บทที่ 1 ภาพรวมของคอมพิวเตอร์ และระบบปฏิบัติการ

วัตถุประสงค์ของเนื้อหา

- ทบทวนโครงสร้างทางฮาร์ดแวร์ของระบบคอมพิวเตอร์ที่มีใช้งานโดยทั่วไปในปัจจุบัน
- อธิบายโครงสร้างพื้นฐานและหน้าที่ของระบบปฏิบัติการ
- อธิบายถึงลักษณะความแตกต่างของโครงสร้างระบบปฏิบัติการที่ถูกนำไปใช้กับฮาร์ดแวร์ที่ออกแบบสำหรับงานเฉพาะด้านที่แตกต่างกัน
- อธิบายถึงพื้นฐานของลักษณะการทำงานประสานกันระหว่างซอฟต์แวร์ของระบบปฏิบัติการ และโปรแกรมผู้ใช้

สิ่งที่คาดหวังจากการเรียนในบทนี้

- นักศึกษาเข้าใจถึงโครงสร้างทางฮาร์ดแวร์ของคอมพิวเตอร์ ที่มีใช้ในปัจจุบัน ซึ่งรวมถึงคอมพิวเตอร์ส่วนบุคคล ไปจนถึงสมาร์ทโฟน และคอมพิวเตอร์ฝังตัวในอุปกรณ์ต่างๆ
- นักศึกษาเข้าใจถึงหลักการพื้นฐานของการทำงานของระบบปฏิบัติการ และหน้าที่ของระบบปฏิบัติการที่มีต่อโปรแกรมผู้ใช้

วัตถุประสงค์ของปฏิบัติการท้ายบท

- นักศึกษาได้ทดลองใช้งานระบบปฏิบัติการที่มีใช้งานในปัจจุบันอย่างน้อยสองระบบปฏิบัติการ
- นักศึกษาได้ทดลองพัฒนาโปรแกรมภาษาซี/ซีพลัสพลัส สำหรับระบบปฏิบัติการทีมีความแตกต่างกัน
- นักศึกษาได้เห็นตัวอย่างถึงหลักการพัฒนาโปรแกรมขั้นพื้นฐาน ที่ใช้ร่วมกันระหว่างระบบปฏิบัติที่แตกต่างกัน และหลักการที่อาจมีความแตกต่างกันระหว่างระบบปฏิบัติการ

สิ่งที่คาดหวังจากปฏิบัติการท้ายบท

- นักศึกษาสามารถพัฒนาโปรแกรมภาษาซี/ซีพลัสพลัส เพื่อให้ทำงานบนระบบปฏิบัติการไมโครซอฟต์วินโดวส์ ด้วยชุดพัฒนา Visual Studio หรือที่ทดแทนกันได้ ในระดับเบื้องต้น (Window Console)
- นักศึกษาสามารถใช้งาน Virtual Machine อาทิเช่น VisualBox หรือ VMWare และใช้งานลินุกซ์อิมเมจได้
- นักศึกษาสามารถใช้งาน Linux ได้ในระดับเบื้องต้น และสามารถพัฒนาโปรแกรมภาษาซี/ซีพลัสพลัส เพื่อให้ทำงานบนลินุกซ์ในระดับเบื้องต้น (text terminal input/output)

เวลาที่ใช้ในการเรียนการสอน

- ทฤษฎี 2 ชั่วโมง
    - โครงสร้างทางฮาร์ดแวร์โดยทั่วไป (1.1-1.2) 1 ชั่วโมง
    - โครงสร้างระบบปฏิบัติการโดยทั่วไป (1.3-1.4) 1 ชั่วโมง
- ปฏิบัติ 2 ชั่วโมง
    - ทบทวนการเขียนโปรแกรมบนระบบปฏิบัติการวินโดวส์ 1 ชั่วโมง
    - ใช้งานลินุกซ์เบื้องต้น และเขียนโปรแกรมบนระบบปฏิบัติการลินุกซ์ 1 ชั่วโมง

บทที่ 1 ภาพรวมของคอมพิวเตอร์ และระบบปฏิบัติการ

1.1 โครงสร้างทั่วไปของระบบคอมพิวเตอร์

สภาพแวดล้อมของระบบคอมพิวเตอร์สามารถมองเป็นระดับชั้นดังนี้

- **ระดับฮาร์ดแวร์** ซึ่งมีองค์ประกอบย่อยที่สำคัญคือ หน่วยประมวลผลกลาง (Central Processing Unit - CPU) หน่วยความจำ (Memory) และหน่วยรับส่งข้อมูลเข้าออก (Input/output (I/O) Devices)
- **ระดับระบบปฏิบัติการ** ทำหน้าที่ควบคุมการทำงานของฮาร์ดแวร์ และจัดสรรทรัพยากรต่างๆ เพื่อให้โปรแกรมสามารถทำงานได้โดยสะดวก เราอาจสรุปหน้าที่การทำงานของระบบปฏิบัติการได้อย่างสั้นๆ ในสองนัยยะ นัยยะแรกคือความหมายของการเป็น**ผู้จัดสรรทรัพยากร (Resource allocator)** ที่คอยจัดสรรทรัพยากรให้กับโปรแกรมที่ทำงาน และนัยยะที่สองคือความหมายของการเป็น**โปรแกรมควบคุม (Control Program)** เพื่อจัดการการสั่งการ (execute) ให้โปรแกรมทำงานได้และป้องกันไม่ให้ความผิดพลาดหรือการทำงานที่ไม่เหมาะสมของโปรแกรม ส่งผลโดยรวมต่อระบบและระบบปฏิบัติการยังมีหน้าที่ควบคุมดูแลการทำงานของหน่วยรับส่งข้อมูลต่างๆ ด้วย
- **ระดับโปรแกรม** มีองค์ประกอบคือโปรแกรมต่างๆ ที่ทำงานอยู่บนระบบ โดยอาจแบ่งออกเป็นสองกลุ่มคือ **โปรแกรมระบบ (System programs)** ที่ทำหน้าที่จัดการการทำงานของคอมพิวเตอร์ในหน้าที่ต่างๆ ที่ได้รับมอบหมายหรือเพิ่มเติมจากระบบปฏิบัติการ และ**โปรแกรมผู้ใช้ (Application programs)** เป็นโปรแกรมที่ทำงานติดต่อโดยตรงกับผู้ใช้ในงานต่างๆ ที่ผู้ใช้เป็นผู้สั่งการ เริ่มการทำงาน และจบการทำงานจากคำสั่งของผู้ใช้เองโดยตรง
- **ระดับผู้ใช้** คือผู้ใช้งานที่เข้าถึงระบบคอมพิวเตอร์เพื่อการสั่งงานอย่างใดอย่างหนึ่ง

ระดับความแตกต่างทางโครงสร้างของฮาร์ดแวร์ของคอมพิวเตอร์

คอมพิวเตอร์ที่มีใช้งานในปัจจุบันมีลักษณะที่หลากหลายและถูกนำไปใช้ควบคุมอุปกรณ์ไฟฟ้าและเครื่องมือเครื่องจักรกลต่างๆ ในปัจจุบัน ตัวอย่างของความหลากหลายที่เราพบเห็นได้ดังเช่น

- (จากอดีต) คอมพิวเตอร์ในช่วงเริ่มแรก ต้องใช้พื้นที่ขนาดใหญ่ และกินกำลังไฟสูงมาก จึงต้องสร้างขึ้นในลักษณะติดตั้งประจำที่ และใช้พื้นที่ขนาดใหญ่ การเชื่อมต่อใช้งานจะผ่าน terminal คอมพิวเตอร์เหล่านี้ถูกเรียกว่าเมนเฟรม (mainframe) ต่อมาเมื่อมีการพัฒนาอุปกรณ์อิเล็กทรอนิคส์ที่กินไฟน้อยลง (ทรานซิสเตอร์และไอซี) เราจึงสามารถสร้างคอมพิวเตอร์ที่มีขนาดเล็กลงได้ คอมพิวเตอร์ที่เล็กลงมาจึงถูกเรียกว่า มินิคอมพิวเตอร์ (minicomputer) และต่อมามีการออกแบบวงจรคำนวณไว้บนชิพเดียวทำให้สามารถสร้างคอมพิวเตอร์ที่มีขนาดเล็กลง ราคาถูก สามารถนำมาใช้งานส่วนบุคคลได้ไม่ต้องแชร์ร่วมกัน จึงถูกเรียกว่าเป็นคอมพิวเตอร์ตั้งโต๊ะ (desktop computer) หรือคอมพิวเตอร์ส่วนบุคคล (personal computer -PC)
- คอมพิวเตอร์ส่วนบุคคลที่ถูกออกแบบสร้างขึ้นเพื่อให้มีขีดความสามารถสูง และเหมาะสมกับงานเฉพาะด้าน (หรืองานที่ต้องใช้พลังการคำนวณ หรือการแสดงผล สูงกว่าปกติ) เกิดเป็นเวิร์คสเตชัน (workstation)
- คอมพิวเตอร์ที่ถูกออกแบบมาให้สามารถเป็นผู้ให้บริการจากผู้ใช้บริการจำนวนมากเรียกว่า เซอร์เวอร์ หรือเครื่องแม่ข่าย (server)
- ในปัจจุบัน มีการนำเอาเทคโนโลยีและอุปกรณ์ที่ใช้ในเครื่องคอมพิวเตอร์ส่วนบุคคล เครื่องเวิร์คสเตชัน และเครื่องเซอร์เวอร์ ประกอบกันเข้าเป็นเครื่องที่มีพลังการคำนวณสูงมาก (โดยมักจะมีอุปกรณ์ภายในเช่น ซีพียู การ์ดช่วยประมวลผลต่างๆ จำนวนมาก เชื่อมต่อกันผ่านเครือข่ายภายในความเร็วสูง) เรามักเรียกเครื่องคอมพิวเตอร์ในลักษณะนี้ว่า ซูปเปอร์คอมพิวเตอร์ (supercomputer)

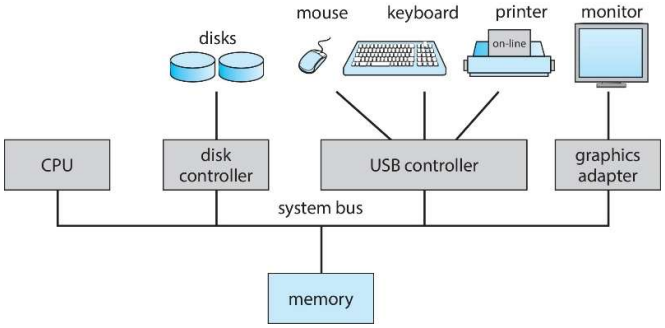
- ในอีกด้านหนึ่งของการคำนวณ มีการออกแบบเครื่องคอมพิวเตอร์ให้มีขนาดเล็กกะทัดรัด เหมาะแก่การพกพา เกิดเป็น PDA (Personal Data Assistant) นำไปใช้งานเฉพาะบุคคล เช่นการบันทึกการนัดหมาย การรับส่งอีเมล์ และอื่นๆ และต่อมาได้ควบรวมเครื่องคอมพิวเตอร์ในลักษณะนี้เข้ากับโทรศัพท์ เกิดเป็นสมาร์ทโฟน (smartphone) โดยสมาร์ทโฟนในปัจจุบันที่นิยมใช้กันมักจะใช้ระบบทัชสกรีนเป็นหลัก
- ยังมีการออกแบบคอมพิวเตอร์ขนาดเล็กฝังตัว (embedded computer) เพื่อนำไปใช้ควบคุมอุปกรณ์ไฟฟ้าต่างๆ เช่น เครื่องปรับอากาศ ไมโครเวฟ เครื่องซักผ้า ไปจนถึงรถยนต์ และอุปกรณ์เครื่องจักรกลอื่นๆ อีกมาก สังเกตว่าคอมพิวเตอร์ในลักษณะนี้ ผู้ใช้จะไม่ได้ติดต่อใช้งานกับคอมพิวเตอร์ในลักษณะที่เห็นเป็นคีย์บอร์ดหรือเมาส์ตามปกติ แต่สั่งการผ่านตัวเครื่องใช้ไฟฟ้าผ่านปุ่ม หรืออุปกรณ์ต่างๆ ที่ผู้ใช้งานคุ้นชินกับระบบเดิม
- ในปัจจุบันมีการออกแบบระบบคอมพิวเตอร์ฝังตัวให้เข้าไปอยู่ในอุปกรณ์ต่างๆ มากยิ่งขึ้น (เช่นหลอดไฟ ปลั๊กไฟ ประตู หน้าต่าง และอื่นๆ) และออกแบบระบบเครือข่ายเพื่อให้สามารถควบคุมจากส่วนกลางได้โดยสะดวก เกิดเป็น IoT (Internet of Things)

ผู้ใช้คาดหวังอะไรจากระบบปฏิบัติการ

- ความง่ายต่อการใช้งาน โดยไม่ต้องพะวงถึงว่าคอมพิวเตอร์จะต้องจัดสรรทรัพยากรให้กับโปรแกรมของตนอย่างไร
- ในสภาพของระบบที่มีหลายผู้ใช้ เช่นคอมพิวเตอร์เมนเฟรม หรือมินิคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการยูนิกซ์/ลินุกซ์ ที่ให้ผู้ใช้เชื่อมต่อเข้าผ่านทางเทอร์มินัลในปัจจุบัน ผู้ใช้แต่ละคนจะต้องได้รับการจัดสรรทรัพยากรอย่างเพียงพอ โดยหน้าที่ของระบบปฏิบัติการในประเด็นนี้คือ การจัดสรรเวลาการใช้ทรัพยากรที่ผู้ใช้คนหนึ่งมิได้ใช้งาน เพื่อไปมอบให้กับผู้ใช้คนอื่น ดังนั้นสภาพของระบบปฏิบัติการที่ควรจะเป็นก็คือ ผู้ใช้ทุกคนต้องรู้สึกพึงพอใจต่อเวลาการใช้ทรัพยากรต่างๆ ที่ตนได้รับ
- ในสภาพของระบบคอมพิวเตอร์ที่จัดสรรทรัพยากรโดยเฉพาะเจาะจงบางส่วนให้กับผู้ใช้ โดยผู้ใช้ใช้งานบนเวิร์คสเตชันและมีทรัพยากรอีกส่วนจัดสรรผ่านเซอร์เวอร์ นั้น ระบบปฏิบัติการบนเซอร์เวอร์ จะต้องทำหน้าที่คอยดูแลจัดสรรทรัพยากรให้เพียงพอต่อผู้ใช้คนหนึ่งๆ เท่าที่จำเป็น แต่ในขณะเดียวกันก็ต้องใช้ทรัพยากรที่มีอยู่ให้เกิดประโยชน์สูงสุด
- ปัจจุบันเริ่มมีคอมพิวเตอร์ที่มีขนาดเล็กที่พกพาได้สะดวก เช่นสมาร์ทโฟน หน้าที่ของระบบปฏิบัติการในคอมพิวเตอร์ประเภทนี้จึงต้องสามารถรีดประสิทธิภาพการใช้งานจากฮาร์ดแวร์ที่มีอยู่ออกมาอย่างจำกัด ให้ได้สูงที่สุด และการออกแบบจะต้องสามารถทำให้รองรับการใช้งานที่หลากหลายของผู้ใช้แต่ละคนให้ได้ (มีความเป็นมิตรต่อผู้ใช้แต่ละคน)
- คอมพิวเตอร์อีกแบบหนึ่ง ทำงานโดยรับส่งข้อมูลโดยตรงกับผู้ใช้น้อยที่สุดหรือไม่มีเลย เราเรียกคอมพิวเตอร์แบบนี้ว่าคอมพิวเตอร์ฝังตัว (Embedded computer) ระบบปฏิบัติการหากมีอยู่ในระบบเหล่านี้ จะถูกออกแบบมาเพื่อการจัดการฮาร์ดแวร์ และตัวโปรแกรมที่ทำงานเป็นหลัก

1.2 โครงสร้างทางฮาร์ดแวร์และระบบ

คอมพิวเตอร์ที่เราใช้งานในปัจจุบันโดยส่วนใหญ่ จะมีองค์ประกอบทางฮาร์ดแวร์ดังต่อไปนี้



- มี CPU ทำหน้าที่ประมวลผล ซึ่งอาจจะมีหนึ่งหน่วย หรือมากกว่าหนึ่งหน่วย
- มีบัสระบบที่ใช้เชื่อมระหว่างซีพียูกับองค์ประกอบต่างๆ โดยองค์ประกอบแต่ละตัวอาจจะใช้บัสที่เชื่อมต่อกับซีพียูต่างมาตรฐานกันไป
    - บัสระบบที่ใช้จัดการปัจจุบันที่นิยมกัน PCIe
- มีอุปกรณ์ควบคุมฮาร์ดแวร์ (device controllers) ที่เชื่อมต่อระหว่างฮาร์ดแวร์ที่ทำหน้าที่รับส่งข้อมูลต่างๆ กับดาต้าบัสของระบบ (โดยใช้ดาต้าบัสร่วมกับซีพียู)
- การทำงานของซีพียู และดีไวซ์ต่างๆ จะทำงานไปพร้อมๆ กัน โดยจะมีหน่วยความจำสำรอง (buffer) ส่วนหนึ่งอยู่ใน hardware controller ที่ซีพียูและดีไวซ์ผลัดกันเข้าใช้งานเพื่อรับส่งข้อมูลระหว่างกัน และมีซอฟต์แวร์ที่เรียกว่า device driver ทำหน้าที่เป็นสื่อกลางระหว่างดีไวซ์ต่างๆ กับองค์ประกอบอื่นๆของระบบปฏิบัติการ
- การส่งสัญญาณจากทางดีไวซ์เพื่อให้ซีพียูรับทราบว่าได้ส่งข้อมูลมาเตรียมรอไว้แล้วให้จัดการต่อได้ อาศัยการอินเทอร์รัพต์

ระบบปฏิบัติการ อาจจะแบ่งส่วนของซอฟต์แวร์ออกเป็นสองส่วนโดยคร่าวๆ ส่วนหนึ่งเป็นส่วนซอฟต์แวร์ที่จะต้องฝังตัวทำงานอยู่ในคอมพิวเตอร์ตลอดเวลา และเป็นแกนของการทำงานของระบบโดยรวม เราเรียกส่วนนี้ว่าเคอร์เนิล (Kernel) กับอีกส่วนหนึ่งที่เป็นโค้ดที่เสริมเข้ามาเพื่อทำหน้าที่ตอบสนองต่อสภาพแวดล้อมที่แตกต่างกัน เราเรียกว่า System Programs (ซึ่งในที่นี้เราอาจรวมไปถึงพวก device drivers และ module ต่างๆ)

