บทที่ 9
การจัดการแฟ้มข้อมูล

หน้าที่สำคัญของระบบปฏิบัติการ คือ การจัดการแฟ้มข้อมูล (Files management) ซึ่งเป็นการช่วยอำนวยความสะดวกแก่ผู้ใช้งานในการเรียกใช้แฟ้มข้อมูลต่าง ๆ ผู้ใช้งานสามารถเรียกใช้คำสั่งเพื่อดำเนินการต่าง ๆ กับแฟ้มข้อมูลได้ เช่น การสร้างแฟ้มข้อมูล การเปิดแฟ้มข้อมูล หรือการลบแฟ้มข้อมูล การทำงานในระบบคอมพิวเตอร์ทั้งหมดจำเป็นต้องมีการเก็บและนำข้อมูลไปใช้งาน ขณะที่โพรเซสกำลังทำงานข้อมูลจะเก็บไว้ในหน่วยความจำ ถ้าเครื่องคอมพิวเตอร์ดับไม่ว่าด้วยสาเหตุใดก็ตาม ข้อมูลทั้งหมดจะสูญหายไปดังนั้นจึงจำเป็นต้องจัดเก็บข้อมูลเหล่านี้ไว้ในหน่วยจัดเก็บข้อมูลสำรอง ซึ่งอาจจะเป็นแผ่นดิสก์เก็ต ฮาร์ดดิสก์ หรืออุปกรณ์อื่น ๆ ในการจัดเก็บข้อมูลเหล่านี้มีจุดประสงค์เพื่อนำมาใช้งานต่อไป จึงจำเป็นต้องมีการกำหนดชื่อเพื่อแทนกลุ่มข้อมูล ซึ่งเราเรียกว่า แฟ้มข้อมูล

9.1 แนวคิดเกี่ยวกับแฟ้มข้อมูล

แฟ้มข้อมูล คือ ชื่อของสารสนเทศที่สัมพันธ์กัน ซึ่งถูกบันทึกไว้ในหน่วยเก็บข้อมูลสำรอง (Secondary storage) ในมุมมองของผู้ใช้แฟ้มข้อมูลคือการจัดสรรที่เล็กที่สุดของหน่วยเก็บข้อมูลสำรอง ซึ่งข้อมูลนี้ไม่สามารถเขียนไปยังหน่วยความจำหลักได้ เว้นแต่จะถูกจัดเก็บภายในแฟ้มของข้อมูล ปกตินั้นแฟ้มข้อมูลจะถูกแสดงโดยทางโปรแกรมซึ่งจะแสดงข้อมูลของแฟ้มข้อมูลด้วย ข้อมูลที่อยู่ในแฟ้มข้อมูลจะถูกแสดงออกมาเป็นตัวเลขตามตัวอักษรที่อยู่ข้างใน หรือเป็นเลขฐานสอง แฟ้มข้อมูลนั้นอาจจะเป็นอิสระต่อกัน เช่น แฟ้มข้อมูลของข้อความหรืออาจมีการจัดรูปแบบแฟ้มข้อมูลที่มีความเข้มงวด แฟ้มข้อมูลถูกกำหนดเป็นโครงสร้างตามชนิดของข้อมูลดังนี้

1. Text File คือ ลำดับของตัวอักษรที่เรียงกันในบรรทัด (หรือหน้า)

2. Source File คือ ลำดับของโปรแกรมย่อย (Subroutine) และฟังก์ชัน (อาจเป็นการประกาศค่าตามประโยค)

3. Object File คือ ลำดับของไบต์ ที่จัดเรียงในบล็อคที่ตัวเชื่อมโยง (Linker) ของระบบเข้า

4. Executable File คือ ลำดับของส่วนของรหัสโปรแกรมซึ่งตัว Load โปรแกรม (Loader) นำเข้ามายังหน่วยความจำและสั่งให้ทำงาน (Execute)

9.1.1 คุณลักษณะของแฟ้มข้อมูล (File Attributes)

จุดประสงค์ในการออกแบบระบบปฏิบัติการอย่างหนึ่งก็คือ ต้องการที่จะให้ผู้ใช้เป็นอิสระจากอุปกรณ์ใด ๆ (Device independent) ดังนั้น ในการเข้าถึงแฟ้มข้อมูลใด ๆ จะต้องมีรูปแบบเดียวกัน นอกจากนั้นวิธีการในการเข้าถึงแฟ้มข้อมูล ไม่จำเป็นต้องกำหนดรายละเอียดหรือหมายเลขตำแหน่งที่เก็บให้ยุ่งยากวุ่นวาย เพียงแค่ระบุชื่อและนามสกุลของแฟ้มข้อมูลให้ถูกต้องก็เพียงพอแล้ว คุณลักษณะของแฟ้มข้อมูลที่แตกต่างกันขึ้นอยู่กับระบบปฏิบัติการ แต่โดยปกติจะประกอบด้วยสิ่งเหล่านี้

- ชื่อ (Name) ชื่อแฟ้มข้อมูลที่เป็นสัญลักษณ์เป็นเพียงข้อมูลที่เก็บไว้ในรูปแบบที่อ่านได้โดยมนุษย์
- ตัวระบุ (Identifier) แท็กที่ไม่ซ้ำกันนี้มักจะเป็นหมายเลขตัวระบุแฟ้มข้อมูลภายในระบบแฟ้ม
- ประเภท (Type) ข้อมูลนี้จำเป็นสำหรับระบบที่สนับสนุนประเภทต่าง ๆ ของแฟ้มข้อมูล
- ตำแหน่ง (Location) ข้อมูลนี้เป็นตัวชี้ไปยังอุปกรณ์และตำแหน่งของแฟ้มที่อยู่ในฮาร์ดดิสก์
- ขนาด (Size) จะบอกขนาดปัจจุบันของแฟ้มข้อมูล (หน่วยเป็นไบต์) ซึ่งขนาดสูงสุดจะถูกกำหนดเอาไว้
- การป้องกัน (Protection) จะใช้ควบคุมสิทธิ์ในการเข้าถึงแฟ้มข้อมูลต่าง ๆ เช่น การอ่านเขียนแฟ้มข้อมูล
- วันเวลาของผู้ใช้ (Time, Date, and User Identification) ข้อมูลนี้จะถูกเก็บไว้เมื่อมีการแก้ไขแฟ้มข้อมูลนี้ในครั้งล่าสุด ซึ่งมีประโยชน์สำหรับป้องกันความปลอดภัย

ข้อมูลเกี่ยวกับแฟ้มข้อมูลทั้งหมดนั้นถูกเก็บอยู่ในไดเรกทอรี่ ซึ่งถูกจัดเก็บในหน่วยเก็บข้อมูลสำรอง โดยปกติไดเรกทอรี่จะประกอบไปด้วยชื่อแฟ้มข้อมูลและรายละเอียดที่ไม่ซ้ำกัน รวมทั้งคุณลักษณะของแฟ้มข้อมูล อาจจะใช้เวลานานในการบันทึกข้อมูลเหล่านี้ ในระบบที่มีอยู่หลายแฟ้มข้อมูล ไดเรกทอรี่ที่เก็บแฟ้มข้อมูลก็จะมีขนาดใหญ่ตามไปด้วย เนื่องมาจากแฟ้มข้อมูลในไดเรกทอรี่ต้องการพื้นที่ในการจัดเก็บข้อมูลเช่นเดียวกับข้อมูลชนิดอื่น ๆ

9.1.2 การดำเนินการกับแฟ้มข้อมูล (File Operations)

แฟ้มข้อมูลจะมีชนิดข้อมูลที่จับต้องไม่ได้ การที่จะกำหนดแฟ้มข้อมูลให้เหมาะสมได้ต้องพิจารณาตัวดำเนินการทั้งหมดที่อยู่ในแฟ้มข้อมูล ระบบปฏิบัติการจะช่วยสนับสนุนทางด้านการสร้าง เขียน และอ่านแฟ้มข้อมูล หรือบางทีก็เป็นการลบหรือตัดทอนแฟ้มข้อมูล ในการตรวจสอบแฟ้มข้อมูลนั้น ระบบปฏิบัติการจะใช้พื้นฐานต่าง ๆ ที่มีอยู่ในการตรวจสอบแฟ้มข้อมูล การดำเนินการจะทำได้ง่ายคล้ายกับการเปลี่ยนชื่อแฟ้มข้อมูลโดยมีการดำเนินการกับแฟ้มข้อมูลดังนี้

1. การสร้างแฟ้มข้อมูล (Creating a file) จะมีอยู่สองขั้นตอนในการสร้าง ขั้นตอนแรกต้องรู้ที่ว่างว่ามีอยู่ในระบบหรือไม่ ขั้นตอนที่สองการสร้างแฟ้มข้อมูลใหม่จำเป็นต้องสร้างอยู่ในไดเรกทอรี่

