ คือ เป็นล่ามแปลคำสั่งต่างๆที่ผู้ใช้ต้องการเพื่อให้ Kernal เข้าใจ ซึ่งตัว Shell ของ Unix นี้สามารถที่จะเปลี่ยนเป็นอะไรก็ได้ ไม่เหมือนกับ DOS

3. Utility & Application : ซึ่งส่วนนี้ถ้าเทียบกับ DOS แล้วก็คือ External command ซึ่งจะถูกนำมาเอ็กซีคิ้วก็ต่อเมื่อมีการเรียกใช้

ระบบแฟ้มของ Unix เป็นแบบระบบไดเร็กทอรี่เดียวแต่มีหลายชั้น โดยส่วนที่อยู่บนสุดเรียกว่า root รูปที่ 13.10 โดยที่ผู้ใช้แต่ละคนก็จะมี directory เป็นของตนเอง เช่น สมชายเมื่อเข้าสู่ระบบก็จะมาไดเร็กทอรี่ somchai



รูปที่ 13.10 แสดงตัวอย่างระบบแฟ้มของ Unix