<u>การเริ่มต้นทำงานของระบบคอมพิวเตอร์</u>

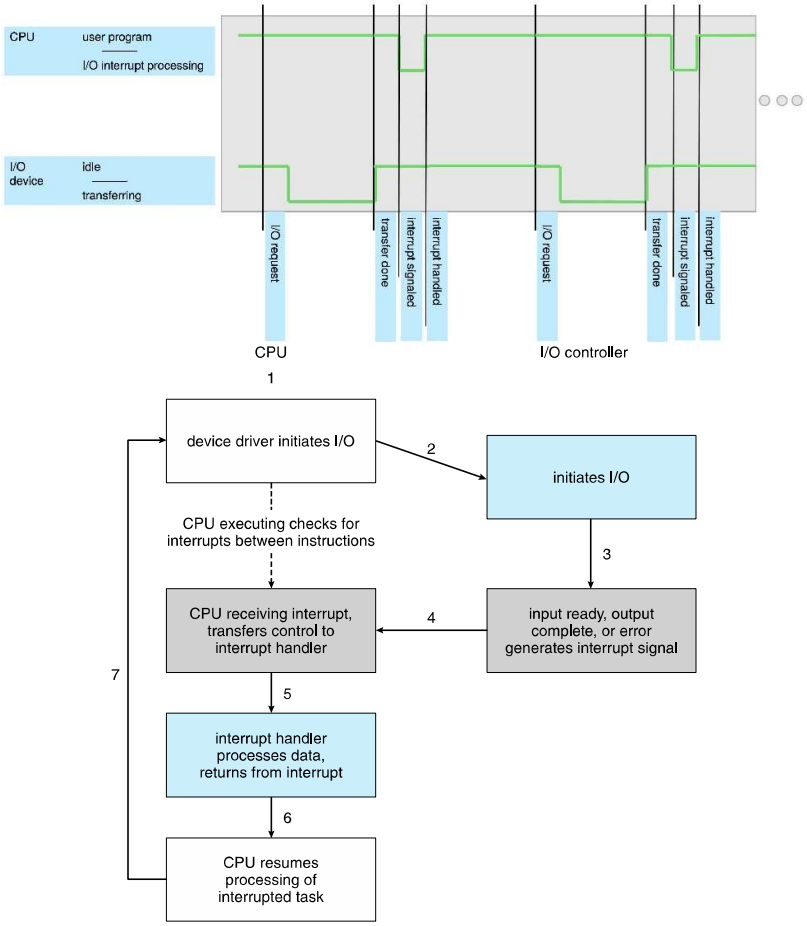
เมื่อคอมพิวเตอร์เริ่มทำงาน กระบวนการโดยทั่วไปมักเป็นเช่นนี้

- โปรแกรมพื้นฐานที่อยู่ในหน่วยความจำพิเศษที่กำหนดไว้ (ส่วนนี้จะวางตัวอยู่คงที่ โดยคงอยู่เสมอแม้ว่าไม่มีไฟเลี้ยงเข้าระบบ โดยทั่วไปมักใช้ EEPROM ในการจัดเก็บ) เริ่มต้นทำงาน โดยทั่วไปเราเรียกว่า bootstrap program หรือ firmware หน้าที่ของโปรแกรมพื้นฐานนี้ทำหน้าที่เซ็ตสภาพเริ่มต้นให้กับฮาร์ดแวร์ส่วนต่างๆ และโหลดเคอร์เนิลของระบบปฏิบัติการเข้ามาในหน่วยความจำ ซึ่งกลไกของการโหลดนี้อาจจะมีความง่ายหรือซับซ้อนขึ้นอยู่กับระบบปฏิบัติการต่างๆ ว่าจะเป็นเช่นใด (เช่นในระบบปฏิบัติการที่ใช้หน่วยความจำสำรองเป็นองค์ประกอบสำคัญ ตัว bootstrap program จะไปอ่านโปรแกรมเพื่อจัดการหน่วยความจำสำรองและโปรแกรมจัดการโหลดเคอร์เนิลจากตำแหน่งในดิสก์ที่กำหนดล่วงหน้า จากนั้นโปรแกรมที่ถูกโหลดเข้ามา ทำหน้าที่เตรียมสภาพเริ่มต้นให้กับหน่วยความจำสำรองแล้วโหลดเคอร์เนิลจากดิสก์ในตำแหน่งที่เหมาะสมต่อไปอีกทีหนึ่ง)

- เมื่อเคอร์เนิลถูกโหลดเข้ามาแล้ว โปรแกรม bootstrap ก็จะส่งการทำงานของระบบต่อให้กับเคอร์เนิล โดยสร้างโปรเซส (process) แรกขึ้นมาเพื่อรองรับการทำงานนี้ (ในลินุกซ์และยูนิกซ์ โปรเซสแรกนี้มีชื่อว่า init ในวินโดวส์มีชื่อว่า system) หน้าที่ของโปรเซสแรกนี้คือการรอคอยเหตุการณ์ (event) ที่จะเกิดขึ้นในระบบ (ในทางปฏิบัติ โพรเซสแรกนี้อาจจะโหลดโพรเซสอื่นๆที่จำเป็นต่อการใช้งานขึ้นมาทำงานไปด้วยได้)

<u>การจัดการ I/O</u>

- การจัดการ I/O มีสองแบบ
    - แบบแรก มีลักษณะที่การรับส่งข้อมูลกระทำกันระหว่างดีไวซ์กับซีพียู โดยการอินเทอร์รัพต์แต่ละครั้งเป็นการติดต่อเพื่อรับหรือส่งข้อมูลกันเป็นช่วงๆ ซึ่งการกระทำในลักษณะนี้ ซีพียูจำเป็นต้องรอคอยการส่งข้อมูลให้เสร็จสิ้น จึงจะดำเนินการใดๆ ต่อไปได้
    - แบบที่สอง ดีไวซ์รับส่งข้อมูลเข้าสู่หน่วยความจำหลักได้โดยตรง (Direct Memory Access) ซึ่งการทำงานเป็นลักษณะที่ซีพียูจะกำหนดพื้นที่หน่วยความจำหลักที่ดีไวซ์นั้นจะใช้งาน และกำหนดการอ่านหรือเขียนดีไวซ์ จากนั้นจะสั่งการให้ดีไวซ์ทำงาน และซีพียูสามารถทำงานต่อไปได้ โดยไม่ติดต่อหน่วยความจำหลักเป็นการชั่วคราว ในขณะที่ดีไวซ์กำลังเข้าถึงหน่วยความจำหลัก เมื่อดีไวซ์ทำงานเสร็จ ก็จะส่งอินเทอร์รัพต์แจ้งซีพียูให้ทราบว่าได้รับส่งข้อมูลกับหน่วยความจำหลักเสร็จสิ้นแล้ว
- การจัดการอินเทอร์รัปต์
    - เหตุการณ์ (event) ที่เกิดขึ้นโดยพื้นฐานแล้วมาจากการอินเทอร์รัพต์ (interrupt) จากฮาร์ดแวร์หรือซอฟต์แวร์ โดยซีพียูจะได้รับการอินเทอร์รัพต์ (trap / exception) จากฮาร์ดแวร์ที่เชื่อมต่ออยู่ (จาก interrupt controller) หรือเกิดจากภายในการทำงานของซีพียูเอง (เช่น devide by zero) ส่วนการอินเทอร์รัพต์ทางซอฟต์แวร์นั้นเกิดขึ้นในลักษณะของการเรียกใช้งานคำสั่งอินเทอร์รัพต์จากโพรเซสใดๆ ซึ่งเราเรียกการทำงานดังกล่าวว่า system call หรือ monitor call
    - เมื่อเกิดการร้องขอ (อินเทอร์รัพต์) ซีพียูจะหยุดการทำงานของโพรเซสปัจจุบันลง โดยจะจัดเก็บตำแหน่งการทำงานของชุดคำสั่งไว้ในตำแหน่งใดตำแหน่งหนึ่งขึ้นอยู่กับสถาปัตยกรรมของซีพียู จากนั้นก็จะไปเปิดตารางอินเทอร์รัพต์ (interrupt vector table) เพื่ออ่านค่า อินเทอร์รัพต์เว็กเตอร์ (interrupt vector) ซึ่งก็คือตำแหน่งเริ่มต้นการทำงานของชุดคำสั่งที่เตรียมไว้ให้ทำงานเมื่อเกิดอินเทอร์รัพต์ (ในทางปฏิบัติ การทำงานเริ่มโดยการจัดเก็บค่าต่างๆ ในเรจิสเตอร์ของซีพียูไว้เพื่อคืนค่าเมื่อการทำงานเสร็จลง) เมื่อการทำงานเสร็จสิ้นลง ก็จะสั่งกระโดดออกจากอินเทอร์รัพต์โดยการคืนค่าตำแหน่งการทำงานของโพรเซสที่ค้างไว้ให้ซีพียูได้ดำเนินการต่อ
    - ในการทำงานภายใต้การอินเทอร์รัพต์ โดยทั่วไปมักจะ disable อินเทอร์รัพต์อื่นๆ ไว้เพื่อกันการอินเทอร์รัพต์ซ้ำซ้อน
    - ระบบปฏิบัติการในปัจจุบันอาศัยอินเทอร์รัปต์เป็นหลักในการจัดการระบบ เราจึงมักเรียกระบบปฏิบัติการว่าเป็น interrupt driven (ทำงานขับเคลื่อนโดยอาศัยอินเทอร์รัปต์เป็นหลัก)



ประเภทของหน่วยควบคุมการรับส่งข้อมูล (I/O)

- Small Computer-Systems Interface (SCSI) สามารถเชื่อมกับดีไวซ์ได้ถึง 7 ตัว
- IDE (Indultrial Drive Electronics) หรือที่เรียกว่า ATAPI (AT Attachment with Packet Interface Extension) คำว่า AT มาจาก IBM PC-AT ซึ่งเป็นคอมพิวเตอร์ในยุคถัดมาของ PC-XT ที่บัสเชื่อมต่อเปลี่ยนมาใช้เป็น 16 บิตแล้ว
    - IDE/ATA-1 เป็นบัส 8 บิตสมัยเก่าใช้ใน IBM PC-XT
    - EIDE/ATA-2
    - UDMA/ATA-4 บัสของ DMA กว้างขึ้นจากเดิม ทำให้ส่งข้อมูลได้ 33MB/s
        - มีรุ่นที่เป็น 33/66/100 (รุ่น 133 ไม่ได้รับให้เป็นมาตรฐาน)
- SATA (Serial Advanced Technology Attachment) เปลี่ยนการรับส่งจากแบบขนานมาเป็นอนุกรม SATA-1 (1.5Gb/s) SATA-2 (3.0Gb/s) SATA-3 (6Gb/s)
- USB (Universal Serial Bus) USB1.1 (1.5/12 Mb/s) USB 2.0 (1.5/12/480 Mb/s) USB3.0 (สูงสุดถึง 4000 Mb/s) และปัจจุบัน 3.1Gen1(USB3.2Gen1 - SuperSpeed 5Gbps) 3.1Gen2(USB3.2Gen2 -SuperSpeed 10Gbps) 3.2Gen2x2 (SuperSpeed 20Gbps) และในอนาคตอันใกล้ USB4 (ที่จะมีโปรโตคอล Thumderbolt 3) เป็นโปรโตคอลฐานและรองรับ USB3.2/2.0 ความเร็ว 40Gbps)
- Firewire (400/800) (ปัจจุบันแทบจะไม่มีให้เห็นใช้งานแล้ว)
- Thunderbolt ซึ่งถูกพัฒนาขึ้นโดยบริษัทอินเทล นิยมนำมาใช้ในงานเช่นเดียวกับที่ USB ใช้งาน และเนื่องจากมีความเร็วในการส่งถ่ายข้อมูลสูง จึงถูกนำมาใช้ในการส่งข้อมูลให้จอแสดงผลด้วย ปัจจุบันเป็นเวอร์ชัน 3 (ซึ่งรองรับ DisplayPort 1.2 2

streams และ USB3.1gen2) (40Bbit/s bidirectional, 80Gbit/s one-way) และล่าสุดเป็นเวอร์ชัน 4 (รองรับ DisplayPort 2.0 และ USB4)

ระบบการจัดเก็บข้อมูล

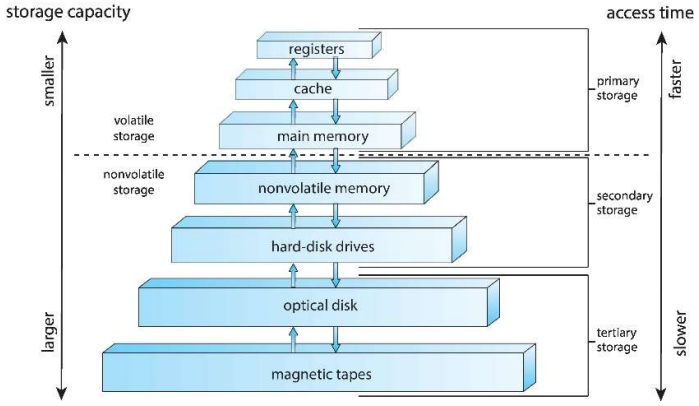
- คอมพิวเตอร์ปัจจุบันแบ่งระบบการจัดเก็บข้อมูลออกเป็น
  - หน่วยความจำหลัก (main memory or primary memory)
    - มักจะเป็นหน่วยความจำที่เข้าถึงพื้นที่ใดๆ โดยใช้ค่าเลขที่อยู่ ได้ตามต้องการ (random access)
    - มักจะเป็นหน่วยความจำที่ไม่สามารถเก็บข้อมูลได้เมื่อไม่มีไฟเลี้ยง (volatile)
      - มีการพัฒนาร่วมกันระหว่าง Micron กับ Intel ผลิต 3D X-point (อ่านว่า ครอสพอยต์) ที่เป็น non-volatile และ CPU intel รุ่นใหม่ๆ รองรับการเข้าถึงโดยตรงได้ (NVDIMM- non volatile main memory) (ติดต่อผ่าน memory interface) ปัจจุบันถูกนำมาใช้ทั้งเป็นหน่วยความจำสำรอง เพื่อเก็บข้อมูลแบบเดียวกันกับ SSD และทั้งใช้งานแบบหน่วยความจำหลักที่เป็น non-volatile
    - สำหรับระบบคอมพิวเตอร์ขนาดเล็ก อาจใช้ Static RAM (เข้าถึงได้เร็ว และใช้พื้นที่มากกว่า)
      - ซีพียูปัจจุบันนำเอา SRAM มาใช้ทำ cache บนซีพียู
    - คอมพิวเตอร์ในปัจจุบันใช้ DRAM (Dynamic RAM) เป็นหน่วยความจำหลัก (เข้าถึงได้ช้ากว่า SRAM เพราะต้อง refresh ข้อมูลตลอดเวลา แต่ใช้พื้นที่น้อยกว่า และทำให้ราคาถูกกว่า)
  - หน่วยความจำสำรอง (Secondary storage/memory) เป็นหน่วยความที่พัฒนาเพิ่มขึ้นมาเพื่อใช้ทำหน้าที่หลักในการจัดเก็บข้อมูลและโปรแกรมที่ไม่ใช้ใช้คำนวณ ณ ขณะนั้นๆ และมักจะใช้หน่วยความจำที่เป็น non-volatile
    - หน่วยความจำสำรองที่นิยมใช้กันในปัจจุบันมักจะใช้เทคโนโลยีจานแม่เหล็ก (เช่นฟล็อปปีดิสก์และฮาร์ดดิสก์) ในยุคใหม่เริ่มมีหน่วยความจำที่ใช้สารกึ่งตัวนำพวก Flash Memory เข้ามามากขึ้น และยังนำเอาหน่วยความจำแบบ RAM ที่เป็น volatile มาใส่วงจรไฟเลี้ยงและแบตเตอร์รี) ไว้ตลอดเพื่อให้เก็บข้อมูลได้แม้ไม่มีไฟเลี้ยงจากภายนอก เรียกว่า Non-volatile RAM (NVRAM)
    - หน่วยความจำสำรองที่มีขนาดใหญ่ขึ้นไปอีก และมักจะใช้เวลานานในการเข้าถึงและการจัดเก็บ จุดประสงค์ของหน่วยความจำประเภทนี้มักใช้ในการจัดเก็บข้อมูลที่ไม่ใช้บ่อยจากในหน่วยความจำสำรอง ยกตัวอย่างเช่น CD-ROM DVD-ROM และเทปแม่เหล็ก เป็นต้น

โครงสร้างทั่วไปของหน่วยความจำ

- Turing Machine (คิดโดย Alan Turing) มีหลักการคือเครื่องที่จะอ่านแถบแม่เหล็กโดยแบ่งพื้นที่ของแถบแม่เหล็กเป็นส่วนๆ เครื่องจะอ่านไปทีละส่วนและประมวลผลตามคำสั่งที่กำหนดไว้ จากนั้นแถบแม่เหล็กนั้นๆ อาจจะถูกเปลี่ยนแปลงข้อมูลได้ด้วย และเมื่อประมวลคำสั่งเสร็จหนึ่งพื้นที่ ก็จะเลื่อนพื้นที่ถัดไปขึ้นมาประมวลต่อ โดยการประมวลอาจจะเดินหน้าไปเรื่อยๆ หรือกระโดดกลับไปมาแล้วแต่ตามต้องการได้
- von Neumann Atchitecture (หรือ von Neumann Machine เสนอโดย John von Neumann) หลักการพัฒนามาจาก Turing Machine แต่ส่วนเก็บข้อมูลที่จะถูกประมวลผลนั้นคือหน่วยความจำแบบที่เขียนอ่านได้ (Read-Write, Random Access Memory)
- ยังมีหน่วยความจำที่ปรากฏบนเมนบอร์ดคอมพิวเตอร์ และภายในซีพียูปัจจุบัน ซึ่งทำหน้าที่สำรองข้อมูลที่ใช้บ่อยจากหน่วยความจำหลัก ทั้งนี้เพราะซีพียูปัจจุบันมีความเร็วสูงมากกว่าหน่วยความจำหลัก หน่วยความจำในระดับที่เร็วสุดและใกล้ส่วนวงจรการคำนวณสูงสุดเรียกว่า เรจิสเตอร์ (Register) ส่วนที่ไกลออกไป ซึ่งมักจะแบ่งออกเป็นส่วนจัดเก็บชุดคำสั่ง และส่วนเก็บข้อมูล เรียกว่า แคช (Cache) ซึ่งปัจจุบัน แคชก็ยังแบ่งออกเป็นหลายระดับ Cache Level1 จัดเก็บ