2. การเขียนแฟ้มข้อมูล (Writing a file) ในการเขียนแฟ้มข้อมูลเราจะทำให้ข้อมูลถูกเขียนขึ้นทั้งชื่อแฟ้มข้อมูลและรายละเอียดที่ถูกเขียนไปยังแฟ้มข้อมูล การระบุชื่อแฟ้มข้อมูลนั้นจะทำให้สามารถหาได้ว่า ที่ตั้งของแฟ้มข้อมูลอยู่ในไดเรกทอรี่ไหน เพื่อหาตำแหน่งของแฟ้มข้อมูลระบบต้องเก็บ ระบบจะทำการเก็บตัวชี้ (Write Pointer) ไปยังตำแหน่งที่หาเจอในครั้งถัดไปที่เขียนอีกครั้ง การชี้ตำแหน่งจะถูกปรับปรุงเมื่อเกิดการเขียนขึ้นทุกครั้ง

3. อ่านแฟ้มข้อมูล (Reading a file) หากเราต้องการอ่านแฟ้มข้อมูลเราก็ต้องรู้ถึงตำแหน่งของแฟ้มข้อมูลในไดเรกทอรี่ โดยต้องให้ตัวชี้ชี้ไปยังตำแหน่งที่อยู่ของแฟ้มข้อมูล ทุกครั้งที่มีการอ่านแฟ้มข้อมูลตัวชี้ (Read pointer) จะมีการปรับปรุงให้เป็นตำแหน่งของแฟ้มข้อมูลปัจจุบัน เพื่อประหยัดพื้นที่และลดความซ้ำซ้อนของระบบในการอ่านและเขียนแฟ้มข้อมูล

4. ที่เก็บแฟ้มข้อมูลภายในแฟ้ม (Repositioning within a file) เริ่มต้นจากการค้นหา ไดเรกทอรี่ที่ต้องการและกำหนดค่าให้ตัวชี้ตำแหน่งแฟ้มข้อมูลปัจจุบัน ที่สามารถค้นหาตำแหน่งที่อยู่ของแฟ้มข้อมูลในไดเรกทอรี่ได้ แฟ้มข้อมูลชี้ตำแหน่งคือ ตำแหน่งที่ทำการระบุค่าของที่เก็บแฟ้มข้อมูล การย้ายตำแหน่งภายในแฟ้มข้อมูลไม่จำเป็นต้องเกี่ยวข้องกับ I/O จริง ๆ เลยการทำงานของแฟ้มนี้เรียกว่า การค้นหา (Seek) ข้อมูล

5. การลบแฟ้มข้อมูล (Deleting a file) ในการลบแฟ้มข้อมูลเราจำเป็นต้องรู้ตำแหน่งของแฟ้มข้อมูลในไดเรกทอรี่ เริ่มจากการค้นหาไดเรกทอรี่ที่มีชื่อจากแฟ้มที่จะลบ โดยจะพบว่ามีการเชื่อมโยงกันของแฟ้มข้อมูลที่อยู่ในไดเรกทอรี่ การลบแฟ้มข้อมูลจึงต้องรู้ตำแหน่งที่แน่นอน

6. การตัดทอนแฟ้มข้อมูล (Truncating a file) เมื่อผู้ใช้ต้องการให้แฟ้มข้อมูลมีคุณลักษณะเหมือนเดิม แต่ต้องการลบเนื้อหาของแฟ้มข้อมูลแทนที่จะลบและสร้างใหม่ ผู้ใช้อาจจะลบเนื้อหาของแฟ้มข้อมูลในบางส่วนแทนที่จะทำการลบแฟ้มข้อมูลทั้งหมด แต่ทำการตัดทอนเนื้อหาบางส่วนที่ไม่เอาแทน ซึ่งช่วยให้พื้นที่ในการจัดเก็บแฟ้มข้อมูลน้อยลง

นอกจากนี้ยังมีการดำเนินการอื่น ๆ อีกเช่น การต่อท้าย (Appending) การเปลี่ยนชื่อ (Renaming) การคัดลอก (Copy) แฟ้มข้อมูลหรือคัดลอกแฟ้มไปสู่อุปกรณ์รับส่งข้อมูลอีกตัวหนึ่ง เช่น เครื่องพิมพ์ หรือ จอภาพ โดยต้องมีการสร้างแฟ้มใหม่และอ่านจากแฟ้มเก่าเพื่อเขียนลงแฟ้มใหม่ การทำงานของแฟ้มข้อมูลโดยส่วนใหญ่มักเกี่ยวข้องกับการค้นหาไดเรกทอรี เพื่อหาช่องที่เกี่ยวข้องกับชื่อแฟ้ม เพื่อหลีกเลี่ยงการค้นหานี้หลาย ๆ ระบบจะเปิดแฟ้มข้อมูลเมื่อถูกใช้ครั้งแรก ระบบปฏิบัติการจะเก็บตารางเล็ก ๆ ที่บรรจุสารสนเทศเกี่ยวกับการเปิดแฟ้มข้อมูลทั้งหมด (Open file table) เมื่อการทำงานเกี่ยวกับแฟ้มถูกร้องขอก็เพียงแต่ใช้ดัชนีในตาราง ทำให้ไม่ต้องใช้การค้นหาอีกต่อไป เมื่อไม่ต้องการใช้แฟ้มข้อมูลนั้นแล้วจะถูกปิดโดยกระบวนการและระบบปฏิบัติการจะเอาแฟ้มข้อมูลออกจาก Open file table

ในบางระบบเมื่อทำการปิดแฟ้มข้อมูล แฟ้มข้อมูลดังกล่าวจะถูกดำเนินการปิดเองโดยอัตโนมัติ แต่เมื่อการทำงานหรือโปรแกรมดังกล่าวเกิดความล้มเหลว ระบบส่วนใหญ่ก็จะใช้วิธีการนี้ในการเปิดปิดแฟ้มข้อมูล แต่ยังไงก็ตามระบบปฏิบัติการก็ต้องการความชัดเจนที่เกิดขึ้นกับการใช้งาน โดยระบบปฏิบัติการจะเช็คค่าความถูกต้องในสิทธิ์ของผู้ใช้ เพื่อให้ผู้ที่ใช้งานสามารถดำเนินการต่าง ๆ กับแฟ้มข้อมูลที่ใช้ได้ โดยสิทธิ์ของการเข้าถึงแฟ้มข้อมูลก็จะถูกเก็บไว้ในพื้นที่เล็ก ๆ ในระบบปฏิบัติการ โดยสรุปมีข้อมูลหลายส่วนที่เกี่ยวข้องกับการเปิดแฟ้มข้อมูล ดังนี้

- **ตัวชี้แฟ้มข้อมูล (File pointer)** ระบบจะไม่ติดตามแฟ้มข้อมูลที่กำลังอ่านและเขียนแฟ้มข้อมูลอยู่ แต่ระบบจะติดตามการอ่านเขียนครั้งสุดท้าย เหมือนตัวชี้ตำแหน่งแฟ้มข้อมูลปัจจุบัน (Current-file-position pointer) ตัวชี้ของแต่ละกระบวนการจะไม่ซ้ำกัน ดังนั้นจึงต้องมีการจัดเก็บแยกคุณลักษณะของแฟ้มข้อมูลบนดิสก์
- **การนับการเปิดแฟ้มข้อมูล (File-open count)** เมื่อแฟ้มข้อมูลที่กำลังใช้งานถูกปิดลง ระบบปฏิบัติการต้องใช้ตารางเปิดแฟ้มข้อมูล (Open file table) ซ้ำอีกครั้ง หรือไม่ก็ใช้ที่ว่างในตาราง โพรเซสหลาย ๆ โพรเซสอาจจะเปิดแฟ้มข้อมูลเดียวกันได้ ระบบต้องรอให้แฟ้มสุดท้ายถูกปิดเสียก่อน จึงจะลบข้อมูลในตารางเปิดแฟ้มอันสุดท้าย และลบตารางนั้นได้
- **ตำแหน่งของแฟ้มข้อมูลบนดิสก์ (Disk location of the file)** การทำงานของแฟ้มข้อมูลส่วนใหญ่ต้องการให้ระบบปรับปรุงข้อมูลภายในแฟ้มข้อมูล สำหรับข้อมูลที่จำเป็นเพื่อระบุ

ตำแหน่งแฟ้มข้อมูลบนดิสก์จะถูกเก็บไว้ในหน่วยความจำ เพื่อหลีกเลี่ยงที่จะต้องอ่านจากดิสก์สำหรับการทำงานแต่ละครั้ง

- **สิทธิ์ในการเข้าใช้ (Access rights)** แต่ละกระบวนการมีการเข้าถึงไม่เหมือนกัน ข้อมูลที่ถูกเก็บเอาไว้ในกระบวนการมีจำนวนมาก จึงต้องมีการกำหนดสิทธิ์ในการเข้าใช้ เพื่อป้องกันการขโมยข้อมูล โดยต้องดูจากคำขอใช้แฟ้มข้อมูลที่ร้องขอเข้ามา