ข้อมูลที่ส่วนการคำนวณสามารถดึงไปใช้ได้โดยตรง (มักแบ่งออกเป็นสองส่วน ส่วนแรกสำหรับเก็บชุดคำสั่ง ส่วนที่สองเก็บข้อมูล โดยจะโหลดลงเรจิสเตอร์หรืออ่านจากเรจิสเตอร์ไปเก็บ) Level2 จัดเก็บข้อมูลที่คาดว่าจะถูกนำไปใช้ หรือที่พึ่งนำไปใช้หรือคำนวณเสร็จ (ข้อมูลจะถูกส่งถ่ายไปยัง Level 1 หรือส่งมาจาก Level1) และปัจจุบันอาจพบ Level 3 ซึ่งมักจะปรากฏในซีพียูที่มีหน่วยประมวลผลย่อยมากกว่าหนึ่งหน่วย โดยในระดับนี้จะใช้ร่วมกันระหว่างแต่ละหน่วยย่อย

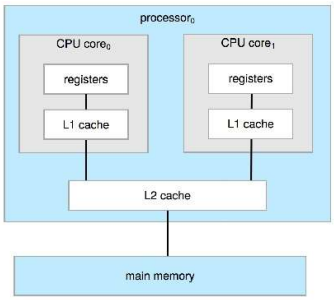
- ระดับชั้นความเร็ว/สะดวก ในการเข้าถึงหน่วยความจำ registers < cache < main memory < electronic disk < magnetic disk < optical disk < magnetic tapes



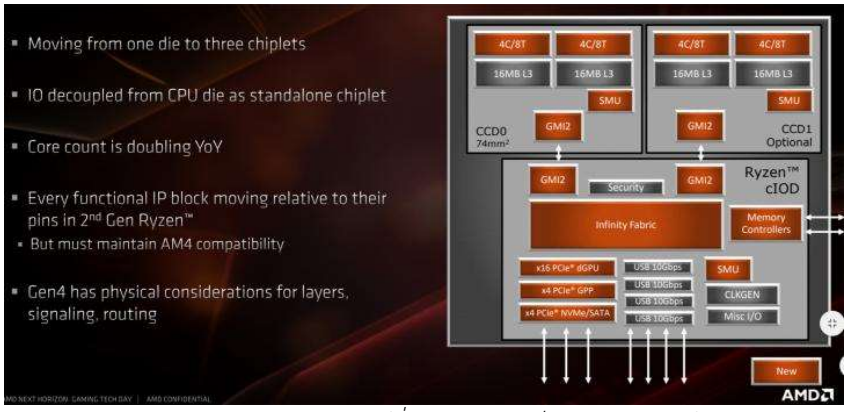
ประเภทสถาปัตยกรรมคอมพิวเตอร์ เมื่อพิจารณาจากโครงสร้างของซีพียู

- **Single-Processor Systems** เป็นระบบคอมพิวเตอร์ที่มีหน่วยประมวลผลกลางเพียงหน่วยเดียว ซึ่งเป็นโครงสร้างของคอมพิวเตอร์ในยุคก่อนหน้า และมีใช้ต่อมาจนถึงยุคปัจจุบัน
- **Multi-Processor Systems (หรือ Parallel System หรือ Tightly Coupled Systems)** เป็นระบบคอมพิวเตอร์ที่มีหน่วยประมวลผลกลางมากกว่าหนึ่งตัว โดยอาจจะแชร์บัสเชื่อมต่อหน่วยความร่วมกัน และดีไวซ์พื้นฐานต่างๆ ร่วมกันในระดับใดระดับหนึ่ง ข้อดีของระบบหลายซีพียูนี้คือ
    - การเพิ่มประสิทธิภาพของการคำนวณ (Increased throughput) (ทำงานเสร็จเร็วขึ้น)
    - ความประหยัดคุ้มค่าในการขยายประสิทธิภาพ (Economy of scale) การเพิ่มจำนวนหน่วยประมวลผลทำได้สะดวกกว่าการออกแบบซีพียูตัวเดียวที่เร็วมากๆ
    - การเพิ่มความมั่นคงของระบบ (Increased Reliability) การออกแบบฮาร์ดแวร์และซอฟต์แวร์ที่เหมาะสม จะทำให้เมื่อซีพียูตัวหนึ่งตัวใดเกิดปัญหา ก็จะสามารถโอนถ่ายงานไปให้ซีพียูตัวอื่นทำได้โดยไม่มีผลกระทบต่อระบบโดยรวม
        - Graceful degradation ยังคงทำงานต่อได้ แต่ประสิทธิภาพการทำงานจะเริ่มตกลง และงานบางส่วนอาจเสียหายต้องเริ่มต้นใหม่
        - Fault tolerant ยังคงทำงานต่อไปได้โดยที่งานใดๆ ที่ทำอยู่จะไม่มีผลกระทบใดๆ (ผู้ใช้จะไม่รับทราบเลยว่าเกิดความผิดพลาดเกิดขึ้น) ระบบนี้มักจะเห็นในคอมพิวเตอร์ทำงานในสภาพแวดล้อมแบบวิกฤต เช่นในยานอวกาศ ดาวเทียม เป็นต้น
- ระบบแบบหลายซีพียูนี้มีสองแบบ
    - Asymmetric Multiprocessing จะมีซีพียูอย่างน้อยหนึ่งตัวทำหน้าที่แจกจ่ายงานไปให้ซีพียูตัวอื่นๆทำ (เป็น Master/Slave)

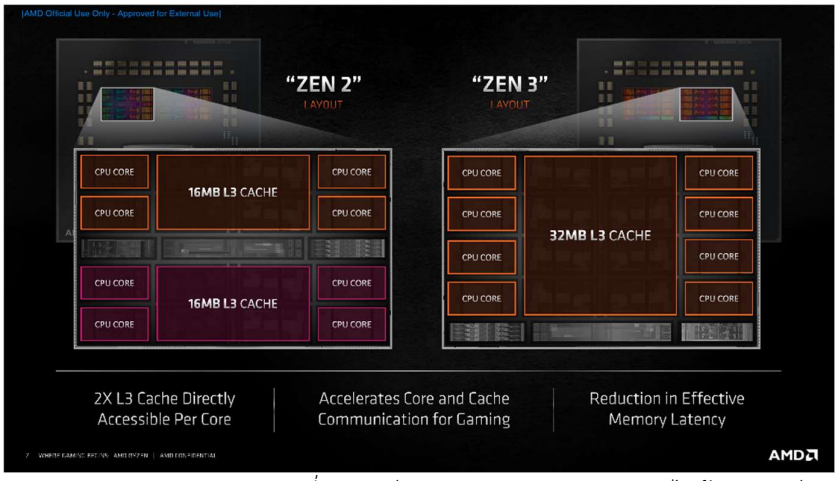
- Symmetric Multiprocessing ซีพียูแต่ละตัวจะมีหน้าที่การทำงานเสมอกัน โดยงานต่างๆ ของซีพียูจะได้รับจัดสรรให้ทำงานบนซีพียูใดๆ ได้ทุกตัว และงานของผู้ใช้เองก็สามารถถูกจัดสรรการทำงานบนซีพียูได้ทุกตัว (ระบบปฏิบัติการก็จะทำงานอยู่ได้บนทุกๆ คอร์ และสามารถถ่ายโอนการทำงานระหว่างคอร์ไปมาได้โดยอิสระ)
- Multi-core design ซีพียูในปัจจุบัน เริ่มหันมาใช้ระบบ Symmetric Multiprocessing มากขึ้นเรื่อยๆ โดยการออกแบบชิพให้มีหน่วยประมวลผลมากกว่าหนึ่งหน่วย



ตัวอย่างลักษณะของซีพียูแบบ Dual-core



ตัวอย่างลักษณะซีพียูแบบ 16 คอร์ที่แบ่งกลุ่มออกเป็นกลุ่มละ 4 คอร์ (Zen 2)

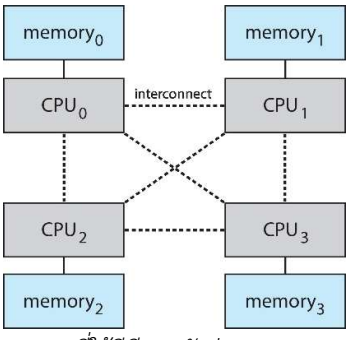


ความแตกต่างระหว่าง Zen2 ที่กลุ่มคอร์ (CCX-core complex) ประกอบไปด้วย 4 คอร์ และ Zen3 ที่กลุ่มคอร์ประกอบไปด้วย 8 คอร์

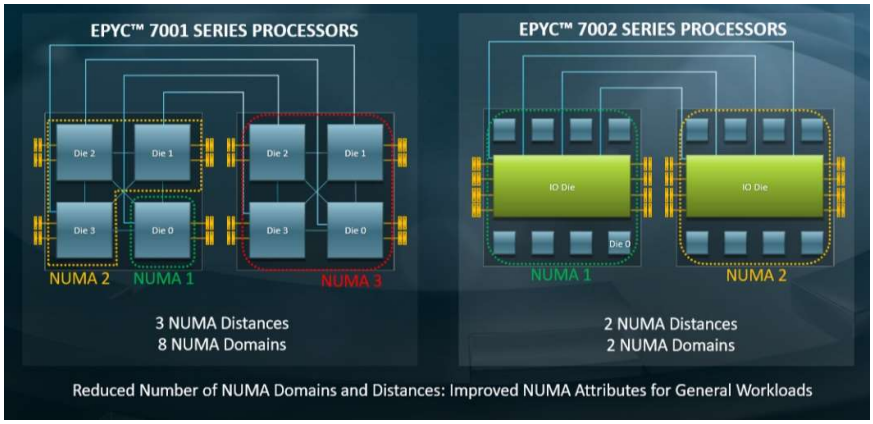
- การรับส่งข้อมูลระหว่างซีพียูในระบบหลายซีพียูนี้อาจทำในลักษณะเป็นบัสพิเศษใช้ร่วมกัน หรืออาจจะเป็นแบบ switch ซึ่งจะสามารถสลับเลือกเส้นทางการส่งข้อมูลจากหน่วยประมวลผลหนึ่งๆ ไปยังหน่วยอื่นได้โดยสะดวก

- ซีพียูในยุคแรกๆ อย่างเช่น Pentium D ประกอบไปด้วยซีพียู 2 คอร์ ติดต่อซึ่งกันและกันผ่าน FSB (Front-side bus) ซึ่งเป็นช่องทางสื่อสารที่ซีพียูต้องใช้กับอุปกรณ์รอบข้างอื่นๆ ด้วย
- ซีพียูปัจจุบันมักจะใช้บัสความเร็วสูงแยกออกมาจากช่องทางปกติ ในการติดต่อสื่อสารระหว่างกัน ตัวอย่างเช่นอินเทลใช้ ring bus ระหว่างคอร์บนชิพเดียวกันในซีพียูที่มีจำนวนคอร์ไม่มาก และการเชื่อมต่อแบบ mesh interconnect สำหรับซีพียูในระดับเวิร์คสเตชัน ที่มีจำนวนคอร์มากขึ้น และ QPI (QuickPath Interconnect) ระหว่างซีพียูในเซอร์เวอร์ ซึ่งมีซีพียูมากกว่าหนึ่งตัวบนบอร์ดเดียวกัน สำหรับ AMD ก่อนหน้านี้ได้ใช้ Hypertransport ต่อมาในปัจจุบันใช้ Infinity fabric เพื่อเป็นช่องทางสื่อสารระหว่างคอร์ทั้งบนชิพเดียวกันและระหว่างซีพียู (และระหว่างอุปกรณ์อื่น)
- ในปัจจุบันเริ่มมีการพัฒนาบัสความเร็วสูงใหม่ชื่อ Compute Express Link (CXL) เพื่อใช้สื่อสารระหว่างซีพียูกับหน่วยความจำ และซีพียูกับอุปกรณ์ต่างๆ โดยออกแบบให้ทำงานบนบัส PCI express (เริ่มที่ gen 5)

- ในการเชื่อมต่อซีพียูมากกว่าหนึ่งชิพเข้าด้วยกัน ซึ่งในปัจจุบันจะมี memory controller ประจำแต่ละชิพ ทำให้การเข้าถึงหน่วยความจำจากซีพียูคอร์หนึ่งๆ ไปยังหน่วยความจำแต่ละส่วน อาจใช้เวลา/เส้นทาง ที่แตกต่างกัน ทำให้เกิดลักษณะการเข้าถึงหน่วยความจำไม่เป็นแบบเดียวกัน Non-Uniform Memory Access (NUMA) ซึ่งส่งผลต่อประสิทธิภาพการทำงานของซอฟต์แวร์ผู้ใช้ หากระบบปฏิบัติการไม่สามารถจัดสรรซีพียูคอร์และหน่วยความจำที่เหมาะสมให้กับผู้ใช้



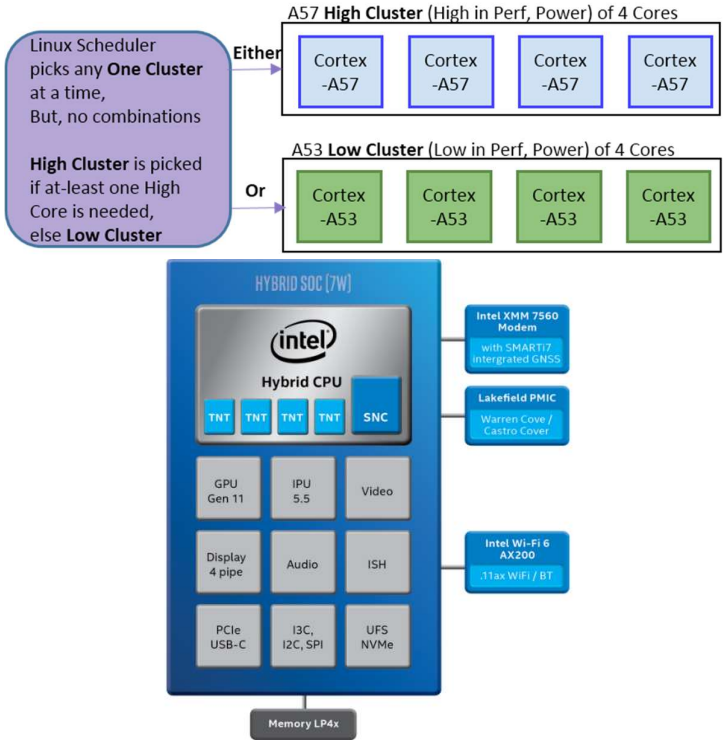
ลักษณะการเชื่อมต่อซีพียู Xeon ที่ใช้ซีพียู 4 ตัวผ่าน QPI และหน่วยความจำแบบ NUMA



ลักษณะการแบ่งโหนด NUMA ในซีพียูตระกูล Epyc รุ่นแรก และรุ่นที่สองของ AMD

- ปัจจุบันยังมีระบบคอมพิวเตอร์ในลักษณะหลายตัวอยู่ในเคสเดียวกัน (blade server) โดยถูกออกแบบมาเพื่อความสะดวกในการดูแลและการประหยัดพื้นที่ (มักใช้ในงานเซอร์เวอร์)

- ในปัจจุบัน มีการออกแบบผสมผสานซีพียูที่มีลักษณะเป็น Heterogeneous โดยใช้ซีพียูหลายคอร์ซึ่งมีสถาปัตยกรรมต่างกันรวมเข้าบนชิปเดียวกัน ทั้งนี้เพื่อให้ซีพียูตัวเล็กกว่าที่ประหยัดพลังงานกว่า (เพราะมีชุดคำสั่งรองรับน้อยกว่า และวงจรมีขนาดเล็กกว่า) ทำงานในช่วงที่ระบบไม่ต้องใช้พลังการประมวลผลมากนัก (และไม่ต้องการใช้ชุดคำสั่งพิเศษ) แต่เมื่อต้องการพลังการคำนวณมากขึ้น ก็จะให้โพรเซสผู้ใช้หันไปใช้ซีพียูตัวที่ใหญ่กว่า ที่มีพลังการคำนวณมากกว่า (แต่กินไฟมากกว่า) ส่งผลทำให้คอมพิวเตอร์ดังกล่าวสามารถประหยัดไฟมากกว่าที่จะให้ซีพียูตัวใหญ่ทำงานอย่างเดียว
  - ซีพียูตระกูล ARM เรียกการทำงานแบบนี้ว่าbig.LITTLE ซึ่งปัจจุบันใช้กันอย่างแพร่หลายในสมาร์ทโฟน
  - ซีพียูของ Intel ในปัจจุบันมีการออกแบบซีพียูตระกูล Lakefield i5-L16G7 และ i3-L13G4 ที่ใช้ซีพียูประสิทธิภาพสูงจากสถาปัตยกรรม Sunny Cove และซีพียูประหยัดพลังงานจาก Tremont small cores (ที่ใช้ในซีพียู Atom)
  - ซีพียูของ Intel ในอนาคตอันใกล้ในชื่อ Alder Lake จะมีลักษณะเป็น Heterogeneous ซึ่งจะมีจำนวนคอร์ใหญ่และคอร์เล็กมากขึ้น (คาดว่ารุ่นสูงสุดจะมีคอร์ใหญ่ 8 คอร์ และคอร์เล็ก 8 คอร์)



- **<u>ระบบแบบคลัสเตอร์ (Clustered Systems)</u>**
  - มักสร้างมาจากคอมพิวเตอร์แบบใดแบบหนึ่งข้างต้น นำมาเชื่อมต่อกันด้วย LAN และมีซอฟต์แวร์ที่ช่วยทำหน้าที่ดูแลกระจายงานและข้อมูลระหว่างคอมพิวเตอร์ด้วยกัน
  - ระบบแบบนี้มักใช้หน่วยความจำสำรองร่วมอีกส่วนหนึ่งที่เชื่อมบน LAN ด้วย (Storage-Area Network – SAN)
  - ระบบแบบคลัสเตอร์ มีข้อเด่นที่สามารถเพิ่มความมั่นคงในระบบได้ดีขึ้น โดยหากคอมพิวเตอร์หน่วยใดเสียหาย หน่วยอื่นก็สามารถรับหน้าที่การทำงานต่อไปได้โดยที่ระบบรวมไม่สะดุด (high-availability)
    - Asymmetric clustering จะมีเครื่องอย่างน้อยเครื่องหนึ่งอยู่ในสภาพ stand-by เตรียมเฝ้าดูว่าเครื่องใดล่ม ก็จะเข้าทำงานแทนเครื่องนั้นต่อไปในทันที

- Symmetric clustering เครื่องแต่ละเครื่องมีซอฟต์แวร์คอยตรวจดูการทำงานของเครื่องอื่นๆ และเข้าทำงานแทนกันและกันในกรณีที่เครื่องใดเครื่องหนึ่งเกิดล่ม