ระบบปฏิบัติการบางระบบจะให้สิ่งอำนวยความสะดวกในการเชื่อมต่อหรือเปิดแฟ้มข้อมูล โดยจะมีคีย์ในการเปิดแฟ้มข้อมูล เพื่อป้องกันกระบวนการอื่นที่เราไม่ต้องการเข้าถึงแฟ้มข้อมูล แฟ้มข้อมูลที่ถูกล็อคมีประโยชน์สำหรับการใช้แฟ้มข้อมูลร่วมกันโดยหลาย ๆ กระบวนการ เช่น เราต้องล็อคแฟ้มข้อมูลเพื่อให้ผู้ใช้ที่มีสิทธิ์จริง ๆ มาแก้ไขในภายหลัง นอกจากนี้ยังอาจจะทำให้ระบบปฏิบัติการที่แตกต่างกันสามารถเข้ามาใช้หรือล็อคแฟ้มข้อมูลร่วมกันได้ โดยเมื่อมีการล็อคแฟ้มข้อมูลผู้ใช้งานจะต้องรู้ว่าระบบปฏิบัติการที่ตนเองกำลังใช้นั้นจะมีกลไกในการปลดล็อคแฟ้มข้อมูลที่จะใช้อย่างไรเพื่อให้เกิดความเท่าเทียมกัน ผู้ใช้จึงต้องมีความสำคัญในการปลดล็อดแฟ้มข้อมูลเหล่านี้

9.1.3 ประเภทของแฟ้มข้อมูล (File types)

เมื่อเรามีการออกแบบระบบแฟ้มที่อยู่ในระบบปฏิบัติการ เราก็ต้องพิจารณาว่าระบบปฏิบัติการควรรู้จักและสนับสนุนแฟ้มข้อมูลประเภทไหน เราต้องรู้ว่าระบบปฏิบัติการต้องการแฟ้มข้อมูลประเภทไหนที่สามารถทำงานร่วมกับแฟ้มข้อมูลในลักษณะที่เหมาะสมได้ เทคนิคโดยทั่วไปในการใช้ประเภทของแฟ้มข้อมูลคือ การที่เรารู้ถึงส่วนหนึ่งที่อยู่ในชื่อแฟ้มข้อมูล ชื่อจะแบ่งออกเป็นสองส่วน โดยชื่อและส่วนที่จะขยายจะถูกค้นด้วยอักขระ

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

ภาพที่ 9.1 ประเภทของแฟ้มข้อมูลทั่วไป
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.511)

ในที่นี้จะทำให้ระบบปฏิบัติการสามารถแยกออกว่าเป็นแฟ้มข้อมูลประเภทไหน โดยทั่วไปแล้วเราจะแบ่งแฟ้มข้อมูลออกเป็นชนิดต่าง ๆ ขึ้นอยู่กับการใช้งานของแฟ้มข้อมูลนั้น ๆ โดยส่วนมากเราจะแยกแยะได้จากนามสกุลหรือส่วนขยาย (Extension) ที่แตกต่างกัน ดังภาพที่ 9.1

9.1.4 โครงสร้างของแฟ้ม (File Structure)

ประเภทของแฟ้มข้อมูลนั้นยังสามารถใช้เพื่อแสดงโครงสร้างที่อยู่ภายในของแฟ้มข้อมูลได้ โดยโครงสร้างของแฟ้มข้อมูลจะต้องตรงกับความหมายของโปรแกรมที่ใช้งานอยู่ การเพิ่มเติมบางส่วนของแฟ้มข้อมูลจะต้องสอดคล้องกับโครงสร้างที่จำเป็นของระบบปฏิบัติการ ตัวอย่างเช่น ระบบปฏิบัติการได้มีการสร้างโครงสร้างเพื่อจะระบุที่อยู่ในหน่วยความจำ โดยการระบุตำแหน่งแรกของชุดคำสั่งบางระบบปฏิบัติการได้ขยายความคิดนี้เป็นชุดของระบบ เพื่อสนับสนุนโครงสร้างของแฟ้มข้อมูลกับชุดคำสั่งพิเศษสำหรับการดำเนินการจัดการกับโครงสร้างของแฟ้มข้อมูล

โครงสร้างข้อมูล หมายถึง ลักษณะการจัดแบ่งพิกัดต่าง ๆ ของข้อมูลสำหรับแต่ละระเบียน (Record) ในแฟ้มข้อมูลเพื่อให้คอมพิวเตอร์สามารถรับไปประมวลผลได้ ประกอบด้วยส่วนต่าง ๆ ดังนี้

1. บิต (Bit : Binary Digit) คือ หน่วยของข้อมูลที่เล็กที่สุดที่เก็บอยู่ในหน่วยความจำภายในคอมพิวเตอร์ ซึ่งบิตจะแทนด้วยตัวเลข คือ 0 หรือ 1 อย่างใดอย่างหนึ่งเท่านั้น เรียกตัวเลข 0 หรือ 1 ถ้ามีหนึ่งหลักว่าเป็นบิต 1 บิต

2. ไบต์ (Byte) หรือ ตัวอักขระ (Character) คือ หน่วยของข้อมูลที่นำบิตหลาย ๆ บิตมารวมกันแทนตัวอักษรแต่ละตัว เช่น A, B, ..., Z, 0, 1, 2, ... ,9 และสัญลักษณ์พิเศษอื่น ๆ เช่น $, &, +, -, *, / โดยตัวอักษร 1 ตัวจะแทนด้วยบิตจำนวน 7 หรือ 8 บิต ซึ่งตัวอักษรแต่ละตัวจะเรียกว่า ไบต์ เช่น ตัวอักษร A เมื่อเก็บอยู่ในคอมพิวเตอร์จะเก็บเป็น 1000001 ส่วนตัวอักษร B จะเก็บเป็น 1000010 เป็นต้น

3. เขตข้อมูล (Field) หรือคำ (Word) คือ หน่วยของข้อมูลที่เกิดจากการนำอักขระหลาย ๆ ตัวมารวมกัน เพื่อแทนความหมายของสิ่งหนึ่งเป็นคำที่มีความหมาย เช่น ชื่อ นามสกุล เป็นต้น

4. ระเบียน (Record) คือ หน่วยของข้อมูลที่มีการนำเขตข้อมูลหลาย ๆ เขตข้อมูล ที่มีความสัมพันธ์กันมารวมกัน หรือค่าของข้อมูลในแต่ละเขตข้อมูลมารวมกัน เพื่อแสดงรายละเอียดข้อมูลในเรื่องใดเรื่องหนึ่ง เช่น ระเบียนหนึ่ง ๆ ของพนักงานประกอบด้วย ฟิลด์ต่าง ๆ เช่น รหัสนักศึกษา ชื่อ หลักสูตร เป็นต้น

5. แฟ้มข้อมูล (File) คือ หน่วยของข้อมูลที่มีการนำระเบียนหลาย ๆ ระเบียนที่มีความสัมพันธ์กันมารวมกัน

6. ฐานข้อมูล (Database) คือ หน่วยของข้อมูลที่มีการนำแฟ้มข้อมูลหลาย ๆ แฟ้มข้อมูล ที่มีความสัมพันธ์กันมารวมกัน เช่น ฐานข้อมูลในระบบทะเบียนนักศึกษาจะประกอบด้วย แฟ้มข้อมูลรายวิชา นักศึกษา การลงทะเบียน ผลการเรียนประจำเทอม โปรแกรมวิชา และคณะ เป็นต้น

9.1.5 โครงสร้างของแฟ้มข้อมูลภายใน (Internal File Structure)

ในระบบดิสก์ มักมีขนาดของบล็อกที่กำหนดโดยขนาดของเซกเตอร์ การเข้าใช้พื้นที่ในดิสก์ทั้งหมด (Disk I/O) ถูกแบ่งเป็นหน่วยของบล็อก (Physical record) และทุกบล็อกมีขนาดเท่ากัน โดย Physical

record จะต้องตรงกับความยาวของ logical record การห่อ (Packing) จำนวนของ Logical record ไปสู่ Physical block เป็นวิธีโดยทั่วไปในการแก้ปัญหานี้

ขนาดของ Logical record ขนาดของ Physical block และเทคนิคการห่อ เป็นตัวกำหนดว่า Logical record จำนวนเท่าไรที่จะอยู่ในแต่ละ Physical block การห่อสามารถทำได้โดยโปรแกรมประยุกต์ของผู้ใช้หรือโดยระบบปฏิบัติการ

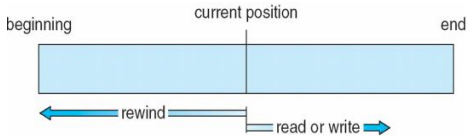
อีกกรณีหนึ่ง แฟ้มข้อมูลอาจถูกพิจารณาเป็นบล็อกแบบเรียงลำดับฟังก์ชัน I/O พื้นฐานทั้งหมดจะทำงานในแบบของบล็อก การแปลงจาก Logical record ไปเป็น Physical block คือปัญหาอย่างง่ายของซอฟแวร์ สังเกตได้ว่าพื้นที่ว่างในดิสก์มักถูกจัดสรรเป็นบล็อก บางส่วนของบล็อกสุดท้ายของแต่ละแฟ้มข้อมูลอาจจะเสียไป ถ้าแต่ละบล็อกมี 512 ไบต์ ดังนั้นแฟ้มขนาด 1949 ไบต์ ควรมี 4 บล็อก (2048 ไบต์) ที่เหลือ 99 ไบต์จะเสียไป นั่นคือ การสูญเสียพื้นที่ย่อยภายใน (Internal fragmentation) ฉะนั้นถ้าบล็อกขนาดใหญ่ก็จะเสียพื้นที่มากตามไปด้วย

9.2 วิธีการเข้าถึงแฟ้มข้อมูล

แฟ้มข้อมูลถูกใช้เก็บข้อมูลสารสนเทศ เมื่อถูกนำมาใช้ย่อมมีการเข้าถึงเพื่ออ่านข้อมูล โดยวิธีในการเข้าถึงข้อมูลนั้นก็แล้วแต่ระบบปฏิบัติการว่าเป็นชนิดไหน แต่การเข้าถึงโดยส่วนใหญ่จะมีวิธีดังนี้

9.2.1 วิธีเข้าถึงโดยลำดับ (Sequential Access)

เป็นเทคนิคที่ใช้ในการเก็บข้อมูลอย่างหนึ่ง เพื่อให้สามารถเรียกมาใช้ได้อย่างมีประสิทธิภาพ ใช้ในโปรแกรมประเภทคลังข้อมูล การเข้าถึงหน่วยเก็บข้อมูลหรือสื่อบางชนิด เช่น แถบแม่เหล็ก (Tape) ซึ่งจะเก็บข้อมูลไว้โดยเรียงไปตามลำดับ วิธีที่ใช้ในการเข้าถึงข้อมูลขึ้นอยู่กับระยะทางของตำแหน่งของข้อมูลที่บรรจุไว้ในสื่อนั้นดังแสดงในภาพที่ 9.2 หากข้อมูลถูกเก็บอยู่ตอนปลายกว่าจะเข้าถึงข้อมูลย่อมจะเสียเวลานาน



ภาพที่ 9.2 แสดงแฟ้มข้อมูลที่มีการเข้าถึงแบบเรียงลำดับ
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.513)

9.2.2 การเข้าถึงโดยตรง (Direct Access)

หมายถึง การเข้าถึงข้อมูลโดยใช้เวลาในการค้นหาข้อมูลได้เร็วเท่ากันหมด ไม่ขึ้นกับตำแหน่งที่เก็บ หมายความว่า ไม่ว่าข้อมูลจะเก็บอยู่ที่ส่วนใดของสื่อที่ใช้บันทึก หัวอ่าน (Read head) ก็จะเจาะตรงลงไปอ่านได้เลย เช่น การอ่านข้อมูลจากจานบันทึก ซึ่งผิดกับการอ่านข้อมูลจากแถบบันทึกที่ต้องอ่านเรียงไปตามลำดับตั้งแต่ต้นเทปไปจนกว่าจะถึงข้อมูลที่ต้องการทุกครั้ง ทำให้ช้ากว่ากันมาก

โดยจำนวนของบล็อกจะถูกป้องกันโดยระบบปฏิบัติการที่จะใช้หมายเลขในการป้องกัน ซึ่งการเรียงลำดับของบล็อกคือ 0 ถัดไปก็เป็น 1 ในความเป็นจริงบล็อกที่อยู่ในฮาร์ดดิสก์อาจจะเป็น 14703 สำหรับจุดเริ่มต้นและ 3192 เป็นลำดับถัดไป การใช้หมายเลขกำกับในแต่ละบล็อกช่วยให้ระบบปฏิบัติการตัดสินใจใช้แฟ้มข้อมูลและช่วยป้องกันแฟ้มข้อมูลจากการเข้าถึงที่ไม่ต้องการได้ โดยระบบที่มีหมายเลขกำกับอยู่กับบล็อกจะเริ่มต้นที่ 0 ส่วนระบบอื่นจะเริ่มต้นที่ 1

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

ภาพที่ 9.3 การจำลองคำสั่งการเข้าถึงแบบเรียงลำดับบนแฟ้มข้อมูลที่มีการเข้าถึงแบบโดยตรง

9.2.3 วิธีการเข้าถึงอื่น ๆ (Other Access Methods)

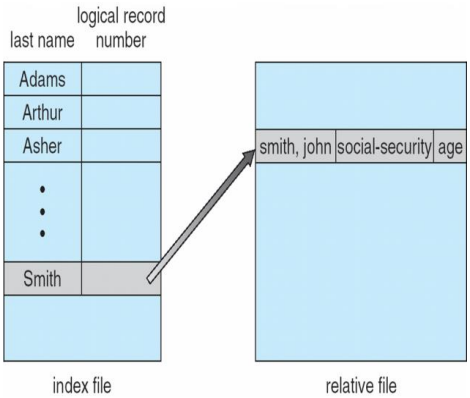
วิธีการเข้าถึงแบบอื่นที่สามารถทำได้นอกจากวิธีการเข้าถึงแบบ Direct-access วิธีการเหล่านี้โดยทั่วไปแล้วเกี่ยวข้องกับโครงสร้างของตัวชี้ของแฟ้มข้อมูล ตัวชี้ของแฟ้มข้อมูลนี้เหมือนกับหน้าดัชนี (Index) ที่อยู่ท้ายเล่มของหนังสือ ประกอบไปด้วยตัวชี้ตำแหน่งในหลาย ๆ ส่วน เอาไว้สำหรับค้นหาข้อมูลในแฟ้มข้อมูล ในการค้นหาครั้งแรกใช้ดัชนีและใช้ตัวชี้ในการเข้าถึงแฟ้มข้อมูล และค้นหาข้อมูลที่เราต้องการวิธีเสริมนี้เกี่ยวข้องกับการสร้างดัชนีให้แฟ้มข้อมูล ดัชนีนี้บรรจุตัวชี้บล็อกในการหาระเบียนในแฟ้มข้อมูล ขั้นแรกต้องค้นหาดัชนีแล้วใช้ตัวชี้ในการเข้าถึงแฟ้มข้อมูลโดยตรงและหาระเบียนที่ต้องการ

ตัวอย่าง แฟ้มข้อมูลเก็บราคาสินค้า แต่ละระเบียนประกอบด้วย รหัสสินค้า 10 หลัก ราคา 6 หลัก รวมเป็น 16 ไบต์ ถ้าดิสก์ของเรามีขนาด 1024 ไบต์/บล็อก สามารถเก็บได้ 64 ระเบียน/บล็อก แฟ้มที่มี 120,000 ระเบียนควรจะมี 2,000 บล็อก (2 ล้านไบต์) ดัชนีควรมี 2,000 ช่อง ของทุก ๆ 10 หลักหรือ 20,000 ไบต์ แล้วควรเก็บไว้ในหน่วยความจำ ในการค้นหาราคาของสินค้าเราสามารถค้นหาดัชนีในการค้นหานี้เราควรจะรู้ว่าบล็อกไหนบรรจุระเบียนที่ต้องการแล้วจึงเข้าถึงบล็อกนั้น โครงสร้างข้อมูลแบบนี้ทำให้เราสามารถค้นหาแฟ้มข้อมูลขนาดใหญ่ได้ด้วยการทำ I/O เพียงนิดเดียว

ถ้าแฟ้มข้อมูลมีขนาดใหญ่ แฟ้มดัชนีก็ต้องมีขนาดใหญ่เกินที่จะเก็บไว้ในหน่วยความจำได้ วิธีแก้วิธีหนึ่ง คือ สร้างดัชนีสำหรับแฟ้มดัชนี (Index file) แฟ้มข้อมูลปฐมภูมิ (Primary index file) ควรจะบรรจุตัวชี้ไปยังแฟ้มข้อมูลทุติยภูมิ (Secondary index file) ซึ่งควรจะชี้ไปยังข้อมูลจริง

ตัวอย่าง ดัชนีแบบเรียงลำดับของ IBM indexed sequential-access เป็นวิธีการเข้าถึงของ IBM (ISAM) ใช้ Master index เล็ก ๆ ซึ่งชี้ไปยังบล็อกของดิสก์ของดัชนีทุติยภูมิ (Secondary index) ดัชนีทุติยภูมิจะชี้ไปยังบล็อกของแฟ้มข้อมูลจริง แฟ้มข้อมูลถูกเก็บแบบเรียงลำดับโดยการกำหนดคีย์ขึ้นมาเพื่อหาข้อมูลที่ต้องการ ขั้นแรกเราต้องค้นหาแบบ Binary search เพื่อหา Master index เพื่อจัดเตรียมหมายเลขบล็อกของดัชนีทุติยภูมิ บล็อกถูกอ่านแล้วค้นหาแบบ Binary search อีกครั้ง

เพื่อหาบล็อกที่บรรจุระเบียนที่ต้องการขั้นสุดท้าย บล็อกนี้จะถูกค้นหาแบบเรียงลำดับ ด้วยวิธีการนี้ระเบียนต่าง ๆ สามารถระบุตำแหน่งจากคีย์ของมัน โดยการอ่านแบบโดยตรงทั้ง 2 ครั้ง ภาพที่ 9.4 แสดงสถานการณ์คล้าย ๆ กัน ซึ่งนำไปใช้โดยดัชนีและแฟ้มข้อมูลแบบสัมพันธ์ของ VMS index และ Relative files