- ระบบคอมพิวเตอร์แบบคลัสเตอร์นี้ถูกนำไปใช้ในหลายงาน อาทิเช่น
  - เซอร์เวอร์ที่ให้บริการผู้ใช้จำนวนมากๆ โดยการกระจายงานของผู้ใช้แต่ละกลุ่มไปยังหน่วยคอมพิวเตอร์แต่ละหน่วย
  - การประมวลผลแบบขนานที่มีประสิทธิภาพสูง (High-performance computing HPC) โดยในกรณีเช่นนี้ซอฟต์แวร์เองก็จะต้องถูกพัฒนาให้สามารถแตกงานย่อยออกได้เป็นจำนวนมากเพื่อส่งงานและข้อมูลเข้าไปประมวลในคอมพิวเตอร์แต่ละตัว

1.3 โครงสร้างพื้นฐานของระบบปฏิบัติการ

โดยทั่วไปแล้ว งานที่ผู้ใช้จะสั่งการให้คอมพิวเตอร์ประมวลนั้น มักจะเสียเวลาในการเตรียมงานเพื่อป้อนให้คอมพิวเตอร์มากกว่าเวลาที่คอมพิวเตอร์ประมวลผลจริง ดังนั้น คอมพิวเตอร์จึงมักจะอยู่ในสภาพที่รอคอยงาน (idle) อยู่เกือบตลอดเวลา

เพื่อให้คอมพิวเตอร์สามารถถูกใช้งานได้อย่างคุ้มค่า คอมพิวเตอร์หนึ่งเครื่อง จึงมักถูกสั่งงานหลายๆ งานไปเตรียมรอไว้เพื่อให้คอมพิวเตอร์ประมวลผล

ข้อดีของการประมวลผลแบบหลายงาน (multiprogramming)

- ทำให้ใช้คอมพิวเตอร์ได้คุ้มค่ามากขึ้น เพราะผู้ใช้เพียงคนเดียว(และให้งานเพียงงานเดียว) ไม่สามารถทำให้ CPU และ I/O ทำงานได้ตลอดเวลา
- ระบบประมวลผลหลายงาน จะจัดการงานต่างๆ ที่ถูกส่งเข้าไปในระบบ (Job ซึ่งประกอบไปด้วยชุดคำสั่งและข้อมูล) ดังนั้น ซีพียูจะได้รับงานไปทำได้อย่างต่อเนื่อง
- งาน (Job) จำนวนหนึ่ง (จากทั้งหมดที่มอบหมาย) จะถูกอ่านเข้าไปเก็บไว้ในหน่วยความจำหลักเพื่อรอคิวในการประมวลต่อไป
- Job Scheduling จัดการเลือกงานที่รอคอย นำมาประมวลโดยทำให้เกิดประสิทธิภาพการใช้งานคอมพิวเตอร์สูงสุด
- เมื่องานใดต้องรอรับส่งข้อมูลกับ I/O ระบบปฏิบัติการจะสลับเอางานอื่นขึ้นมาทำแทนในทันที (เพื่อซีพียูจะได้ไม่ต้องรอในขณะที่ข้อมูลจะรับส่งเสร็จสิ้น

ในระบบสมัยเก่าเช่นในเมนเฟรม งานแต่ละงานจะถูกโหลดขึ้นสู่หน่วยความจำหลักรอไว้ และงานแต่ละตัวจะถูกทยอยนำเข้าไปประมวล โดยระบบปฏิบัติการจะทำหน้าที่ Job scheduling ดังนั้นงานแต่ละงานจะดำเนินไปจนกว่าจะเสร็จสิ้น งานอื่นจึงจะมีโอกาสได้รันบ้าง

ระบบปัจจุบันหันมาใช้ในรูปแบบของการแบ่งเวลาครอบครองซีพียูออกเป็นส่วนย่อยๆ แล้วให้โอกาสงานแต่ละงานเข้าไปประมวล **งาน (Process)** ในที่นี้มีความหมายในเชิงคล้ายคลึงกันกับ Job (Job ถูกเรียกถึงโปรแกรมและข้อมูลที่ถูกโหลดขึ้นสู่หน่วยความจำหลักเพื่อรอประมวลในคอมพิวเตอร์สมัยเก่า ในขณะที่ Process คือโปรแกรมและข้อมูลที่ถูกโหลดขึ้นสู่หน่วยความจำหลักที่อาจจะรอประมวลผลอยู่หรือกำลังประมวลผลอยู่ในขณะนั้น ส่วน Program คือชุดคำสั่งและอาจจะประกอบด้วยข้อมูลคงที่บางส่วนที่เก็บไว้เป็นไฟล์ในหน่วยความจำสำรอง ที่ระบบปฏิบัติการจะอ่านขึ้นมาและจัดเตรียมพื้นที่ในหน่วยความจำเพื่อวางชุดคำสั่ง ข้อมูลเดิม ข้อมูลอื่นๆ ที่งานนั้นต้องการเพิ่มเติม และพื้นที่สแต็ก (กลายเป็นงานหนึ่งงาน) ทำให้บางทีเราอาจเรียก program ว่าเป็น passive entity ในขณะที่งาน process ถูกเรียกเป็น active entity)

- รอบการประมวล (เวลาที่งานหนึ่งถูกประมวล และต้องหยุดลงเพื่อให้งานอื่นได้โอกาสมาประมวลบ้าง จนกระทั่งวนกลับมายังงานเดิมอีกครั้ง) มักจะมีค่าเวลาน้อยๆ เช่นน้อยกว่าหนึ่งวินาทีเป็นต้น
- ในระบบหลายผู้ใช้ ผู้ใช้แต่ละคนก็จะมีงานอย่างน้อยหนึ่งตัวอยู่ในหน่วยความจำหลักที่กำลัง(ผลัดกัน)ประมวลผลอยู่
- ในกรณีที่มีงานมากกว่าหนึ่งงานกำลังทำงานอยู่พร้อมกัน ก็เป็นหน้าที่ของระบบปฏิบัติการที่จะทำหน้าที่สลับโอกาสการประมวลผลหรือเข้าครอบครองซีพียูไปมาระหว่างงานแต่ละตัว CPU scheduling
- หากหน่วยความจำหลักไม่พอใช้งานในขณะใดขณะหนึ่ง ก็จะต้องมีงานบางตัวที่ถูกย้ายไปเก็บไว้ในหน่วยความจำสำรองเพื่อเปิดโอกาสให้งานอื่นเข้ามาครอบครองพื้นที่หน่วยความจำหลักเพื่อเปิดโอกาสให้ประมวลได้ (ซีพียูไม่สามารถเข้าถึงพื้นที่หน่วยความจำสำรองได้โดยตรง) กลไกการย้ายเข้าออกของงานระหว่างหน่วยความจำหลักและหน่วยความสำรองนี้เรียกว่า swapping
- โดยอาศัยการจัดการหน่วยความจำเสมือน Virtual Memory (จะเรียนต่อไปในภายหลัง) ทำให้สามารถจัดการงานหลายๆ งานโดยงานบางงานอาจจะถูกระบบปฏิบัติการส่งให้ครอบครองพื้นที่หน่วยความจำหลักและหน่วยความจำสำรอง โดยที่ตัวงานนั้นๆ (และผู้ใช้) ยังคงมองเห็นราวกับว่าตัวงานนั้นครอบครองพื้นที่หน่วยความจำหลักอยู่เพียงอย่างเดียว

ลักษณะการบริหารจัดการการประมวลผลของระบบปฏิบัติการ

- ระบบปฏิบัติการ ทำงานในลักษณะ inturrupt driven นั่นคือติดต่อประสานการทำงานกับดีไวซ์ต่างๆ รอบตัวด้วยกลไกของการอินเทอร์รัพต์
- เมื่อซอฟต์แวร์ทำงานผิดพลาด จะเกิด exception หรือ trap ขึ้นเพื่อหยุดการทำงานของตน และส่งต่อการทำงานให้ระบบปฏิบัติการได้ตัดสินใจต่อ
    - เช่น Divide by Zero ที่จะถูกกระตุ้นจากซีพียู เป็น hardware interrupt ชนิดหนึ่ง
- ในระบบปฏิบัติการที่ดี จะต้องมีกลไกการดึงคืนการครอบครองทรัพยากรของระบบ (โดยเฉพาะเวลาครอบครองซีพียู) จากงานที่ทำงานไม่เหมาะสม เช่นการทำงานวนรอบโค้ดจุดเดิมไปเรื่อยๆ ไม่สิ้นสุด (infinite loop) การครอบครองทรัพยากรไว้ไม่ปล่อย ในลักษณะที่ทำให้งานแต่ละตัวไม่สามารถดำเนินต่อไปได้ (deadlock) การเข้าถึงหรือ แก้ไขพื้นที่ชุดคำสั่งของตนเอง หรือการเข้าถึงหรือแก้ไขข้อมูลหรือชุดคำสั่งของงานอื่น หรือของระบบปฏิบัติการโดยมิได้รับอนุญาต
- เพื่อเป็นการป้องกันไม่ให้งานของผู้ใช้ เข้ามาเปลี่ยนแปลงงานของระบบปฏิบัติการโดยมิได้รับอนุญาต ฮาร์ดแวร์และระบบปฏิบัติการปัจจุบันจึงถูกออกแบบให้งานแต่ละตัวสามารถมีระดับสิทธิ (privilege) ในการทำงานได้อย่างน้อยสองระดับ เราเรียกระบบการจัดการนี้ว่า **dual-mode operation** สิทธิสองระดับนี้คือ
    - **User mode** เป็นโหมดปกติที่งานของผู้ใช้ทำงาน และ
    - **Kernel mode** เป็นโหมดที่งานของระบบปฏิบัติการทำงาน
- ฮาร์ดแวร์จะรองรับ mode bit โดยจะถูกเซ็ตโดยระบบปฏิบัติการในช่วงที่งานจะได้รับการครอบครองซีพียูเพื่อทำงาน
    - ดังนั้นในขณะใดขณะหนึ่ง ซีพียูจะรู้ว่าตนกำลังทำงานของผู้ใช้ในระดับปกติ หรือในระดับ kernel
    - ชุดคำสั่ง (instruction) ของซีพียูบางตัวจะสงวนไว้สำหรับ kernel mode เท่านั้น ดังนั้นหากงานที่อยู่ในระดับ user mode พยายามทำชุดคำสั่งดังกล่าว ก็จะเกิด exception ขึ้น
    - เมื่องานในระดับผู้ใช้เรียกชุดคำสั่งที่เตรียมไว้โดยระบบปฏิบัติการ (system call) โหมดการประมวลผลจะถูกยกระดับจาก user mode ขึ้นสู่ kernel mode เป็นการชั่วคราว โดยสิทธิจะคืนกลับเป็น user mode เมื่อการประมวลชุดคำสั่งที่ระบบปฏิบัติการได้เตรียมไว้นั้นเสร็จสิ้นลง (และส่งการทำงานต่อไปยังการทำงานปกติของงานนั้นๆ ต่อไป)
    - ระบบปฏิบัติการ มีกลไกการป้องกันการครอบครองทรัพยากรของงานใดงานหนึ่งเกินความเหมาะสมได้โดยอาศัย timer counter ที่เป็นฮาร์ดแวร์ในระบบ โดยระบบปฏิบัติการเซ็ตค่าฐานเวลานับไว้ จากนั้นเซ็ตค่าตัวนับ

โดยค่าตัวนับจะลดลงทีละหนึ่งเมื่อฐานเวลาครบตามกำหนด เมื่อตัวนับถอยลงถึงศูนย์ ก็จะเกิด interrupt ขึ้น ซีพียูจะหยุดงานที่กำลังกระทำลงชั่วคราวเพื่อทำงานพิเศษที่ระบบปฏิบัติการเตรียมไว้ โดยระบบปฏิบัติการก็จะสามารถตรวจสอบได้ว่ามีงานใดกำลังทำงานค้างด้วยเวลาเกินกว่าที่กำหนดหรือไม่อย่างไร เช่น หยุดรอสิ่งใดสิ่งหนึ่งเกินกว่าช่วงเวลาที่กำหนด ระบบปฏิบัติการก็จะได้สามารถแจ้งผู้ใช้ให้หยุดงานนั้นๆ หรืออาจจะหยุดการทำงานของงานนั้นลงโดยอัตโนมัติก็ได้

1.3.1 การจัดการงานของระบบปฏิบัติการ (Process management)

งาน (process) คือชุดคำสั่งและข้อมูลที่ได้รับการจัดสรรพื้นที่หน่วยความจำหลักเพื่อให้ดำเนินการได้ ซึ่งเราอาจเรียกว่าเป็น active entity ชุดคำสั่งและข้อมูลเหล่านี้ โดยทั่วไปมักจะถูกจัดเตรียมไว้ในหน่วยความจำสำรองก่อนเพื่อความสะดวกในการเรียกหาโดยระบบปฏิบัติการ ชุดคำสั่งที่วางตัวอยู่ในหน่วยความจำสำรองที่มิได้ถูกประมวลผลนี้เรียกว่า โปรแกรม (program) ดังนั้นเราอาจเรียกได้ว่าเป็น passive entity)

เมื่อระบบปฏิบัติการอ่านโปรแกรมขึ้นมาเพื่อสร้างเป็นงาน ระบบปฏิบัติการก็จะต้องเตรียมทรัพยากรที่งานนั้นต้องการ ซึ่งประกอบไปด้วยเวลาในการครอบครองซีพียู หน่วยความจำ(หลัก) I/O ที่งานนั้นๆ ต้องการใช้ และไฟล์ในหน่วยความจำสำรอง รวมทั้งจะต้องเตรียมข้อมูลตั้งต้นที่งานนั้นๆ ต้องใช้ (PCB และอื่นๆ)

เมื่องานเสร็จสิ้นลง (process termination) ก็เป็นหน้าที่ของระบบปฏิบัติการที่จะดึงคืนทรัพยากรทั้งหมดที่มอบให้กับงานนั้นๆ กลับมาเพื่อนำไปให้งานอื่นๆ ได้ใช้ต่อไป

งานที่เป็นแบบ single thread นั้น จะมี program counter ซึ่งทำหน้าที่ชี้ตำแหน่งคำสั่ง(ถัดจากตำแหน่งคำสั่งที่กำลังทำงานใน)ปัจจุบันที่กำลังทำงานเพียงตัวเดียว โดยตัวนับดังกล่าวจะเปลี่ยนไปเรื่อยๆ เพื่อทำงานไปตามกระบวนการ (ตามแนวคิดของ Turing machine และ von Neumann architecture) แต่สำหรับงานที่เป็น multithread ก็จะมี program counter ประจำแต่ละเธรดแยกกัน

ดังนั้น ระบบคอมพิวเตอร์ในปัจจุบัน มักจะมีงานที่รันอยู่หลายงาน และอาจจะมีผู้ใช้ที่เป็นเจ้าของงานได้มากกว่าหนึ่งผู้ใช้ รวมทั้งบางระบบปฏิบัติการมีการกระจายงานของตัวระบบปฏิบัติการเองไปบนซีพียูที่มีมากกว่าหนึ่งหน่วยในระบบด้วย จึงเกิดสภาพการทำงานหลายงานไปพร้อมกัน (concurrency) โดยการจัดสรรเวลาการครอบครองซีพียูที่มีอยู่ (ไม่ว่าตัวเดียว หรือหลายตัว) ไปให้แก่งานแต่ละงานหรือเธรดแต่ละเธรด

หน้าที่ของระบบปฏิบัติการที่เกี่ยวเนื่องกับการบริหารจัดการงานในระบบ

- การสร้างและทำลายงาน
    - การสร้างงาน เริ่มต้นจองทรัพยากรที่จำเป็น กำหนดสภาวะเริ่มต้นให้กับตารางควบคุมงาน และพื้นที่หน่วยความจำที่ได้รับครอบครอง โหลดชุดคำสั่งมาจากหน่วยความจำสำรอง...
    - การทำลายงาน ปิดสตรีมทั้งหมดที่งานนั้นๆ ใช้อยู่ เรียกคืนพื้นที่หน่วยความจำที่งานนั้นจองเพิ่ม และเรียกคืนพื้นที่หน่วยความจำที่งานนั้นได้รับตอนเริ่ม และยกเลิกการใช้งานตารางควบคุมงาน และทรัพยากรอื่นๆ ที่งานนั้นใช้ลงทั้งหมด
- การหยุดการทำงานของงานนั้น และการสั่งให้งานนั้นทำงานต่อเนื่องจากที่ได้ค้างไว้
- จัดการกลไกการทำงานประสานกันระหว่างงาน (process synchronization)
- จัดการกลไกการสื่อสารกันระหว่างงาน (process communication)
- จัดการกลไกการแก้ไขสภาวะติดตายของงาน (deadlock handling)

1.3.2 การจัดการหน่วยความจำของระบบปฏิบัติการ (Memory Management)

หน้าที่โดยทั่วไปของระบบปฏิบัติการในการจัดการหน่วยความจำหลักมีดังนี้

- การจัดเตรียมพื้นที่หน่วยความจำและมอบให้กับงานต่างๆ เพื่อพร้อมที่จะนำเอาข้อมูลมาบรรจุ และการนำเอาพื้นที่หน่วยความจำที่เคยเก็บข้อมูลซึ่งไม่ได้ใช้แล้วของงานต่างๆ กลับคืนมาเพื่อจะได้นำไปจัดสรรให้กับงานอื่นที่ต้องการต่อไป
- การจัดเตรียมพื้นที่หน่วยความจำเพื่อที่จะเป็นที่วางตัวของชุดคำสั่งของงานต่างๆ และการรับคืนพื้นที่ดังกล่าวมาเมื่องานนั้นเสร็จสิ้นลง
- การจัดสรรการใช้งานพื้นที่หน่วยความจำของระบบโดยทั่วไป
- ในกรณีที่ระบบปฏิบัติการรองรับจำนวนงานที่ใช้พื้นที่หน่วยความจำเกินกว่าที่ระบบมีอยู่ ก็จะต้องมีกลไกการจัดสรรหน่วยความจำที่เน้นให้ส่งผลให้ประสิทธิภาพการใช้งานซีพียู (CPU utilization) สูงที่สุด (เช่น งานที่ไม่ได้ใช้บ่อยก็ swap ลงหน่วยความจำสำรอง นำงานที่ใช้บ่อยขึ้นมาในหน่วยความจำหลักแทนที่ เป็นต้น)
- การเก็บข้อมูลการใช้พื้นที่หน่วยความจำว่า งานใดเป็นเจ้าของพื้นที่หน่วยความจำในส่วนไหน
- การควบคุมกลไกการนำเอางาน (ทั้งชุดคำสั่งและข้อมูล) ในส่วนใดเข้าหรือออกจากหน่วยความจำหลัก
- กลไกการจัดสรรเรื่องการจองและคืนพื้นที่หน่วยความจำเพิ่มเติมเมื่องานใดต้องการ