ภาพที่ 9.4 ตัวอย่างของแฟ้มดัชนีและแฟ้มข้อมูลแบบสัมพันธ์
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.516)

9.3 โครงสร้างของไดเรกทอรี่

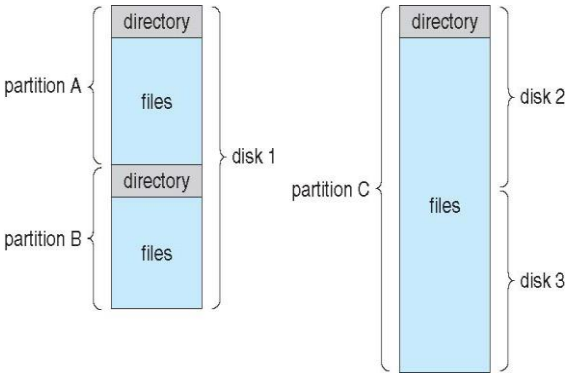
ไดเรกทอรี่ (Directories) เป็นที่เก็บรวบรวมชื่อของแฟ้มข้อมูล และข้อมูลที่สำคัญของแฟ้มข้อมูล โดยมีวัตถุประสงค์สำคัญคือ เพิ่มประสิทธิภาพการทำงานของระบบ และความสะดวกสำหรับผู้ใช้ ด้วยการช่วยให้ผู้ใช้สามารถเข้าถึงแฟ้มข้อมูลที่ต้องการได้อย่างรวดเร็วมากขึ้น รวมถึงช่วยให้ผู้ใช้สะดวกสบายในการใช้แฟ้มข้อมูล ทั้งการตั้งชื่อแฟ้มข้อมูล การจัดกลุ่มแฟ้มข้อมูลตามความต้องการของผู้ใช้

ระบบแฟ้มข้อมูลของคอมพิวเตอร์นั้นยังสามารถขยายออกไปได้อีกในบางระบบ บรรจุแฟ้มข้อมูลไว้เป็นล้านแฟ้มข้อมูลในดิสก์ขนาดเทราไบต์ (Terabyte) การที่จะจัดการกับข้อมูลทั้งหมดต้องมีวิธีการจัดการที่ดี วิธีการจัดการนั้นจะเกี่ยวกับการใช้ไดเรกทอรี่ ดังนั้นเราจะอธิบายโครงสร้างพื้นฐานบางอย่างของโครงสร้างไดเรกทอรี่ ที่เป็นจุดเด่นสำหรับการเก็บข้อมูล ดังนั้นสามารถอธิบายโครงสร้างพื้นฐานบางอย่างของโครงสร้างไดเรกทอรี่ ที่เป็นจุดเด่นสำหรับการเก็บข้อมูล

ในหนึ่งดิสก์ สามารถใช้ระบบแฟ้มข้อมูลได้หลาย ๆ ระบบบนดิสก์เดียว หรือจะแบ่งดิสก์ออกเป็นส่วน ๆ สำหรับแต่ละระบบแฟ้มข้อมูล ระบบแฟ้มข้อมูลแบ่งออกเป็นพาร์ติชั่น (Partitions) ใน IBM เรียก minidisks ในเครื่อง PC และ Macintosh เรียก Volume โดยทั่วไป แต่ละระบบบนดิสก์ถูกบรรจุอย่างน้อย 1 พาร์ติชั่น ผู้ใช้จำเป็นต้องเกี่ยวข้องกับ Logical directory และโครงสร้างของแฟ้มข้อมูล และสามารถมองข้ามปัญหาทางกายภาพในการจัดการพื้นที่ว่างของแฟ้มข้อมูล ด้วยเหตุนี้พาร์ติชั่นสามารถมองเป็นดิสก์เสมือนได้ (Virtual disk)

ในแต่ละ Volume สามารถแบ่งออกมาเพื่อเป็นดิสก์ได้ Volumes สามารถมีระบบได้หลายระบบปฏิบัติการ และอนุญาตให้บูทหรือรันได้มากกว่าหนึ่ง ในแต่ละ Volumes ที่มีระบบแฟ้มข้อมูลอยู่

จะต้องเก็บข้อมูลของระบบแฟ้มข้อมูลเอาไว้ด้วย ในข้อมูลนั้นจะเก็บ Device directory (หรือ Volume table of contents device directory) คือไดเรกทอรี่ที่บรรจุไปด้วย ชื่อ ที่อยู่ ขนาด และชนิดของแฟ้มข้อมูลทั้งหมดบน volumes ในภาพที่ 9.5



ภาพที่ 9.5 ตัวอย่างการจัดระเบียบของระบบแฟ้มข้อมูล
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.516)

เมื่อพิจารณาเข้าไปในโครงสร้างของไดเรกทอรี่โดยเฉพาะแล้ว การนำไดเรกทอรี่ไปใช้มีวิธีในการดำเนินการดังนี้

1. ค้นหาแฟ้มข้อมูล (Search for a file) เมื่อผู้ใช้หรือโปรแกรมเรียกใช้แฟ้มข้อมูลใด ๆ ระบบต้องค้นหาแฟ้มข้อมูลนั้นจากไดเรกทอรี่

2. สร้างแฟ้มข้อมูล (Create a file) แฟ้มข้อมูลใหม่จำเป็นต้องถูกสร้างและเพิ่มเข้าในไดเรกทอรี่

3. ลบแฟ้มข้อมูล (delete a file) เมื่อแฟ้มข้อมูลไม่ต้องการแล้ว ต้องสามารถลบมันออกจากไดเรกทอรี่

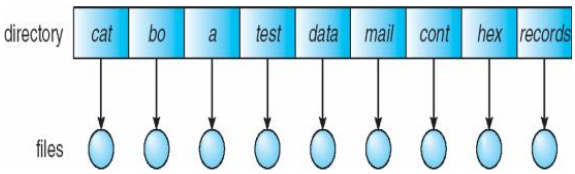
4. แสดงไดเรกทอรี่ (List a directory) การดูแฟ้มข้อมูลในไดเรกทอรี่ ต้องสามารถแสดงชื่อแฟ้มข้อมูลในไดเรกทอรี่ และเนื้อหาของไดเรกทอรี่สำหรับแต่ละแฟ้มในรายการ

5. เปลี่ยนชื่อแฟ้มข้อมูล (Rename a file) เพราะว่าชื่อของแฟ้มข้อมูลนั้นมีหลากหลายเนื้อหาและหลากหลายผู้ใช้ เราต้องสามารถเปลี่ยนชื่อ เมื่อเนื้อหาภายในถูกเปลี่ยนหรือมีการใช้แฟ้มข้อมูลนั้นเปลี่ยนแปลงไป

6. การข้ามระบบแฟ้มข้อมูล (Traverse the file system) การสามารถเข้าถึงในหลายไดเรกทอรี่และหลาย ๆ แฟ้มข้อมูลภายในโครงสร้างไดเรกทอรี่ สำหรับความน่าเชื่อถือ คือ การรักษาเนื้อหาและโครงสร้างของระบบแฟ้มข้อมูลทั้งหมด การรักษานี้มักทำได้โดยการคัดลอกแฟ้มข้อมูลทั้งหมดลงเทปแม่เหล็ก เทคนิคนี้เป็นการ Backup ในกรณีที่ระบบล่มหรือแฟ้มข้อมูลเสีย ในกรณีนี้แฟ้มข้อมูลควรถูกคัดลอกลงเทป และพื้นที่ว่างในดิสก์ของแฟ้มเหล่านั้นก็จะถูกปล่อยให้แฟ้มอื่นได้ใช้ต่อไป

9.3.1 ไดเรกทอรี่ระดับเดียว (Single-Level Directory)

โครงสร้างต่าง ๆ ของไดเรกทอรี่ คือ ไดเรกทอรี่ระดับเดียว แฟ้มข้อมูลทั้งหมดจะถูกเก็บไว้ในไดเรกทอรี่เดียว เป็นโครงสร้างไดเรกทอรี่ที่ง่ายต่อการศึกษาและเข้าใจดังแสดงในภาพที่ 9.6



ภาพที่ 9.6 โครงสร้างไดเรกทอรี่ระดับเดียว
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.519)

ไดเรกทอรี่ระดับเดียวมีความสำคัญอยู่ที่ข้อกำจัดของมัน อย่างไรก็ตามเมื่อตัวเลขของแฟ้มข้อมูลเพิ่มขึ้นหรือระบบมีมากกว่าหนึ่ง ผู้ใช้แฟ้มข้อมูลทั้งหมดที่อยู่ในไดเรกทอรี่เดียวกัน จะต้องมีชื่อไม่เหมือนกัน ถ้าผู้ใช้เรียกแฟ้มข้อมูลที่มีชื่อเดียวกันขึ้นมามันจะเป็นการฝ่าฝืนกฎที่แฟ้มข้อมูลต้องเป็นชื่อเฉพาะแฟ้มข้อมูลนั้น (Unique-name)