1.3.3 การจัดการหน่วยความจำสำรองของระบบปฏิบัติการ (Storage Management)

หน้าที่หนึ่งของระบบปฏิบัติการที่สำคัญก็คือ การบริหารจัดการหน่วยความจำสำรองให้งานแต่ละงานที่รันอยู่นั้น สามารถเห็นโครงสร้างของหน่วยความจำสำรองในรูปแบบที่คงที่ชัดเจนเป็นมาตรฐานเดียว แม้ว่าหน่วยความจำสำรองอาจจะมีโครงสร้างการจัดเก็บทางกายภาพที่แตกต่างไปอย่างใดก็ตาม ความแตกต่างทางกายภาพของหน่วยความจำสำรองมีดังเช่น

- การนำเสนอหน่วยของพื้นที่เก็บข้อมูลที่ไม่ขึ้นกับสภาพทางกายภาพ เช่น ระบบไฟล์ ที่เป็นหน่วยข้อมูลเรียงต่อเนื่องกันในทางปฏิบัติทางกายภาพ ความต่อเนื่องนี้อาจจะถูกแปรเป็นกลไกอื่นที่ไม่ได้ต่อเนื่อง ซึ่งงานแต่ละงานจะไม่เห็นความแตกต่างทางกายภาพเหล่านี้
- การอ่านและเขียนข้อมูลในพื้นที่ทางกายภาพที่มีประสิทธิภาพแตกต่างกัน จะไม่ส่งผลต่อโครงสร้างการเข้าถึงข้อมูล (เว้นแต่อาจจะส่งผลเรื่องความเร็ว) เช่นความเร็วในการเข้าถึงข้อมูลในหน่วยแรก หรือหน่วยใดๆ นับจากนั้น ความจุของหน่วยเก็บข้อมูล ความเร็วในการส่งถ่ายข้อมูล วิธีการเข้าถึงข้อมูล(เช่นอาจเข้าถึงหน่วยที่ต้องการได้ทันที หรือต้องอ่านเป็นชุดแล้วดึงเอาหน่วยเดียวมาใช้) สิ่งเหล่านี้ระบบปฏิบัติการจะจัดการเองทั้งหมดโดยงานที่เรียกใช้ข้อมูลจะไม่ต้องมารับภาระต่างๆ ในจุดนี้

โครงสร้างโดยทั่วไปของระบบข้อมูลในหน่วยความจำสำรองที่ใช้กันในปัจจุบัน จะมองอยู่ในรูปของ "ไฟล์" (File) ซึ่งประกอบไปด้วยข้อมูลเรียงต่อเนื่องกัน ข้อมูลที่อยู่ภายในอาจจะเป็นตัวเลข ตัวอักษร หรือตัวอักษรผสมตัวเลขที่อ่านได้ด้วยรหัสในมาตรฐานใดมาตรฐานหนึ่ง (เช่นเป็น ASCII Unicode หรืออื่นใด เป็นต้น) ซึ่งเราเรียกว่าไฟล์ตัวอักษร Text file หรือข้อมูลที่อยู่ภายในอาจจะเป็นข้อมูลที่ไม่สามารถอ่านได้โดยมนุษย์ โดยเป็นการจัดเก็บตามวิธีการเฉพาะของงานหนึ่งๆ เราเรียกว่า Binary File โครงสร้างของไฟล์อาจจะไม่มีกฎกำหนดตายตัว เช่นแต่ละบรรทัดของข้อความในไฟล์อาจจะยาวได้ไม่เท่ากัน หรืออาจจะเป็นโครงสร้างที่มีกฎตายตัว เช่นเป็นอะเรย์ของแบบตัวแปรโครงสร้างที่มีขนาดชัดเจน เป็นต้น

เพื่อความสะดวกในการจัดกลุ่ม ระบบปฏิบัติการจะจัดกลุ่มของไฟล์ไปวางไว้เป็นกลุ่มด้วยกันเรียกว่า "ไดเรกตอรี" Directory หรือ โฟลเดอร์ Folder ซึ่งแต่ละไดเรกตอรีจะสามารถมีไฟล์ได้หลายตัววางอยู่ด้วยกัน และยังสามารถที่จะมีไดเรกตอรีย่อยภายใน ที่ทำหน้าที่เป็นกลุ่มย่อย จัดเก็บไฟล์ไว้ภายในได้อีก

ระบบปฏิบัติการมีหน้าที่จัดการเกี่ยวกับโครงสร้างของไฟล์ดังต่อไปนี้

- การบริหารจัดการโครงสร้างไฟล์และไดเรกตอรี
- การควบคุมว่าผู้ใช้ใด(ที่เป็นเจ้าของงานใด) สามารถเข้าถึงไฟล์และไดเรกตอรีใดได้บ้าง (access control)
- การสร้างและลบไฟล์ และไดเรกตอรี
- การเปลี่ยนแปลงโครงสร้างไฟล์และไดเรกตอรีในเบื้องต้น (เช่นการเปลี่ยนชื่อ การเปลี่ยนสิทธิ)
- ระบบการนำเสนอ (mapping) โครงสร้างไฟล์และไดเรกตอรี ไปสู่โครงสร้างทางกายภาพของหน่วยความจำสำรอง
- กลไกการแบ็คอัปไฟล์จากหน่วยความจำสำรองหนึ่งไปยังอีกหน่วยหนึ่ง (เช่นหน่วยความจำสำรองที่มีความทนทานในการเก็บข้อมูลสูงกว่า หรือมีความจุสูงกว่า เป็นต้น)

หน่วยความจำสำรอง มีหน้าที่ในการจัดเก็บข้อมูล (และชุดคำสั่ง) ที่ไม่สามารถจะจัดเก็บไว้ในหน่วยความจำของระบบได้ทั้งหมด หรือข้อมูล และชุดคำสั่งที่จะจัดเก็บนั้นไม่ได้ใช้งานบ่อย ๆ หรือต้องการจัดเก็บไว้เป็นเวลานานๆ รวมทั้งต้องสามารถรักษาข้อมูลและชุดคำสั่งไว้ได้แม้ไม่มีไฟเลี้ยงในระบบ สำหรับระบบปฏิบัติการใหม่ๆ การทำงานของงานหนึ่งๆ อาจจะต้องเรียกใช้ชุดคำสั่งและข้อมูลเพิ่มเติมจากหน่วยความจำสำรอง เพื่อจะทำงานย่อยๆ หนึ่งให้เสร็จก่อนที่จะทำงานอื่นๆ ต่อไป เช่น คอมไพเลอร์ เวลาคอมไพล์ จะต้องบันทึกไฟล์ต้นฉบับของเราลงหน่วยความจำสำรองก่อน จากนั้นจึงเรียกโปรแกรมที่ทำหน้าที่คอมไพล์ขึ้นมาคอมไพล์ ตามด้วยโปรแกรมที่ทำหน้าที่เชื่อมไฟล์ที่คอมไฟล์แล้วกับไลบรารีมาตรฐาน ดังนั้นจะเห็นว่าการจัดการหน่วยความจำสำรองที่มีประสิทธิภาพ จะช่วยเพิ่มประสิทธิภาพในการใช้งานของโปรแกรมต่างๆ ได้ กลไกการจัดการหน่วยความจำสำรอง และอัลกอริทึมที่ระบบปฏิบัติการใช้เพื่อจัดการหน่วยความจำสำรองจึงเป็นสิ่งสำคัญ

ระบบปฏิบัติการจัดการหน่วยความจำสำรองในประเด็นต่างๆ ดังนี้

- การจัดบริหารพื้นที่ว่างในหน่วยความจำสำรอง
- การเลือกพื้นที่ว่างที่มีอยู่ออกมาเพื่อนำมาใช้ในการจัดเก็บข้อมูล
- การจัดสรรการใช้งานพื้นที่ของงานต่างๆ กับหน่วยความจำสำรอง (disk scheduling)
- หน่วยความจำสำรองที่ระบบปฏิบัติการต้องเรียกใช้บ่อยๆ จำเป็นต้องมีกลไกในการบริหารจัดการเป็นพิเศษเพื่อเพิ่มประสิทธิภาพในการใช้งานให้สูงที่สุด
- หน่วยความจำสำรองที่ระบบปฏิบัติการไม่ค่อยได้เรียกใช้ หรือมีความเร็วในการเข้าถึงและการอ่านข้อมูลต่ำ (เช่นซีดีรอม) การจัดการก็ไม่จำเป็นต้องมีอัลกอริทึมพิเศษมากเท่า แต่เพื่อความสะดวกในการเข้าใช้งาน ระบบปฏิบัติการก็ยังคงต้องมีกลไกจัดการขึ้นพื้นฐานให้แก่หน่วยความจำสำรองเหล่านี้ด้วย
- กลไกอันหนึ่งที่ใช้ในการเพิ่มประสิทธิภาพในการเข้าถึงข้อมูลคือการแคชชิง Caching หรือการอ่านข้อมูลจากหน่วยความจำที่มีความเร็วในการเข้าถึงต่ำกว่า มาไว้ในพื้นที่หน่วยความจำที่มีความเร็วในการเข้าถึงสูงกว่า เป็นการล่วงหน้า หรือคงการเก็บข้อมูลไว้ในหน่วยความจำที่มีความเร็วสูงกว่าโดยไม่จำเป็นต้องเขียนลงหน่วยความจำที่มีความเร็วต่ำกว่าตลอดเวลา
    - ปัญหาของการมีระบบแคชก็คือ ข้อมูลจะมีปรากฏได้มากกว่าหนึ่งชุดในระบบคอมพิวเตอร์ ทำให้ต้องมีการบริหารจัดการที่ต้องทำให้แน่ใจว่า ข้อมูลที่จะใช้งานนั้น จะต้องเป็นข้อมูลล่าสุด โดยเฉพาะอย่างยิ่ง ข้อมูลในแคช จะต้องมีความล่าสุดเสมอ
    - สำหรับระบบที่มีซีพียูมากกว่าหนึ่งหน่วย แต่ละหน่วยข้อมูลอาจจะมีแคชแยกกันเป็นอิสระ ระบบปฏิบัติการและฮาร์ดแวร์ จะต้องสามารถทำให้ข้อมูลที่ซ้ำกันที่ให้กับซีพียูแต่ละตัวนั้น จะต้องเป็นข้อมูลชุดเดียวกัน
    - สำหรับระบบกระจาย (ที่มีคอมพิวเตอร์มากกว่าหนึ่งตัว เช่นคลัสเตอร์) ข้อมูลชุดเดียวกันที่ถูกเรียกใช้จากต่างหน่วยคอมพิวเตอร์ก็จะต้องถูกบริหารจัดการให้เหมือนกันด้วย

1.3.4 การบริหารจัดการหน่วยรับส่งข้อมูล (I/O Systems)

ระบบปฏิบัติการ มีหน้าที่ที่จะต้องซ่อนความซับซ้อนของหน่วยรับส่งข้อมูลที่มีลักษณะแตกต่างกันไป ให้เป็นมาตรฐานเดียวหรือน้อยมาตรฐานที่สุด ในมุมมองของงานที่กำลังทำงานอยู่บนระบบ (พยายามทำให้กลไกการติดต่อดีไวซ์ที่แตกต่างกันสามารถใช้กลไกการติดต่อแบบเดียวกันได้ยิ่งดี)

ระบบปฏิบัติการมีหน้าที่บริหาร I/O ดังเช่น

- การจัดสรรพื้นที่หน่วยความจำที่ไว้เก็บข้อมูลที่ส่งถ่ายโอนระหว่างดีไวซ์กับงานใดๆ ซึ่งรวมถึง
  - Buffering (การจัดเก็บข้อมูลมาพักไว้ก่อนที่จะส่งต่อ ไม่ว่าจากดีไวซ์ไปยังงาน หรือจากงานไปยังดีไวซ์)
  - Caching (การนำเอาข้อมูลมาเก็บไว้ในพื้นที่หน่วยความจำที่สามารถเข้าถึงได้รวดเร็วกว่า เพื่อที่จะสามารถอ่านได้อย่างเร็ว หรือเขียนได้อย่างเร็ว เช่นดิสก์แคช ที่ทำหน้าที่อ่านข้อมูลจากดิสก์ล่วงหน้ามาเก็บไว้ในหน่วยความจำหลัก เพื่อรอให้งานเข้ามาอ่าน หรือการเก็บพักข้อมูลที่งานเขียนลงมาไว้ แล้วค่อยเขียนลงดิสก์ในภายหลัง เป็นการลดภาระการรอคอยของงาน)
  - Spooling (การพักข้อมูลจากแหล่งกำเนิดข้อมูล ซึ่งมักจะเป็นงานใดๆ เพื่อส่งทอดต่อให้กับดีไวซ์ที่อ่านข้อมูลเข้าไปทำงานต่อได้ช้ากว่า เช่น printer spooling ที่ทำหน้าที่รับข้อมูลจากซีพียูเพื่อนำส่งไปยังดีไวซ์ภายนอกที่ทำงานช้ากว่า และในขณะที่ส่งข้อมูลออกก็สามารถรับข้อมูลใหม่เข้ามาในคิวได้ตลอดเวลา
- การให้บริการดีไวซ์ไดรเวอร์สำหรับอุปกรณ์ที่เป็นมาตรฐาน (เช่น HID คีย์บอร์ด เมาส์ พอร์ตอนุกรม พอร์ตขนาน)
- การให้บริการดีไวซ์ไดรเวอร์สำหรับอุปกรณ์เฉพาะ (เช่นสแกนเนอร์)

1.3.5 การบริหารจัดการด้านความปลอดภัย (Protection and Security)

ระบบปฏิบัติการยังมีหน้าที่เกี่ยวกับการรักษาความปลอดภัยของระบบในสองด้าน

- การรักษาความปลอดภัยจากงานต่างๆ ภายในระบบปฏิบัติการเอง (งานแต่ละงานจะสามารถใช้ทรัพยากรของระบบเท่าที่ได้รับอนุญาต)
- การรักษาความปลอดภัยจากระบบภายนอก หรือจากงานที่มีจุดประสงค์ร้ายต่อระบบ (เช่นการโจมตีผ่านระบบเครือข่าย หรือจากซอฟต์แวร์เช่นไวรัส หรือโทรจันที่รันอยู่ในระบบเอง)

ระบบปฏิบัติการโดยทั่วไปจะมีกลไกสำหรับการจัดการรักษาความปลอดภัยโดยใช้กลไกต่อไปนี้

- มีการลงทะเบียนผู้ใช้เป็นกิจลักษณะ (user identities หรือ user id หรือ security id) ซึ่งจะประกอบไปด้วยชื่อผู้ใช้ และรหัสแทนชื่อผู้ใช้ เฉพาะบุคคลนั้นๆ
- ไฟล์ และงานภายในระบบ จะถูกกำกับไว้ด้วยรหัสแทนชื่อผู้ใช้ เพื่อที่ระบบปฏิบัติการจะสามารถพิจารณาว่างานและไฟล์แต่ละตัวจะมีสิทธิในการเข้าถึงและทำงานได้อย่างไร
- เพื่อความสะดวกในไฟล์ที่มักใช้ร่วมกันหลายผู้ใช้ ก็จะมีการกำหนดรหัสกลุ่ม (group id) เพื่อใช้ในการควบคุมการเข้าถึงและใช้งานอีกทางหนึ่ง โดยรหัสนี้จะถูกใช้ควบคู่ไปกับรหัสผู้ใช้ในแต่ละไฟล์และงานต่างๆ
- ระบบปฏิบัติการยังมีกลไกการยกระดับการเข้าถึงไฟล์หรืองานเป็นกรณีพิเศษ (privilage escalation) เพื่อให้งานของผู้ใช้สามารถเข้าถึงการทำงานเฉพาะอย่างที่สูงเกินกว่าการกำหนดตามปกติได้

1.3.6 การบริหารจัดการด้านอื่นๆ ของระบบปฏิบัติการ

การคำนวณแบบกระจาย Distributed Computing

สำหรับระบบคอมพิวเตอร์แบบกระจาย (distributed computing) ซึ่งจะมีคอมพิวเตอร์หลายหน่วยเชื่อมต่อกันเป็นระบบใหญ่ การเชื่อมต่ออาจจะเป็นลักษณะใช้ LAN – Local Area Network WAN- Wide Area Network (เช่นการเชื่อมต่อแบบ ADSL ระหว่างอาคาร) Metropolitan Area Network (MAN) (เชื่อมต่อในระดับเมือง) หรือกว่านั้น (ซึ่งยังรวมไปถึงพวก Cloud Computing ด้วย)

ในระบบแบบกระจาย ก็จะมีระบบปฏิบัติการเครือข่าย (Network Operating System) ที่จะทำหน้าที่บริหารจัดการทรัพยากรที่ใช้ร่วมกันภายในระบบเครือข่าย ซึ่งอาจจะประกอบไปด้วย

- การติดต่อกันระหว่างงานของคอมพิวเตอร์ภายในกลุ่ม (communication)
- การแชร์ไฟล์ระหว่างกัน
- การแชร์ดีไวซ์ระหว่างกัน (เช่นเครื่องพิมพ์)
- การแชร์ระบบการเข้าถึงระบบ (authentication)

ลักษณะเฉพาะของระบบเฉพาะด้าน (Special-Purpose Systems) และการออกแบบระบบปฏิบัติการที่สนับสนุนการทำงานในลักษณะดังกล่าว

คอมพิวเตอร์ที่ใช้อย่างแพร่หลายกันในปัจจุบันมักจะเป็นคอมพิวเตอร์ส่วนบุคคล (personal computer) ที่ออกแบบให้ทำงานในลักษณะทั่วไป โดยจะมีจอภาพ คีย์บอร์ด (และเมาส์) เป็นอุปกรณ์หลักในการเชื่อมต่อ (และอาจจะมีอุปกรณ์อื่นเช่น ไมโครโฟน ลำโพง กล้อง และอื่นๆ) และมีอุปกรณ์เชื่อมต่อกับระบบเครือข่าย (การ์ดเครือข่ายที่ใช้สายเชื่อมต่อ หรือแบบไร้สาย)