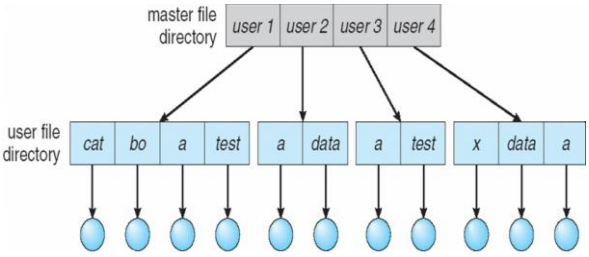
ตัวอย่าง ในห้องเรียนวิชาการเขียนโปรแกรมหนึ่งนั้นมีนักเรียน 23 คน ต้องการเรียกงานชื่อ prog2 อีก 11 คนเรียกโปรแกรม assign2 ถึงแม้ว่าแฟ้มข้อมูลที่ถูกเลือกโดยทั่วไปนั้นจะแสดงให้เห็นเนื้อหาของแฟ้มข้อมูล บ่อยครั้งที่มีเรื่องข้อจำกัดของความยาวของชื่อแฟ้มข้อมูล และยากที่จะทำให้ชื่อแฟ้มข้อมูลไม่ซ้ำกัน ระบบปฏิบัติการ MS-DOS ใช้ตัวอักษรในการตั้งชื่อแฟ้มข้อมูลความยาวทั้งหมดไม่เกิน 11 ตัวอักษร ขณะที่ระบบปฏิบัติการ UNIX ใช้ตัวอักษรไม่เกิน 255 ตัวอักษร

ยิ่งไปกว่านั้นผู้ใช้เพียงคนเดียวบนระบบปฏิบัติการ Single-level directory อาจจะเป็นเรื่องยากที่จะค้นหา จดจำแฟ้มข้อมูลทั้งหมดรวมไปถึงจำนวนแฟ้มข้อมูลที่เพิ่มขึ้นมาที่มีผู้ใช้จำนวนไม่น้อยที่มีแฟ้มข้อมูลเป็นร้อยบนเครื่องคอมพิวเตอร์เครื่องเดียว และเท่ากับจำนวนตัวเลขแฟ้มข้อมูลบนระบบอื่น มันคงจะเป็นเรื่องที่น่ากลัวมากที่เราจะเก็บเส้นทางทั้งหมดของแฟ้มข้อมูลไว้

9.3.2 ไดเรกทอรี่สองระดับ (Two-Level Directory)

พวกเราได้เห็นมาแล้วว่าไดเรกทอรี่ระดับเดียว บ่อยครั้งที่ทำให้สับสนชื่อแฟ้มข้อมูลระหว่างผู้ใช้หลายคน วิธีการมาตรฐานที่จะมาแก้เรื่องนี้ที่จะมาแบ่งไดเรกทอรี่สำหรับผู้ใช้แต่ละคน ในโครงสร้างของไดเรกทอรี่สองระดับ แต่ละผู้ใช้จะมีไดเรกทอรี่ของตนเอง (User file directory : UFD) โดยจะสร้างโครงสร้างจำลองขึ้นมาสำหรับผู้ใช้เพียงคนเดียว เมื่อผู้ใช้เริ่มใช้งานระบบหลัก (Master file directory : MFD) ถึงจะทำการค้นหา MFD จะเป็นตัวดัชนีสำหรับผู้ใช้ และแต่ละจุดจะเป็นที่ไปยังผู้ใช้แต่ละคน

เมื่อผู้ใช้อ้างอิงแฟ้มข้อมูลจะมีเพียง UFD เดียวที่จะถูกค้นหา ดังนั้นผู้ใช้หลาย ๆ คนถึงจะมีแฟ้มข้อมูลชื่อเหมือนกันได้ ตราบเท่าที่ยังมีแฟ้มข้อมูลทั้งหมดที่อยู่ในแต่ละ UFD ที่เหมือนกัน ที่จะสร้างแฟ้มข้อมูลสำหรับผู้ใช้ระบบปฏิบัติการจะทำการค้นหาว่าแฟ้มข้อมูลนั้นมีอยู่แล้วหรือไม่



ภาพที่ 9.7 โครงสร้างของไดเรกทอรี่สองระดับ
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.519)

การระบุชื่อแฟ้มข้อมูลในไดเรกทอรี่สองระดับ เราต้องให้ทั้งชื่อของผู้ใช้และชื่อแฟ้มข้อมูล ไดเรกทอรี่สองระดับสามารถมองเป็นต้นไม้ ราก (Root) ของต้นไม้คือ MFD ทายาทโดยตรงคือ UFD ทายาทของ UFD คือ แฟ้มข้อมูลของตัวมันเอง แฟ้มข้อมูลคือ ใบไม้ของต้นไม้นั่นเอง ชื่อของผู้ใช้และชื่อของแฟ้มข้อมูลกำหนดเส้นทาง (Path) ในต้นไม้จากราก (MFD) สู่ใบ (แฟ้มที่ต้องการ) ดังนั้นชื่อของผู้ใช้และชื่อไฟล์กำหนดชื่อของเส้นทาง (Path name) แฟ้มข้อมูลทุกแฟ้มในระบบมีชื่อของเส้นทาง เพื่อระบุชื่อแฟ้มข้อมูลที่มีลักษณะเฉพาะ ผู้ใช้ต้องรู้ชื่อของเส้นทางของแฟ้มที่ต้องการ

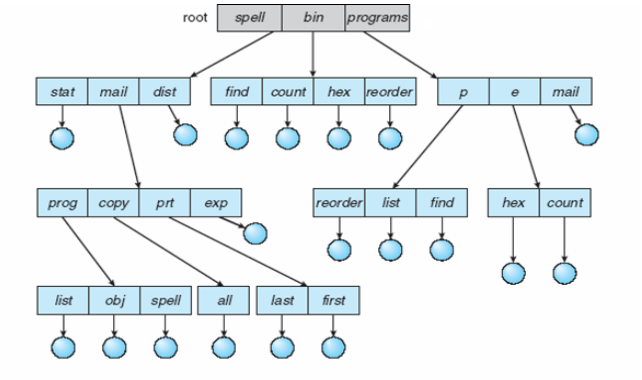
การลบ ระบบปฏิบัติการจะกำหนดขอบเขตการค้นหาอยู่ในแต่ละ UFD ฉะนั้นมันจะไม่สามารถจะไปลบแฟ้มข้อมูลที่ชื่อเหมือนกันของผู้ใช้คนอื่นได้

ตัวอย่าง ในระบบ MS-DOS อาจเป็น “C:\userb\test” แยกเป็น partition ชื่อของไดเรกทอรี่ และชื่อแฟ้มข้อมูลในระบบ VMS ไฟล์ “login.com” อาจเป็น “u:[sst.jdeck]login.com;1”

- “u” คือชื่อของ partition
- “sst” คือ ชื่อไดเรกทอรี่
- “jdeck” คือ ชื่อของไดเรกทอรี่ย่อย (Subdirectory)
- “1” คือ หมายเลขเวอร์ชัน

9.3.3 ไดเรกทอรี่ที่มีโครงสร้างแบบต้นไม้ (Tree-structured directory)

ในการใช้งานปกติ ผู้ใช้แต่ละคนจะมีไดเรกทอรี่ปัจจุบัน (Current directory) ซึ่งใช้เก็บแฟ้มข้อมูลที่ผู้ใช้สนใจในปัจจุบันหรือเป็นตำแหน่งที่กำลังใช้งานอยู่ เมื่อมีการอ้างอิงแฟ้มข้อมูลก็จะเกิดการค้นหาไดเรกทอรี่ปัจจุบัน ถ้าแฟ้มข้อมูลที่ต้องการไม่มีในไดเรกทอรี่ปัจจุบัน ผู้ใช้ต้องระบุชื่อของเส้นทาง (Path name) หรือเปลี่ยนไดเรกทอรี่ปัจจุบันไปเป็นไดเรกทอรี่อื่น โปรแกรมเรียกระบบจะทำให้ชื่อไดเรกทอรี่เป็นเหมือนพารามิเตอร์และใช้มัน เพื่อกำหนดไดเรกทอรี่ปัจจุบันใหม่ ดังนั้นผู้ใช้จึงสามารถเปลี่ยนไดเรกทอรี่ปัจจุบันของผู้ใช้ได้ตามต้องการ