- คอมพิวเตอร์เหล่านี้อาจจะมีลักษณะใหญ่พอสมควรที่เรียกว่าเป็นคอมพิวเตอร์ตั้งโต๊ะ (desktop computer) หรือเล็กลงในขนาดหนังสือหรือวางตักได้ (notebook/laptop computer) ซึ่งในแบบหลังถูกออกแบบให้สามารถใช้ไฟเลี้ยงหลักจากไฟบ้านหรือแบตเตอรี่ได้
- ด้วยการพัฒนาลดขนาดของอุปกรณ์ต่างๆ ในปัจจุบันจึงมีคอมพิวเตอร์ขนาดเล็กลงที่สามารถใช้ถือได้สะดวก และใช้ปากกาหรือนิ้วสัมผัสควบคุม เรียกว่า แท็ปเล็ต Tablet และโทรศัพท์ที่รวมเอาขีดความสามารถของคอมพิวเตอร์เข้าด้วยกัน และใช้หน้าจอสัมผัสเป็นหลัก (บางเจ้าอาจจะคงปุ่มไว้ให้ใช้งาน แต่แทบจะไม่ได้พบเห็นแล้วในปัจจุบัน) ซึ่งเรียกว่าสมาร์ทโฟน
- ระบบปฏิบัติการสำหรับ desktop/labtop computer ถูกออกแบบมาให้ทำงานโดยทั่วไปเป็นหลัก โดยอาจจะไม่ได้คำนึงถึงการประหยัดพลังงานมากนัก (สำหรับ labtop ก็จะมีโหมดประหยัดพลังงานเพิ่มเติมเข้ามาด้วย)
- ระบบปฏิบัติการสำหรับ Tablet และ สมาร์ทโฟน จะถูกออกแบบมาเพื่อให้ประหยัดพลังงานเป็นหลัก และรองรับซีพียูที่ออกแบบมาเฉพาะ (อย่างกรณี big.LITTLE) ตัวอย่างระบบปฏิบัติการที่นิยมใช้ในสมาร์ทโฟนปัจจุบันอย่างเช่น iOS (Apple) และ Android (Google)

ยังมีระบบคอมพิวเตอร์ที่มีหน้าที่เฉพาะด้าน อาทิเช่น

- ระบบเวลาจริง(หรือระบบทันเวลา) (Real-time systems) ซึ่งงานที่รันบนระบบ จะต้องสามารถประมวลข้อมูลและส่งผลลัพธ์ให้ได้ภายในเวลาที่กำหนด ซึ่งระบบปฏิบัติการจะต้องออกแบบมาเพื่อประกันให้ได้ว่างานแต่ละงานที่รันบนระบบจะต้องให้ผลลัพธ์ภายในเวลาที่กำหนด(หลังจากได้รับข้อมูลเข้า) ระบบพวกนี้อาจจะเป็นระบบฝังตัว เช่นระบบคอมพิวเตอร์ที่ใช้ในการคำนวณจังหวะจุดระเบิด การปิด-เปิดวาวล์ไอดีและไอเสียของรถยนต์ ที่จะเกิดการทำงานไม่ทันไม่ได้โดยเด็ดขาด หรือระบบในดาวเทียม เป็นต้น ระบบปฏิบัติการที่ถูกออกแบบมาเฉพาะเพื่องานนี้เรียกว่า Real-time OS

- ระบบมัลติมีเดีย (Multimedia Systems) ระบบแบบนี้จะมีข้อมูลภาพ และข้อมูลเสียง ที่ต้องรับประกันว่าข้อมูลภาพและเสียงสามารถส่งไปแสดงผลได้ทันตามมาตรฐานของการแสดงภาพและเสียงนั้นๆ
- ระบบคอมพิวเตอร์มือถือขนาดเล็ก (Handheld Systems) เช่น PDA smartphone ระบบพวกนี้มีข้อจำกัดที่ขนาดแบตเตอร์รีที่เล็ก ทำให้ต้องควบคุมพลังงานที่ใช้ ส่งผลทำให้การออกแบบซีพียู หน่วยความจำหลัก หน่วยความจำสำรองและส่วนรับส่งข้อมูล มีประสิทธิภาพได้จำกัด การออกแบบระบบปฏิบัติการจึงต้องทอนฟังก์ชันที่ไม่จำเป็นออกให้มากที่สุด และต้องรีดประสิทธิภาพของฮาร์ดแวร์ให้ได้มากที่สุด

1.3.7 ระบบปฏิบัติการที่เปิดเผยโค้ดต้นฉบับ และระบบปฏิบัติการฟรี

- ระบบปฏิบัติการที่ถูกพัฒนาขึ้นในช่วงแรกๆ และหลายระบบปฏิบัติการในปัจจุบัน จะอยู่ในรูปของการไม่เปิดเผยโค้ดต้นฉบับให้ผู้ใช้ (closed-source) และเป็นซอฟต์แวร์ที่มีเจ้าของเฉพาะไม่ให้ใช้ได้ฟรี (proprietary software)
    - ซอฟต์แวร์โดยทั่วไปที่มีลักษณะต้องจ่ายเงินเพื่อขออนุญาตได้สิทธิใช้งานเหล่านี้ มักจะมีกลไกการป้องกันการทำสำเนา (copy protection) และมีกลไกการป้องกันสิทธิทางดิจิทัล (Digital Right Management- DRM) อยู่ภายใน
    - กลไกการป้องกันอาจอยู่ในลักษณะการของการใช้โค้ดที่อาจไม่ผูก หรือผูกกับฮาร์ดแวร์ของเครื่อง (เช่น ระบบปฏิบัติการอาจจะผูกกับ CPU ID ซึ่งถ้าเปลี่ยนซีพียูก็จะต้องใส่โค้ดใหม่) การบันทึกข้อมูลลับลงในหน่วยความจำสำรอง การเข้ารหัสซอฟต์แวร์ ไปจนถึงการใช้ฮาร์ดแวร์ (dongle) เสียบเข้ากับพอร์ตของคอมพิวเตอร์ เป็นต้น
- ในเวลาต่อมา จึงมีกลุ่มเคลื่อนไหวผลักดันซอฟต์แวร์ที่ต้องเปิดเผยให้แก้ไขโค้ด และใช้งานได้ฟรี Free Software Foundation (FSF) เกิดคำศัพท์ Copyleft (ตรงข้ามกับ Copyright) ซึ่งหมายถึงสำเนาเอาไปใช้ได้เต็มที่ และต่อมา GNO Public License (GPL)
    - GPL มีหลายตัวเลือก แต่โดยทั่วไปแล้ว จะเปิดเผยโค้ดต้นฉบับ และผู้ใช้สามารถแก้ไขและนำไปใช้ได้เต็มที่ (มักจะต้องส่งโค้ดที่แก้ไขแล้ว กลับมายังชุมชนเพื่อแบ่งปันพัฒนากันต่อไป)
    - ตัวอย่างระบบปฏิบัติการที่มีลักษณะนี้ได้แก่ GNU/Linux และ BSD UNIX (ซึ่งรวมถึงคอร์ของ OS X ด้วย)
- ตัวอย่างของ Virutal Machine ที่ใช้งานได้ฟรีอย่างเช่น VirtualBox

1.4 สภาพแวดล้อมและลักษณะการใช้งานระบบคอมพิวเตอร์

ระบบคอมพิวเตอร์ในสมัยยุคเริ่มแรก เนื่องจากมีขนาดใหญ่ กินไฟมาก และมีราคาสูงมาก จึงถูกออกแบบมาเพื่องานเฉพาะด้าน เช่นงานด้านทหาร (การคำนวณวิถีกระสุนปืนใหญ่ เป็นต้น) ต่อมาก็ถูกนำมาใช้ในการคำนวณทางวิศวกรรม หรืองานทางด้านการสร้างภาพ (เรนเดอร์ภาพเพื่อใช้ในงานด้านการวิจัย หรือต่อมาทำภาพยนต์)

การเข้าถึงคอมพิวเตอร์แบบนี้ เริ่มแรกจะเป็น data entry ที่ออกแบบมาวางอยู่ในบริเวณของคอมพิวเตอร์ (คอมพิวเตอร์อาจจะอยู่ในห้องใหญ่ห้องหนึ่ง-ขนาดเต็มพื้นที่) และจะมีห้อง data entry อยู่ข้างๆ เพื่อให้ผู้ใช้เข้าถึงคีย์บอร์ด จอภาพ และเครื่องบันทึกข้อมูลหรืออ่านข้อมูล ต่อมาเริ่มมีการใช้ระบบเครือข่ายเช่น โมเด็ม เพื่อให้ผู้ใช้จากบ้านสามารถเชื่อมต่อด้วยเทอร์มินัลเข้ามายังเครื่องหลักได้

ในยุคปัจจุบัน คอมพิวเตอร์มีขนาดเล็กลงอย่างมาก และมีราคาถูกลง ทำให้การใช้งานเปลี่ยนแปลงไปจากแต่ก่อน คอมพิวเตอร์ถูกนำมาใช้ในแทบจะทุกด้าน ในเครื่องใช้ไฟฟ้า ยานพาหนะ อุปกรณ์เพื่อความพักผ่อนหย่อนใจ และการคำนวณต่างๆ ดังเช่น

- ในสำนักงาน เราอาจจะเห็นคอมพิวเตอร์หลายเครื่องต่อเข้าด้วยกันผ่านเครือข่ายท้องถิ่น หรือในสำนักงานที่ต้องใช้การคำนวณหรือจัดเก็บข้อมูลจำนวนมาก ก็อาจจะมีเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเทอร์มินัล (อุปกรณ์ที่ทำหน้าที่รับ

ข้อมูลจากผู้ใช้และแสดงข้อมูลให้ผู้ใช้) เชื่อมต่อเข้ากับเมนเฟรม(เดี๋ยวนี้หาดูยากแล้ว) หรือมินิคอมพิวเตอร์ (ยังพอเห็นได้อยู่ในสำนักงานขนาดใหญ่ เช่นธนาคาร หรือมหาวิทยาลัย—เช่นเครื่อง queen ในม.ของเราก็ใช่ด้วย) สำหรับเมนเฟรมนั้นมักจะเป็นแบบ batch (ใช้ job scheduling) หรือเป็น time sharing (แบบเครื่อง queen ของเรา) นอกจากนี้เราจะเห็นมีการเชื่อมต่อดีไวซ์ส่วนกลาง เช่นเครื่องพิมพ์ เข้ากับระบบเครือข่ายเพื่อใช้ร่วมกันในสำนักงาน รวมทั้งระบบไฟล์เซอร์เวอร์ ที่แชร์พื้นที่เก็บข้อมูลร่วมกัน เป็นต้น และมักจะมีไฟล์วอลล์ ที่เป็นอุปกรณ์เฉพาะทำหน้าที่ป้องกันการโจมตีจากภายนอก

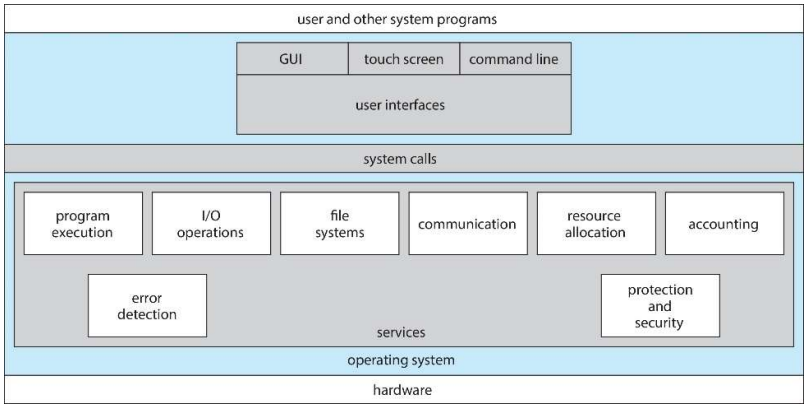
- ในบ้าน เรามักจะเห็นคอมพิวเตอร์เครื่องเดียว หรือสองสามเครื่อง เชื่อมต่อผ่านเครือข่ายท้องถิ่นวงเดียว และเชื่อมผ่านโมเด็ม (ปัจจุบันเรามักเห็นการใช้ cable modem หรือ ADSL model เสียมากกว่า) ส่วนไฟล์วอลล์มักจะเป็นฟีเจอร์ที่มีอยู่ในโมเด็ม หรือติดตั้งเพิ่มเติมเฉพาะคอมพิวเตอร์แต่ละเครื่อง

โครงสร้างของระบบคอมพิวเตอร์ที่ประกอบไปด้วยคอมพิวเตอร์มากกว่าหนึ่งเครื่อง อาจจะจัดอยู่ในรูปใดรูปหนึ่งดังนี้คือ

- **Client-Server Computing** ระบบแบบนี้จะประกอบไปด้วยคอมพิวเตอร์จำนวนมากที่เป็น Client เป็นผู้รับบริการที่ Server ทำหน้าที่ให้บริการ การบริการอาจจะเป็นดังเช่น การให้บริการพื้นที่เก็บข้อมูล (File Server) การให้บริการเครื่องพิมพ์ (Printer Server) การให้บริการจดหมายอิเล็กทรอนิคส์ (Mail Server) เป็นต้น
- **Peer-to-Peer Computing (P2P computing)** เป็นการให้บริการในลักษณะแลกเปลี่ยนข้อมูลกันระหว่างกันของเครื่องสองเครื่อง (โดยเครื่องหนึ่งๆ จะสามารถมีเครื่องคู่ที่แลกเปลี่ยนข้อมูลได้มากกว่าหนึ่งคู่ ทำให้เกิดเป็นโครงข่ายที่ซับซ้อนได้) แต่ละเครื่องที่แลกเปลี่ยนข้อมูลเรียกว่า Peer ดังนั้นจึงไม่สามารถกำหนดว่าใครเป็นเซอร์เวอร์หรือไคลเอนต์ได้แน่ชัด (แม้ว่าการให้บริการแบบ P2P นี้บางที peer หนึ่งอาจจะเป็นเซอร์เวอร์ อีกตัวหนึ่งอาจจะเป็นไคลเอนต์ก็ได้ แต่ในเวลาอื่นก็อาจจะสลับหน้าที่นี้กันได้ หรืออาจจะเป็นทั้ง Server และ Client ในเวลาเดียวกันก็ได้ เพราะส่งข้อมูลที่ตนมีให้กับอีกฝ่ายที่ไม่มี เป็นต้น) กลไกที่จะทำให้ระบบ P2P นี้ทำงานได้จะประกอบไปด้วยคอมพิวเตอร์กลางที่จะทำหน้าที่เก็บฐานข้อมูลเริ่มต้นว่ามี peer ในอยู่ในระบบบ้าง และ peer แต่ละตัวนั้นก็จะส่งข้อมูลไปยังทั้งโหนดกลาง และ/หรือ โหนดอื่นที่ทราบ หรืออาจจะ broadcast ข้อมูลภายใน Local net ของตนเองก็ได้ และยังสามารถที่จะแลกเปลี่ยนรายการ peer ที่ตนมีให้แก่กันและกัน เพื่อเพิ่มจำนวน peer ที่จะติดต่อได้ให้มากขึ้นเรื่อยๆ เมื่อเวลาผ่านไป bittorrent ถือเป็นการบริการแบบ P2P ที่ใช้กันอยู่ในปัจจุบัน
- **Web-Based Computing** เป็นการให้บริการโดยใช้โปรโตคอลพื้นฐานคือ http โดยในระบบจะมี Web Server ทำหน้าที่เป็นผู้ให้บริการทั้งเอกสารพื้นฐาน และฟังก์ชันที่ใช้ในการคำนวณ (เราเรียกว่า Web Service) ส่วนเครื่องไคลเอนต์ก็จะรันเว็ปเบราว์เซอร์ ซึ่งจะเชื่อมต่อกับเว็ปเซอร์เวอร์ด้วยโปรโตคอล http และสามารถเรียกใช้บริการโดยการคำนวณบางส่วนอาจจะกระทำอยู่บนเครื่องไคลเอนต์เอง (อาจจะด้วยจาวาสคริปต์ แฟลช ซิลเวอร์ไลท์ หรือจาวา หรืออื่นใดแล้วแต่) บางส่วนอาจจะกระทำอยู่บนเว็ปเซอร์เวอร์ ข้อมูลที่เก็บก็อาจจะอยู่บนเซอร์เวอร์ และบางส่วนก็อาจจะนำมาเก็บไว้ในไคลเอนต์ด้วยก็ได้ เนื่องจากในระบบนี้ จำนวนไคลเอนต์อาจจะมีอยู่มาก ทำให้เซอร์เวอร์เพียงตัวเดียวไม่พอรองรับการทำงาน ก็อาจจะมีเซอร์เวอร์ให้บริการจำนวนมาก โดยแบ่งการให้บริการในเซอร์เวอร์แต่ละตัว กระจายให้แก่ผู้ใช้คนใดคนหนึ่งในขณะใดขณะหนึ่ง เราเรียกกลไกนี้ว่า Load balancer
- **Cloud Computing** ซึ่งมีลักษณะการจำลองระบบคอมพิวเตอร์ลงบนเครื่องให้บริการในลักษณะที่ผู้ใช้ไม่จำเป็นต้องทราบถึงรายละเอียดทางฮาร์ดแวร์ (อย่างเช่นสเป็คของเซอร์เวอร์ และไอพีของคอมพิวเตอร์ เป็นต้น) สามารถเพิ่มประสิทธิภาพหรือลดประสิทธิภาพได้ตามที่ผู้ใช้ต้องการ (เนื่องจากเป็นการจำลองระบบลงบนฮาร์ดแวร์ ซึ่งส่วนมากจะใช้หลักการของ Virtualization) ซึ่งมีข้อดีคือการย้ายหรือปรับขยายประสิทธิภาพเซอร์เวอร์ทำได้สะดวก และไม่สะดุดต่อการให้บริการ ส่วนทางฝั่งผู้ให้บริการก็สามารถรวมเซอร์เวอร์เสมือนเหล่านี้เข้าด้วยกันจำนวนมาก (consolidation) เพื่อให้เกิดการใช้งานฮาร์ดแวร์ได้อย่างเต็มที่ และสามารถเพิ่มหรือลดฮาร์ดแวร์ตามความต้องการ ณ ขณะนั้นๆ ได้โดยสะดวก

1.4.1 หน้าที่พื้นฐานและโครงสร้างการติดต่อของผู้ใช้ ของระบบปฏิบัติการ

หน้าที่หลักของระบบปฏิบัติการ คือเป็นโครงสร้างรองรับพื้นฐาน ที่จัดสรรทรัพยากรต่างๆ ให้เหมาะสมและเป็นมาตรฐานเพื่อให้งานแต่ละงานสามารถทำงานได้โดยไม่จำเป็นต้องติดต่อจัดการฮาร์ดแวร์ส่วนต่างๆ ของคอมพิวเตอร์ด้วยตนเองโดยตรง