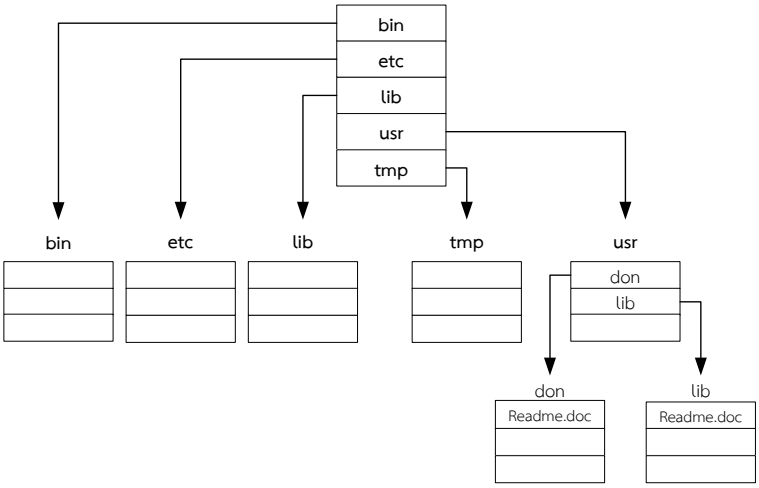
ภาพที่ 9.8 โครงสร้างของไดเรกทอรี่แบบต้นไม้
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.521)

การอ้างอิงถึงแฟ้มข้อมูลใด ๆ ก็ตาม จำเป็นต้องระบุที่อยู่ของแฟ้มข้อมูลนั้น ๆ ให้ถูกต้องว่าอยู่ในไดเรกทอรี่ใด หรือสับไดเรกทอรี่ใด ในการกำหนดที่อยู่หรือเส้นทางที่จะเข้าถึงแฟ้มข้อมูลนั้น ๆ เรียกว่า พาธ (Path) ดังนั้นถ้าต้องการอ้างถึงแฟ้มข้อมูลใด ๆ ในดิสก์ จำเป็นต้องระบุพาธให้ถูกต้องพร้อมชื่อแฟ้มข้อมูลวิธีการอ้างถึงชื่อแฟ้มข้อมูลดังนี้

1. การอ้างชื่อแฟ้มข้อมูลแบบสัมบูรณ์ (Absolute path name)

เป็นการอ้างถึงแฟ้มข้อมูลโดยเริ่มจากราก (Root) เสมอ ตามด้วยชื่อไดเรกทอรี่ย่อยไล่ลงมาตามลำดับชั้นของไดเรกทอรี่จนกระทั่งถึงไดเร็กทอรีที่บรรจุแฟ้มข้อมูลอยู่ และจบลงด้วยชื่อแฟ้มข้อมูลนั้น ๆ ตัวอย่างการอ้างถึงแฟ้มข้อมูลชื่อ Readme.doc ที่อยู่ภายใต้พาธ Root ไดเรกทอรี่ user และไดเรกทอรี่ lib

1) Windows หรือ MS-DOS	\user\lib\readme.doc
2) UNIX หรือ Linux	/user/lib/readme.doc



ภาพที่ 9.9 แบบไดเรกทอรี่ของ UNIX หรือ Linux

2. การอ้างชื่อแบบสัมพัทธ์ (Relative path name)

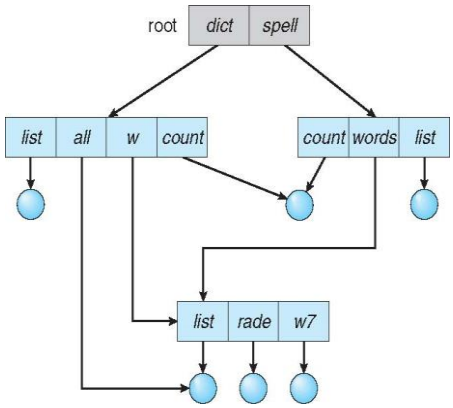
เป็นการอ้างถึงแฟ้มข้อมูลโดยที่ผู้ใช้จะต้องเข้าใจในเรื่องระบบไดเรกทอรี่ปัจจุบัน (Current directory) เนื่องจากการอ้างถึงชื่อแฟ้มข้อมูลจะเริ่มต้นจากไดเรกทอรี่ปัจจุบันแล้วไล่ไปตามลำดับชั้นของไดเรกทอรี่ที่แฟ้มข้อมูลนั้นอยู่และจบลงด้วยชื่อแฟ้มข้อมูลนั้น ตัวอย่างการอ้างถึงแฟ้มข้อมูลชื่อ Readme.doc ที่อยู่ภายใต้พาธ Root ไดเรกทอรี่ user และไดเรกทอรี่ lib โดยที่ไดเรกทอรี่ปัจจุบันอยู่ที่ไดเรกทอรี่ user

1) ระบบปฏิบัติการ Windows หรือ MS-DOS	\lib\readme.doc
2) ระบบปฏิบัติการ UNIX หรือ Linux	/lib/readme.doc

9.3.4 ไดเรกทอรี่กราฟแบบไม่เป็นวงจร (Acyclic-Graph Directory)

เมื่อพิจารณาเหตุการณ์โปรแกรมเมอร์สองคนที่ทำโครงงานร่วมกัน แฟ้มข้อมูลงานจะถูกเก็บในไดเรกทอรี่ย่อยที่แยกมาจากโครงการอื่น และโปรแกรมเมอร์ทั้งสองคนมีการตอบสนองต่อโครงงานเท่า ๆ กัน ทั้งสองคนต้องการมีโครงงานย่อยของตัวเอง ซึ่งไดเรกทอรี่ย่อยนี้ควรจะถูกใช้ร่วมกัน โดยปกติการใช้แฟ้มข้อมูลหรือไดเรกทอรี่ร่วมกันในระบบอาจจะมี 2 หรือมากกว่าต่อหนึ่งการใช้ร่วมกัน

การเปลี่ยนแปลงที่ทำโดยผู้ใช้คนหนึ่งควรจะทำให้คนอื่นเห็นความเปลี่ยนแปลงนี้ได้ในทันที แฟ้มข้อมูลใหม่ที่ถูกสร้างโดยผู้ใช้คนหนึ่ง จะต้องปรากฏในไดเรกทอรี่ย่อยที่ใช้ร่วมกันทันทีโดยอัตโนมัติ ข้อห้ามของการใช้แฟ้มข้อมูลหรือไดเรกทอรี่ร่วมกัน โครงสร้างแบบต้นไม้ห้ามให้มีการใช้แฟ้มข้อมูลหรือไดเรกทอรี่ร่วมกัน กราฟแบบไม่เป็นวงจร (Acyclic graph) อนุญาตให้มีการใช้ไดเรกทอรี่และแฟ้มข้อมูลร่วมกันได้ (ดังภาพที่ 9.10)



ภาพที่ 9.10 โครงสร้างของไดเรกทอรี่กราฟแบบไม่เป็นวงจร
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.523)

การใช้แฟ้มข้อมูลและไดเรกทอรี่ร่วมกันนั้นในทางปฏิบัติแล้วสามารถทำได้หลายวิธี ยกตัวอย่างในระบบ Unix การสร้าง Directory entry นั้นเรียกว่า Link โดยเป็นการใช้พอนย์เตอร์ชี้ไปยังแฟ้มข้อมูลหรือ

ไดเรกทอรี่ย่อย ตัวอย่างเช่น Link อาจเอาไปใช้ในชื่อของเส้นทางแบบสัมบูรณ์ (เรียกว่า Symbolic link) เมื่อเกิดการอ้างอิงแฟ้มข้อมูล เราจะค้นหาไดเรกทอรี่ถ้าไดเรกทอรี่ถูกทำเครื่องหมายเป็น Link ก็จะได้มาซึ่งชื่อของแฟ้มข้อมูลหรือไดเรกทอรี่จริง ๆ

อีกวิธีหนึ่งในการใช้แฟ้มข้อมูลร่วมกัน คือ การคัดลอกสารสนเทศทั้งหมดไปไว้ในไดเรกทอรี่ที่ใช้ร่วมกันทั้งคู่ ดังนั้นไดเรกทอรี่ทั้งคู่ต้องเหมือนกันและเท่ากัน แต่ Link จะแตกต่างกันอย่างชัดเจนจากไดเรกทอรี่ต้นฉบับ ดังนั้นไดเรกทอรี่ทั้งสองจึงไม่เท่ากัน โดยไดเรกทอรี่ต้นฉบับไม่เท่ากับตัวสำเนาที่ใช้ร่วมกัน การคัดลอกไดเรกทอรี่ทำให้ต้นฉบับและตัวสำเนาสามารถแยกจากกันได้

Acyclic graph สามารถปรับให้เข้ากับสถานการณ์ โดยการสร้างต้นไม้แบบง่าย ๆ ได้ แต่เป็นเรื่องที่ยุ่งยากแฟ้มข้อมูลอาจจะมีหลาย Path name ดังนั้นมันแตกต่างอย่างชัดเจนที่จะอ้างถึงแฟ้มข้อมูลเดียวกัน ถ้าเราต้องการค้นหาแฟ้มข้อมูล เก็บสะสมสถิติของแฟ้มข้อมูลทั้งหมดหรือสำรองแฟ้มข้อมูลทั้งหมดเอาไว้ พวกเราไม่สามารถท่องไปในโครงสร้างแบบแชร์ได้มากกว่าหนึ่งครั้ง