ระบบปฏิบัติการให้บริการและอำนวยความสะดวกแก่ผู้ใช้ ในประเด็นต่างๆ ดังต่อไปนี้

- ส่วนติดต่อกับผู้ใช้ขั้นพื้นฐาน (User Interface) ซึ่งทำหน้าที่รับคำสั่งจากผู้ใช้ผ่านทางเทอร์มินัล นำไปประมวลหรือรับการสั่งการให้ประมวลโปรแกรมและข้อมูลชุดหนึ่งๆ และส่งผลการทำงานกลับคืนให้กับผู้ใช้ ส่วนติดต่อพื้นฐานที่สุดมักจะเป็นคีย์บอร์ด และพื้นที่แสดงผลที่เป็นตัวอักษรเรียงกันเป็นแถวและบรรทัด **(command-line CLI)** โดยผู้ใช้สั่งการด้วยการพิมพ์ข้อความไปหนึ่งชุด แล้วจากนั้นระบบปฏิบัติการตอบสนองต่อคำสั่งนั้นอย่างใดอย่างหนึ่ง หรือการแสดงผลในรูปที่เป็นภาพ (Graphics User Interface GUI) โดยผู้ใช้สั่งการด้วยการเลือกองค์ประกอบในภาพ (เช่นคลิกที่ไอคอน) แล้วจากนั้นระบบปฏิบัติการตอบสนองต่อการเลือกนั้นๆ เป็นต้น นอกจากนี้การสั่งการยังรวมถึงการสร้างชุดคำสั่งเรียงไว้เป็นชุดเพื่อสั่งการให้คอมพิวเตอร์ทำงานต่างๆ เป็นลำดับขั้น (Batch)
- การโหลดโปรแกรมขึ้นมาทำงาน (Program execution) ซึ่งรวมไปถึงการสั่งจบการทำงาน ไม่ว่าการจบการทำงานแบบปกติหรือการสั่งจบแบบไม่ปกติ (abnormal program termination)
- การจัดการดีไวซ์ต่างๆ (I/O operations) การกำหนดสภาพเบื้องต้น การรับคำสั่งจากงานมากระทำต่อดีไวซ์ การส่งผลการทำงานและข้อมูลคืนกลับให้กับงานต่างๆที่ร้องขอ
- การจัดการระบบไฟล์ (File system manipulation) รวมไปถึงการเขียนหรืออ่านไฟล์และไดเรกตอรี การสร้าง การลบ การค้นหา การแสดงรายการไฟล์ การจัดการสิทธิในการเข้าถึง เป็นต้น
- การจัดการสื่อสารกับภายนอก (Communications) การเชื่อมต่อกับคอมพิวเตอร์อื่นในระบบเครือข่าย
- การดักจับความผิดพลาด (Error Detection) ซึ่งอาจจะเป็นความผิดพลาดจากการคำนวณ ความผิดพลาดจากการทำงานที่ผิดพลาดของดีไวซ์ หรือการเสียหายของดีไวซ์ ความผิดพลาดที่เกิดจากชุดคำสั่งในโปรแกรม เป็นต้น ซึ่งระบบปฏิบัติการมีหน้าที่ที่จะตรวจจับ และแจ้งแก่ผู้ใช้ให้ทราบ และเลือกจะปฏิบัติต่อความผิดพลาดเหล่านั้น (และรวมถึงการหยุดการทำงานของโปรแกรมถ้าจำเป็น) ระบบปฏิบัติการยังอาจจะมีกลไกการทดลองรันงานไปทีละขั้น (debugging) เพื่อให้ผู้ใช้สามารถดูผลการทำงานไปเป็นขั้นตอน และสามารถตรวจหาตำแหน่งที่ผิดพลาดได้โดยสะดวก
- การจัดสรรทรัพยากรในระบบ (resource allocation) ซึ่งจำเป็นในระบบปฏิบัติการที่รองรับงานมากกว่าหนึ่งงาน โดยทรัพยากรที่กล่าวรวมถึง เวลาที่ครอบครองซีพียู พื้นที่ในหน่วยความจำหลัก ไฟล์ในหน่วยความจำสำรอง ดีไวซ์ต่างๆ เป็นต้น

- การจัดสรรควบคุมทรัพยากรที่ผู้ใช้คนหนึ่งๆ จะสามารถใช้ได้ในระบบ (accounting) ซึ่งจะกำหนดว่าผู้ใช้คนใดจะมีสิทธิในการใช้ทรัพยากรใดได้บ้าง และมีโควต้าหรือเกณฑ์ขนาดไหน (เช่นใช้พื้นที่หน่วยความจำสำรองได้เท่าใด ใช้ดีไวซ์ใดได้บ้าง เป็นต้น)
- การป้องกันรักษาความปลอดภัย (Protection and security) โดยต้องทำหน้าที่ตรวจจับไม่ให้งานภายในระบบใดเข้าถึงทรัพยากรในระบบนอกเหนือจากที่กำหนด ตรวจจับไม่ให้ผู้ใช้เข้าถึงทรัพยากรที่มิได้รับอนุญาต และยังป้องกันไม่ให้มีการโจมตีจากภายนอก หรืองานที่มีจุดประสงค์ร้ายในระบบ ส่วนการจัดการการป้องกันและรักษาความปลอดภัยจะต้องถูกออกแบบมาให้สามารถรองรับการละเมิดในประเด็นต่างๆ อย่างล่วงหน้า (ต้องคาดการณ์ว่าอาจจะเกิดจุดอ่อนที่ใดได้บ้าง และต้องออกแบบป้องกันจุดอ่อนดังกล่าวไว้ทันทีโดยไม่ต้องรอให้เกิดการโจมตีที่จุดอ่อนก่อน) การออกแบบการป้องกันที่ดีก็จะทำให้ระบบมีความปลอดภัย แต่การออกแบบป้องกันที่ไม่ดี ก็จะเป็นจุดที่ทำให้ผู้โจมตีใช้เป็นทางเข้าเพื่อทำลายระบบได้โดยง่าย

<u>การติดต่อของผู้ใช้กับระบบปฏิบัติการ</u>

Command Line Interface (CLI) หรืออาจเรียกว่า Command Line Interpreter ทำหน้าที่รองรับการสั่งการของผู้ใช้แก่ระบบปฏิบัติการโดยตรง

- บางระบบจะใส่องค์ประกอบนี้ไว้ในเคอร์เนิลเลย ส่วนบางระบบจะใช้เป็น system program แทน
- เรามักเรียกส่วนการทำงานนี้ว่า shell (เหมือนกับเปลือกหอยที่หุ้มเนื้อหอยอยู่ เป็นส่วนที่ผู้ใช้สามารถสัมผัส) ในลินุกซ์เราจะเห็น shell มีอยู่หลายตัวให้เลือก เช่น sh ksh bash เป็นต้น ส่วนในดอสคือ command.com
- เชลล์จะรอรับข้อความจากผู้ใช้โดยการแสดงสัญญลักษณ์เตรียมรอรับคำสั่ง shell prompt
- คำสั่งที่ได้รับนั้น บางส่วนจะมีชุดคำสั่งรองรับการทำงานเขียนไว้เป็นส่วนหนึ่งของเชลล์เลย (เรียกว่า internal command) แต่ถ้าคำสั่งที่ได้รับมาไม่มีคำสั่งรองรับภายใน เชลล์จะค้นหาไฟล์ที่มีชื่อสอดคล้องกันเพื่อเรียกขึ้นมาทำงานแทน (external command)

Graphic User Interface (GUI) เป็นวิธีการติดต่อกับผู้ใช้ของระบบปฏิบัติการในยุคหลังๆ ที่มีทรัพยากรมากขึ้น และส่วนการแสดงผลสามารถแสดงผลในลักษณะภาพได้ (แทนของเดิมที่เป็นแถวตัวอักษร) โดยหลักการพื้นฐานของ GUI มักจะออกแบบอยู่บนแนวคิดเลียนแบบโต๊ะทำงานของผู้ใช้ (Desktop metaphor) ที่มีข้าวของวางอยู่บนโต๊ะ (Desktop) และสามารถหยิบจับเลือกใช้งานตามสะดวก (ยังมีแนวทางอื่น เช่น มองเป็นห้องหรือพื้นที่สามมิติที่ผู้ใช้สามารถเดินเข้าไปในพื้นที่เพื่อหยิบของที่วางไว้ แขวน หรือห้อยลงมา เพื่อนำไปใช้ได้ เป็นต้น)

- อุปกรณ์พื้นฐานที่ใช้กับ GUI ก็จะเป็นคีย์บอร์ด (สำหรับใช้พิมพ์ข้อความและสั่งการผสมผสานกับเมาส์) เมาส์ (สำหรับใช้แทนมือของผู้ใช้ที่จะเคลื่อนไปมาบนเดสก์ท็อป และจอภาพที่แสดงโหมดกราฟิก (เป็นสองมิติหรือสามมิติขึ้นอยู่กับแนวคิดของการนำเสนอ)
- ไอคอน (icon) ซึ่งเปรียบเหมือนของที่วางบนโต๊ะเพื่อให้หยิบจับได้ นำเสนอไฟล์ข้อมูล โปรแกรม หรือการสั่งการอย่างหนึ่งอย่างใด เป็นต้น
- การควบคุมสั่งการหลักกระทำผ่านทางเมาส์ โดยเมาส์ก็จะมีปุ่ม หรือ scroll ที่สามารถใช้ผสมผสานกันกับปุ่มบนคีย์บอร์ดเกิดการสั่งการที่ซับซ้อนขึ้นได้มากมาย (เช่นการเลือกเมนูป๊อปอัป การสั่งการ ยกเลิก รันโปรแกรม เปิดโฟลเดอร์-ไดเรกตอรีในมุมมองของ GUI ฯลฯ)
- ระบบ GUI นี้ถูกคิดเริ่มแรกมาจาก Xerox PARC

ระบบปฏิบัติการในปัจจุบันโดยส่วนมากจะมีทั้ง GUI และ CLI ไว้ให้เลือกใช้ โดยการใช้งานตามปกติจะเป็น GUI แต่ก็จะมีโปรแกรมระบบที่มีลักษณะเป็น CLI ไว้ให้ใช้งานด้วย

- วินโดวส์มี Explorer ทำหน้าที่เป็น GUI ในขณะที่มี CMD ทำหน้าที่เป็น CLI
- Mac OS X มี Aqua เป็น GUI และยังคงมี Unix shell ทำหน้าที่เป็น CLI (Mac OS X พัฒนาต่อยอดมาจาก Darwin ซึ่งเป็นยูนิกซ์ดิสตริบิวชันหนึ่ง)
- ลินุกซ์มี KDE และ GNOME เป็น GUI (ในทางปฏิบัติเราสามารถเลือกใช้ตัวใดตัวหนึ่งก็ได้) โดย GUI ทั้งสองพัฒนาต่อยอดเพิ่มเติมจากระบบการแสดงผลกราฟิกพื้นฐานของลินุกซ์ X-window ส่วน CLI นั้นมีให้เลือกหลายตัวเช่น sh ksh bash เป็นต้น (ในปฏิบัติการนักศึกษาจะได้ใช้ bash)
- คอมพิวเตอร์ในยุคหลังที่เริ่มหันมาใช้หน้าจอสัมผัส (Touchscreen Interface) ก็จะมีหน้าจอที่มีไอคอนต่างๆ เพื่อรับการสัมผัสสั่งการ
    - การออกแบบหน้าจอสัมผัส อาจจะเผื่อการใช้เมาส์ (อย่างหน้าจอสัมผัสของ Windows 10) หรืออาจจะไม่ได้เผื่อไว้เลย (อย่างหน้าจอของสมาร์ทโฟนโดยทั่วไป
    - การสั่งการทำโดยการสัมผัส หรือการลากเคลื่อนนิ้ว (gesture)
    - ใช้ Virtual Keyboard เพื่อการกรอกข้อมูล
    - อาจสั่งการโดยอาศัยช่องทางอื่นๆ เช่นการใช้เสียง หรือสั่งการผ่านการรับรู้ทางกล้อง (ใบหน้า การเคลื่อนไหวของมือ เป็นต้น)

<u>การติดต่อของโปรแกรมกับระบบปฏิบัติการ (System Calls)</u>

System calls คือส่วนติดต่อของโปรแกรมผู้ใช้ (งานใดๆ ที่กำลังรันอยู่ในระบบ) กับโปรแกรมย่อย หรือชุดคำสั่งที่ระบบปฏิบัติการได้เตรียมไว้เพื่อให้บริการ ในระบบสมัยก่อน เช่น DOS ชุดคำสั่งนี้จะเขียนด้วย Assembly และการเข้าถึงอาศัยการร้องขออินเทอร์รัพต์ ในระบบปัจจุบันคำสั่งเหล่านี้จะขียนด้วยภาษาระดับสูง เช่น C/C++ เป็นต้น การที่โปรแกรมจะเรียกใช้ชุดคำสั่งเหล่านี้ จะอาศัยไลบรารีที่ระบบปฏิบัติการได้เตรียมไว้ล่วงหน้าเพื่อให้เรียกใช้ ไลบรารีนี้อาจจะอยู่ในรูปไฟล์ชุดคำสั่งที่สามารถเชื่อมเข้ากับโปรแกรมในขณะคอมไพล์ หรือถูกเรียกใช้ในขณะรัน โดยเราเรียกฟังก์ชันที่ระบบปฏิบัติการได้เตรียมไว้เหล่านี้ว่า Application Program Interface (API) จุดมุ่งหมายของการสร้างเป็นฟังก์ชันไว้ให้เรียก แทนการเรียกแบบร้องขออินเทอร์รัพต์หรืออื่นใดนั้น ก็เพื่อการเลี่ยงไม่ให้โปรแกรมที่พัฒนาขึ้นนั้น ต้องขึ้นกับสภาพทางกายภาพหรือฮาร์ดแวร์ของระบบ แต่ให้ขึ้นกับแนวคิดการพัฒนาโปรแกรมแทน ทำให้ระบบปฏิบัติการสามารถรองรับการเปลี่ยนแปลงทางฮาร์ดแวร์ได้อย่างอิสระมากกว่า

หน้าที่ของ API

- เก็บพารามิเตอร์ต่างๆ ที่ต้องใช้โดยการกระทำ System call กับระบบ (เช่น file handle ค่าตำแหน่งหน่วยความจำที่จองมาใช้ชั่วคราว ฯลฯ)
- การเรียกใช้ system call กับระบบปฏิบัติการ (ตามกลไกของระบบปฏิบัติการแต่ละตัว) และรอรับผลลัพธ์ที่ได้ ลักษณะการเรียกใช้ system call อาจมองเปรียบได้กับการเรียกใช้ฟังก์ชันของภาษาซี หรือโปรแกรมย่อย
- ระบบปฏิบัติการจัดเก็บตารางของ system call ซึ่งแต่ละตัวจะมีเลขกำกับ (ในระดับการทำงานของเครื่อง จะเรียกโดยใช้ตัวเลข ในขณะที่ system-call interface จะทำหน้าที่เชื่อมระหว่างชื่อ system call ที่ผู้ใช้อาจเรียกใช้งานโดยตรงผ่าน shell กับตัวเลขประจำ system call
- เพื่ออำนวยความสะดวกแก่ผู้พัฒนาโปรแกรม เพราะผู้พัฒนาแค่ศึกษาว่าพารามิเตอร์แต่ละตัวของ API ต้องการอะไร และคำตอบที่ได้มีโครงสร้างอย่างไร

- ในทางปฏิบัติ ตัวชุดคำสั่งของ API อาจจะเชื่อมอยู่กับโปรแกรม (ซึ่งปัจจุบันมักจะไม่ค่อยใช้แล้ว) หรือเป็นโค้ดที่แยกต่างหากที่ระบบปฏิบัติการจะเรียกขึ้นมาเพื่อดำเนินการ (run-time library) อย่างในวินโดวส์ก็จะใช้ DLL เป็นต้น (หรือในลินุกซ์ก็จะเป็น so)
- ในการเรียก system call แต่ละครั้งแต่ละขั้นตอนใน API โหมดการทำงานก็จะยกระดับเป็น kernel mode ชั่วคราว และคืนกลับเป็น user mode เมื่อทำงานเสร็จแต่ละขั้น

กลไกการผ่านค่าระหว่างงานที่กำลังดำเนินอยู่กับชุดคำสั่งของระบบปฏิบัติการ เมื่อกระทำ system call มีดังเช่น

- การผ่านค่าทางเรจิสเตอร์ในซีพียู (ตัวอย่างพวกนี้ใช้ในการเรียก system call ด้วยการร้องขออินเทอร์รัพต์ ซึ่งใช้เป็นมาตรฐานในระบบปฏิบัติการสมัยเก่าเช่น DOS)
- การจองพื้นที่ในหน่วยความจำไว้ส่วนหนึ่ง นำข้อมูลที่ต้องการส่งผ่านไปเก็บไว้ในพื้นที่ดังกล่าว แล้วนำค่าอ้างอิงของพื้นที่มาเก็บไว้ในเรจิสเตอร์ของซีพียู แล้วจึงกระทำ system call (ใช้ในลินุกซ์ เป็นต้น)
- การเอาข้อมูลไปเก็บไว้ในพื้นที่สแต็กของโปรแกรม แล้วจึงเรียก system call โดยชุดคำสั่งของระบบปฏิบัติการจะอ่านข้อมูลจากพื้นที่สแต็กของโปรแกรมออกมาเพื่อนำไปดำเนินการ
- วิธีการสองอย่างหลัง ทำให้สามารถรับส่งข้อมูลที่มีขนาดใหญ่เกินกว่าที่บรรจุลงในเรจิสเตอร์ได้ทั้งหมด และมีความสะดวกต่อการพัฒนาโปรแกรมระดับสูงมากกว่า

<u>โปรแกรมระบบ (System Programs)</u>

โปรแกรมระบบ เป็นโปรแกรมที่บรรจุชุดคำสั่งที่ใช้ทำงานอย่างหนึ่งอย่างใดที่เกี่ยวข้องโดยตรงต่อการบริหารการจัดการของระบบปฏิบัติการ เป็นทางเลือกอำนวยความสะดวกให้กับการพัฒนาโปรแกรมตามปกติ โดยแทนที่ผู้พัฒนาโปรแกรมจะต้องไปเรียนรู้ API และเขียนโปรแกรมเชื่อมต่อการทำงานของโปรแกรมของตนกับ API และต้องระมัดระวังประเด็นต่างๆ มากมาย ก็เพียงแต่เรียกใช้โปรแกรมระบบ ที่ถูกจัดเตรียมไว้ล่วงหน้าแล้วในระบบหน่วยความจำสำรอง ก็สามารถทำงานดังที่ต้องการได้เช่นเดียวกัน ซึ่งข้อด้อยของการทำเช่นนี้ก็คือ overhead หรือเวลาที่เสียไปในการเรียกใช้โปรแกรมระบบ และทรัพยากรต่างๆ เช่น หน่วยความจำ ที่อาจต้องใช้มากกว่า

การออกแบบโปรแกรมระบบ ยังสามารถออกแบบโดยใช้มุมมองของผู้พัฒนาโปรแกรม และผู้ใช้ระบบเป็นหลัก แทนที่จะออกแบบโดยอาศัย system call เป็นหลัก นั่นคือ โปรแกรมระบบหนึ่งๆ อาจจะเรียกใช้ system call ที่หลากหลาย ซ้อนเหลื่อมกันไปในโปรแกรมระบบแต่ละตัว (คล้ายคลึงกับ API เช่นเดียวกัน)

โปรแกรมระบบอาจแบ่งออกดังเช่น

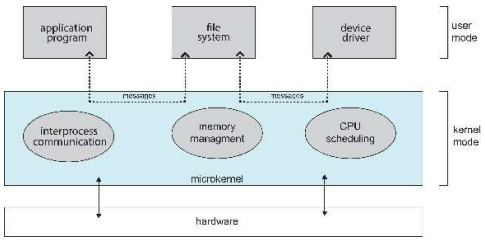
- โปรแกรมจัดการไฟล์ การสร้างไฟล์ใหม่ การลบไฟล์ การสำเนาไฟล์ เปลี่ยนชื่อไฟล์ ย้ายตำแหน่งไฟล์ ส่งข้อมูลในไฟล์ไปยังเครื่องพิมพ์ การแสดงข้อมูลในไฟล์ การเปลี่ยนแปลงโครงสร้างไดเรกตอรี เช่นการสร้าง การลบ การเปลี่ยนชื่อ การย้ายไดเรกตอรี เป็นต้น และอาจรวมถึงการเปลี่ยนแปลงสิทธิการใช้งานไฟล์ การสร้าง link หรือ shortcut เป็นต้น
- โปรแกรมแสดงรายละเอียดต่างๆ ของระบบ เช่นดูหรือเซ็ตเวลา ดูพื้นที่หน่วยความจำที่เหลือ พื้นที่หน่วยความจำสำรองที่เหลือ จำนวนผู้ใช้ ฯลฯ
- โปรแกรมแก้ไขไฟล์ เช่นโปรแกรม text editor (Notepad vi nano)
- โปรแกรมสนับสนุนการพัฒนาโปรแกรม เช่นคอมไพเลอร์ ดีบักเกอร์
- โปรแกรมอำนวยความสะดวกในการโหลดโปรแกรมอื่นขึ้นมาทำงาน
- โปรแกรมติดต่อสื่อสาร เช่นโปรแกรมส่งข้อความระหว่างเครื่อง outlook ftp telnet ssh ฯลฯ

สำหรับระบบปฏิบัติการสมัยใหม่ โปรแกรมดังกล่าวยังไม่เพียงพอต่อผู้ใช้ทั่วไป โดยโปรแกรมสมัยใหม่ได้รวมโปรแกรมผู้ใช้ (Application program) ที่นิยมใช้ หรือมีความจำเป็นต้องใช้ในสภาพแวดล้อมของระบบปฏิบัติการปัจจุบัน โปรแกรมเหล่านี้บางทีเราเรียกว่า ยูทิลิตีโปรแกรม (Utility Program) โปรแกรมเหล่านี้มีดังเช่น

- โปรแกรมสำนักงาน (Office) ที่พนักงานในสำนักงานมีความจำเป็นต้องใช้งานบ่อยๆ เช่น Word Write Calculator Excel Picture viewer ฯลฯ
- โปรแกรมอ่านเอกสารเครือข่าย (เบราว์เซอร์ Web Browser)
- ระบบฐานข้อมูล
- ชุดพัฒนาซอฟต์แวร์ Development toolkit และ Developement Environment (เช่น Visual Studio Eclipse เป็นต้น)
- เกม

1.4.2 ประเภทของระบบปฏิบัติการ แบ่งตามโครงสร้างการออกแบบ

- **Simple Structure** ระบบปฏิบัติการในสมัยแรก แบ่งองค์ประกอบการทำงานในลักษณะง่ายๆ โดยเป็นชุดคำสั่งเล็กๆ ที่ถูกเรียกใช้โดยการร้องขออินเทอร์รัพต์ เช่นระบบปฏิบัติการ DOS ที่มี ROM BIOS เป็นชุดคำสั่งที่เขียนลงใน ROM และฝังตัวเป็นส่วนหนึ่งของหน่วยความจำระบบ (ทำให้โปรแกรมไม่สามารถไปแก้ไขได้ เป็นการรักษาความปลอดภัยแบบเบื้องต้น) การเรียกใช้กระทำทางอินเทอร์รัพต์ ส่วนฮาร์ดแวร์นั้น ก็จะมี device drivers ที่โหลดไปไว้เป็นส่วนหนึ่งในหน่วยความจำตอนบูตเครื่อง และยังมีชุดคำสั่งอีกจำนวนหนึ่งที่ผู้ใช้อาจเพิ่มเข้าไปเพื่อให้เรียกใช้ผ่านทางการทำอินเทอร์รัพต์ ซึ่งเราเรียกชุดคำสั่งนี้ว่า resident program (เพราะวางตัวอยู่ในหน่วยความจำตลอดเวลา) จากนั้นก็จะมีโปรแกรมผู้ใช้ที่จะถูกโหลดขึ้นผ่านทาง shell ของระบบปฏิบัติการ
- **Layered Approach** การออกแบบในลักษณะนี้ จะกำหนดระดับความสำคัญของชุดคำสั่งหรือโมดูลต่างๆ ไว้ต่างกันเป็นเลเยอร์ หรือระดับชั้น โดยระดับล่างสุดมักจะทำหน้าที่ติดต่อกับฮาร์ดแวร์โดยตรง หรือเป็นชุดการทำงานพื้นฐานของระบบปฏิบัติการ ระดับที่สูงขึ้นเป็นโมดูลที่มักจะต้องเรียกใช้โมดูลในระดับต่ำกว่า ส่วนโปรแกรมผู้ใช้จะวางตัวในระดับสูงสุด
- **Microkernel Approach** การออกแบบในลักษณะนี้ จะคัดเอาฟังก์ชันการทำงานพื้นฐานเท่าที่จำเป็นที่สุดของระบบไปวางไว้ร่วมกันเป็นส่วนเดียว ส่วนงานอื่นๆ ที่เหลือจะอาศัยการดึงเข้ามาทำงานร่วมด้วยในภายหลัง เช่นเราอาจจะออกแบบ microkernel ประกอบด้วยการจัดการ process และ memory แต่เราจะสร้างโมดูลการจัดการ I/O file และ network ไว้ต่างหาก เพื่อให้เรียกเข้ามารวมภายหลัง โดยการสื่อสารระหว่างโมดูลจะใช้ message passing ด้วยวิธีเช่นนี้ทำให้เราสามารถถอดและเปลี่ยนโมดูลต่างๆ ที่เป็นบริการพื้นฐานของระบบปฏิบัติการได้โดยไม่ต้องหยุดระบบปฏิบัติการ ตัวอย่างเช่น QNX ที่เป็น real-time OS ตัวหนึ่ง ใช้กลไกการออกแบบในลักษณะนี้



Microkernel system structure

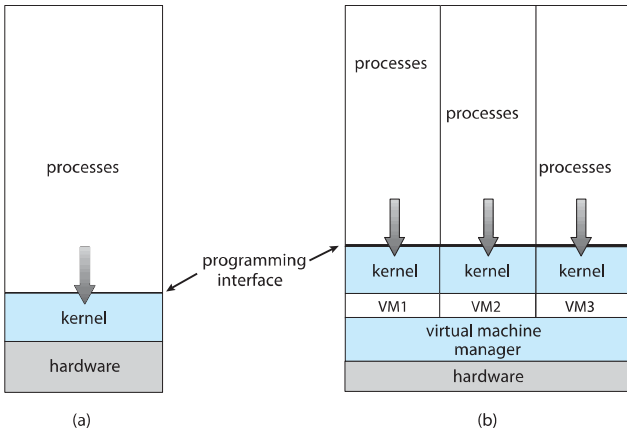
- **Modular Approach** การออกแบบในลักษณะนี้มองระบบปฏิบัติการในรูปแบบที่บริหารจัดการภายในแนวคิดเชิงวัตถุ โดยจะมี kernel เป็นองค์ประกอบหลัก และมีโมดูลต่างๆ ที่ทำงานเฉพาะด้าน ประกอบเข้าด้วยกันกลายเป็นระบบปฏิบัติการที่สมบูรณ์ นี่เป็นแนวคิดของการออกแบบระบบปฏิบัติการหลายตัวในยุคปัจจุบัน

- ตัวอย่างลูกผสมระหว่างโครงสร้างแบบก้อนเดียวกับการใช้ Modular design ได้แก่ระบบปฏิบัติการลินุกซ์ ที่จะมี Kernel หรือแก่นของระบบปฏิบัติการ ทำหน้าที่จัดการกลไกพื้นฐานอย่างการจัดการโพรเซส หน่วยความจำ และระบบเครือข่าย แต่องค์ประกอบอื่นๆ จะเป็นโมดูลที่เปลี่ยนแปลงในภายหลังได้

1.4.3 การจำลองโครงสร้างฮาร์ดแวร์ด้วยซอฟต์แวร์ (Virtual Machine / Virtualization)

Virtual Machine เป็นการจำลองโครงสร้างฮาร์ดแวร์ของระบบโดยตัว VM จะรันเป็นงานหนึ่งบนระบบปฏิบัติการที่เป็นผู้ดูแล (Host) โดย VM จะทำหน้าที่เลียนแบบคำสั่งพื้นฐานของซีพียูหรือส่งต่อการทำงานคำสั่งพื้นฐานของซีพียู และการติดต่อระบบต่างๆ ทางฮาร์ดแวร์ ด้วยชุดคำสั่ง (ที่เป็นซอฟต์แวร์) ดังนั้น VM หนึ่งๆ จะต้องการเวลาในการครอบครองซีพียู การเข้าถึงดีไวซ์ หน่วยความจำหลัก และหน่วยความจำสำรอง เพื่อนำมาจำลองเป็นสภาพทางฮาร์ดแวร์หนึ่ง ทำให้เราสามารถรันระบบปฏิบัติการ (guest) ภายใต้สภาพแวดล้อมที่ VM สร้างขึ้นมาได้

- บริษัท IBM เป็นผู้ริเริ่มพัฒนาขึ้นมาเป็นรายแรกเพื่อใช้บนเมนเฟรมตั้งแต่ปี 1972
- การใช้ VM ทำให้เราสามารถจำลองหน่วยฮาร์ดแวร์เสมือนได้หลายหน่วย และสามารถรัน Guest OS และซอฟต์แวร์ได้จำนวนมาก และสามารถทำให้เคลื่อนย้ายซอฟต์แวร์ทั้งระบบปฏิบัติการไปยังที่ใหม่ที่มีโครงสร้างกายภาพที่แตกต่างได้โดยสะดวก
- การใช้ VM ทำให้เราสามารถควบคุมการทำงานของซอฟต์แวร์ภายใน VM แต่ละตัวให้ทำงานได้เฉพาะภายในสภาพแวดล้อมดังกล่าว โดยจะไม่มีผลกระทบต่อ Guest System อื่นๆ
- VM ส่วนมากมักจะอนุญาตให้แชร์ทรัพยากรของ Host OS ไปให้ Guest OS ได้
- การติดต่อสื่อสารระหว่างกัน มักใช้การติดต่อผ่านระบบเครือข่ายจำลอง (ทำให้เสมือนหนึ่งว่าเป็นคอมพิวเตอร์หลายเครื่องรันอยู่ในระบบเครือข่ายเดียวกัน)
- การใช้ VM ทำให้ผู้พัฒนาโปรแกรมสามารถจำลองสภาพทางฮาร์ดแวร์เพื่อทดสอบซอฟต์แวร์ได้โดยไม่ต้องซื้อฮาร์ดแวร์จริงๆ มาทดลอง รวมถึงผู้ที่ทดสอบการทำงานของซอฟต์แวร์ประสงค์ร้าย เช่นโทรจัน หรือไวรัส สามารถเฝ้าดูพฤติกรรมของซอฟต์แวร์เหล่านั้นโดยไม่ส่งผลร้ายต่อ Host OS
- การใช้ VM ช่วยให้เราสามารถใช้งานฮาร์ดแวร์ได้อย่างเต็มประสิทธิภาพ ทั้งนี้เพราะในหลายการประยุกต์ใช้งานคอมพิวเตอร์มักจะมีประสิทธิภาพเกินจำเป็นต่องานเหล่านั้น ทำให้การใช้คอมพิวเตอร์หนึ่งเครื่อง เป็นการเสียเปล่าทางทรัพยากรโดยใช่เหตุ การใช้ VM เพื่อจำลองฮาร์ดแวร์หลายชุด แล้วนำ Guest OS พร้อมซอฟต์แวร์ไปลงรวมกันภายใต้ฮาร์ดแวร์เดียว ทำให้สามารถใช้ฮาร์ดแวร์เหล่านั้นได้เต็มประสิทธิภาพมากขึ้น (Consolidation)
- ในระบบสมัยใหม่เช่น Blade Server และใน Web Server นิยมเอา VM มาใช้เพื่อผลประโยชน์ของการแบคอัป และการกู้คืนสภาพในสภาวะที่ฮาร์ดแวร์เกิดปัญหา การติดตั้งระบบ และการดูแลจัดการระบบ รวมทั้งการปรับปรุงหรือเปลี่ยนฮาร์ดแวร์ที่มีสภาพกายภาพแตกต่างจากเดิม ทำให้ไม่ต้องเสียเวลาในการติดตั้งระบบปฏิบัติการใหม่ รวมทั้งติดตั้งซอฟต์แวร์ใหม่และปรับ configuration ของส่วนต่างๆ ใหม่ให้เสียเวลา



แนวทางการออกแบบและใช้งานตัว Virtual Machine มีดังเช่น

- Virtualization ในลักษณะเต็มรูปแบบ นั่นคือตัว VM จะจำลองสภาพทางฮาร์ดแวร์อย่างสมบูรณ์ ทำให้เราสามารถใช้ OS ธรรมดาทั่วไปไปรันเป็น Guest OS ได้ในทันที ข้อดีของ VM ชนิดนี้ก็คือการนำไปจำลองฮาร์ดแวร์เพื่อลงระบบนั้นง่าย ไม่ต้องปรับแก้ไขใดๆ และสามารถย้าย Guest OS พร้อมซอฟต์แวร์ทั้งหมดไปรันใน VM อื่นได้ง่ายโดยไม่ต้องแก้ไข แต่ข้อเสียที่สำคัญก็คือ VM ชนิดนี้จะเปลืองทรัพยากรค่อนข้างมาก และ overhead ในการทำงานสูง ทำให้ประสิทธิภาพการรันซอฟต์แวร์ภายใน Guest OS ต่ำกว่าการรันบน Host OS อย่างเห็นได้ชัด ตัวอย่าง VM ชนิดนี้ได้แก่ VMWare และ VirtualBox
- Para-Virtualization ในลักษณะนี้ อาศัยแนวคิดที่ว่า Guest OS นั้นมีโครงสร้างที่คล้ายกันมาก หรือเหมือนกันกับ Host OS โดยสิ่งที่แตกต่างนั้นอาจจะเป็นสภาพทางฮาร์ดแวร์บางอย่างที่จำลองแตกต่างกันไปเล็กน้อย ทำให้ VM สามารถส่งทอดชุดคำสั่งการทำงานส่วนใหญ่ ต่อไปให้ Host OS ประมวลแทนได้ ข้อเสียของการออกแบบ VM ในลักษณะนี้คือตัว GuestOS จะต้องปรับแก้ไขโค้ดภายในเพื่อให้สามารถรันบน VM ได้ และการย้าย Guest OS พร้อมกับซอฟต์แวร์ไปยังเครื่องใหม่ จำเป็นต้องย้ายไปวางบน HostOS และ VM ที่มีโครงสร้างในลักษณะเดิม มิฉะนั้นอาจจะเกิดปัญหาในการทำงานได้ แต่ข้อดีก็คือ overhead ในการทำงานนั้นต่ำมาก ในกรณีของ VM ลักษณะนี้เช่น Xen ในลินุกซ์ ซอฟต์แวร์สามารถรันได้ในความเร็วเกินกว่า 90 เปอร์เซนต์ภายใต้ Guest OS เมื่อเทียบกับการรันบน Host OS
- Virtual Machine เป็นการออกแบบสภาพแวดล้อมเสมือนขึ้นมาโดยไม่ยึดหลักสถาปัตยกรรมใดๆ เป็นการเฉพาะ โดยสภาพแวดล้อมเสมือนดังกล่าวจะสามารถรันโปรแกรมที่มีชุดคำสั่งที่ถูกออกแบบรูปแบบโค้ดไว้เป็นการเฉพาะ โดยการพัฒนาตัว VM นี้ให้มีหลายเวอร์ชันเพื่อไปรันบน HostOS ที่แตกต่างกัน ก็จะทำให้เราสามารถนำซอฟต์แวร์ที่พัฒนาและแปลเป็นภาษาที่ใช้เฉพาะ VM นี้ ไปใช้บนฮาร์ดแวร์และ HostOS ที่หลากหลายแตกต่างกันได้โดยไม่ต้อง port ซอฟต์แวร์ใหม่ ตัวอย่างในกรณี VM แบบนี้ได้แก่ Java Virtual Machine ที่จะรัน bytecode ซึ่งเป็น intermediate code ของจาวา ซึ่ง JVM นี้มีอยู่ทั้งบนวินโดวส์ ลินุกซ์ และระบบปฏิบัติการอื่นๆ ทำให้สามารถรันซอฟต์แวร์ที่พัฒนาขึ้นนี้ได้โดยไม่ต้องแก้ไขเปลี่ยนแปลงตัวซอฟต์แวร์แต่ประการใด ข้อดีอีกอย่างที่สำคัญก็คือ หากซอฟต์แวร์เกิดความผิดพลาดในการทำงาน หรือพยายามเข้าถึงองค์ประกอบอื่นในระบบที่ไม่ได้รับอนุญาต JVM ก็จะสามารถหยุดการทำงานของซอฟต์แวร์นั้นได้โดยจะไม่มีผลเสียหายใดๆ ต่อ Host OS เลยแม้แต่น้อย