อีกปัญหาหนึ่งที่พบคือถ้ามีพื้นที่ว่างในการแชร์แฟ้มข้อมูล เมื่อต้องการพื้นที่ตรงนั้นมาใช้ใหม่ เมื่อได้ทำการลบแฟ้มข้อมูลนั้นไปแล้ว พอนย์เตอร์ยังคงอยู่และมันถูกแขวนไว้โดยไม่ได้อ้างอิงแฟ้มข้อมูลไหนในระบบที่ใช้การแชร์แบบ Link ในสถานการณ์แบบนี้เป็นเรื่องที่ง่ายมาก Link จะหายไปก็ต่อเมื่อแฟ้มข้อมูลนั้นถูกลบออกไป และผู้ใช้จะต้องเข้าใจว่าแฟ้มข้อมูลนั้นถูกลบออกไปแล้วหรือถูกแทนที่ Microsoft windows ก็ใช้วิธีนี้เหมือนกัน

ปัญหาการลบนั้นยังคงอยู่ จนกระทั่งจุดอ้างอิงทั้งหมดถูกลบทิ้งไป อาจจำเป็นจะต้องมีกลไกมากำหนดเมื่อการอ้างอิงครั้งสุดท้ายถูกลบไป การเก็บรายการของการอ้างอิงทั้งหมดไว้เมื่อมีการสร้าง Link ขึ้นมาใหม่ก็จะถูกเพิ่มเข้าไปในรายการเมื่อมีการลบรายการ Link ก็จะถูกลบทิ้งไปด้วย แฟ้มข้อมูลนั้นจะถูกลบทิ้งเมื่อแฟ้มข้อมูล reference นั้นว่างเปล่า

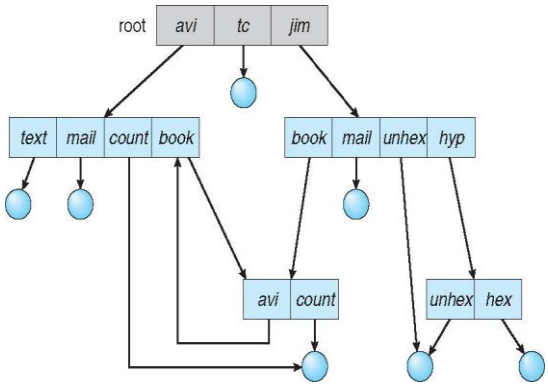
สิ่งที่น่าหนักใจก็คือแฟ้มข้อมูลขนาดใหญ่ การอ้างอิงสามารถแก้ไขได้โดยการเก็บเฉพาะจำนวนเท่านั้น ถ้าจำนวนของแฟ้มข้อมูลนั้นเป็น 0 แฟ้มข้อมูลนั้นคือแฟ้มข้อมูลที่ถูกลบไปแล้ว ในการเลี่ยงปัญหาการแชร์ บางระบบไม่อนุญาตให้แชร์ Link หรือไดเรกทอรี่ เช่น MS-DOS โครงสร้างของไดเรกทอรี่ คือต้นไม้ที่มากกว่า Acyclic graph

9.3.5 ไดเรกทอรี่แบบกราฟโดยทั่วไป (General Graph Directory)

ปัญหาหลักของการใช้ Acyclic graph คือ มันไม่ใช่โครงสร้างที่สมบูรณ์แบบ ถ้าเราเริ่มจาก Two-level และอนุญาตให้ผู้ใช้สร้างไดเรกทอรี่ย่อย ผลก็คือจะกลายเป็นโครงสร้างแบบ Tree-structure ถึงเราจะเพิ่มแฟ้มข้อมูลและไดเรกทอรี่ย่อยอีก ก็ยังเป็นไดเรกทอรี่แบบต้นไม้อยู่ดี แต่ถ้าเราเพิ่มการเชื่อมโยงของไดเรกทอรี่ที่มีอยู่แล้ว โครงสร้างแบบต้นไม้ก็จะเสียไป เป็นผลให้เกิดโครงสร้างแบบกราฟอย่างง่าย

ความได้เปรียบของ Acyclic graph คือ ความสัมพันธ์ที่มีความเรียบง่ายของอัลกอริทึมที่จะท่องไปในกราฟ การค้นหาแฟ้มข้อมูลในไดเรกทอรี่ที่ใช้ร่วมกัน (Share subdirectory) แล้วไม่พบในครั้งแรกควรที่จะเลี่ยงการท่องไปในการแชร์ของพื้นที่ที่แบ่งไว้ของ Acyclic graph เป็นครั้งที่สอง เหตุผลก็เพื่อประสิทธิภาพเป็นหลัก ถ้ามีการค้นหาแบบเฉพาะเจาะจงโดยไม่ต้องทำการค้นหา จำเป็นต้องการเลี่ยงการค้นหาอีกครั้งหนึ่ง การค้นหาครั้งที่สองทำให้เราเสียเวลามาก

ถ้าให้เกิดความสมบูรณ์แล้วการอนุญาตให้มีวงจรในไดเรกทอรี่ คือต้องการหลีกเลี่ยงการค้นหาข้อมูลซ้ำเป็นครั้งที่สอง ถ้าออกแบบอัลกอริทึมไม่ดีจะทำให้เกิดการวนซ้ำ (Loop) ไม่รู้จบ การค้นหาในวงจรก็จะไม่มีวันจบสิ้น วิธีแก้คือจำกัดจำนวนของไดเรกทอรี่ที่จะเข้าใช้ในระหว่างการค้นหา



ภาพที่ 9.11 ไดเรกทอรี่แบบกราฟโดยทั่วไป
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.525)

การเก็บรวบรวมขยะเป็นสิ่งที่จำเป็น เนื่องจากเป็นสิ่งที่เป็นไปได้ในการเกิดวัฏจักรที่แสดงอยู่ในกราฟ ดังนั้นกราฟที่มีโครงสร้างไม่เป็นวัฏจักรจะง่ายในการนำไปใช้ เป็นเรื่องที่ยากในการหลีกเลี่ยงการใช้วัฏจักรเข้าไปเชื่อมต่อในโครงสร้างที่มีอยู่ อย่างไรก็ตามเราสามารถทราบได้ว่าวัฏจักรนั้นจะเสร็จสมบูรณ์ได้เมื่อไร ในกระบวนการขั้นตอนในการลบวัฏจักรในกราฟที่แสดงออกมา จะต้องใช้กระบวนการในการใช้งานที่สิ้นเปลือง

โดยเฉพาะอย่างยิ่งเมื่อเราจะทำการจัดเก็บกราฟลงในดิสก์ เป็นกระบวนการคิดที่ง่ายในกรณีพิเศษที่ใช้กับไดเรกทอรี่ และสามารถเชื่อมโยงข้ามไดเรกทอรี่ที่มีการกีดขวางได้ วัฏจักรที่จะหลีกเลี่ยงและไม่ใช้กรณีพิเศษจะมีตำแหน่งที่จะเกิดข้อผิดพลาด มีอัลกอริทึมในการค้นหาวงจรในกราฟแต่ใช้เวลาในการค้นหามาก ดังนั้นโดยทั่วไปโครงสร้างไดเรกทอรี่แบบต้นไม้น่าใช้กว่าโครงสร้างของกราฟแบบไม่มีวงจร

9.4 การป้องกันการสูญหายของข้อมูล

เมื่อข้อมูลถูกเก็บไว้ในระบบคอมพิวเตอร์ สิ่งที่เราต้องการคือความปลอดภัยทางด้านกายภาพและความปลอดภัยทางด้านการเข้าถึงข้อมูล ความเชื่อถือของการเกิดความซ้ำซ้อนของสำเนาแฟ้มข้อมูลคอมพิวเตอร์ในทุกวันนี้มีระบบหรือโปรแกรมที่ทำงานโดยอัตโนมัติอยู่แล้ว การคัดลอกแฟ้มข้อมูลระหว่างดิสก์นั้น เราสามารถทำได้เป็นระยะ ดังนั้นควรมีการแบ็คอัพข้อมูลเอาไว้เพื่อป้องกันการที่แฟ้มข้อมูลถูกทำลายโดยบังเอิญ แฟ้มข้อมูลในระบบนั้นสามารถที่จะรองรับความเสียหายจากปัญหาทางด้านฮาร์ดแวร์ได้ (เช่น ข้อผิดพลาดในการอ่านและเขียน) ไม่มีไฟเลี้ยง หรือเกิดความผิดพลาดล้มเหลวจากการที่หัวอ่านข้อมูลสกปรก อุณหภูมิภายในเครื่องสูง หรือจากปัญหาอื่น ๆ ข้อผิดพลาดที่อยู่ในระบบแฟ้มข้อมูล ยังสามารถทำให้แฟ้มข้อมูลสูญหายได้

ดังนั้นการป้องกันการสูญหายของข้อมูลจึงเป็นเรื่องสำคัญ ซึ่งอาจจะไม่เห็นความสำคัญในขณะที่ใช้เครื่องเพียงคนเดียวหรือไม่ได้มีการติดต่อกับเครื่องอื่น ๆ แต่มันจะสำคัญมากขึ้น เมื่อมีการติดต่อระหว่างกันมีการแชร์แฟ้มข้อมูลระหว่างกัน การป้องกันข้อมูลจึงมีความสำคัญมากขึ้น

9.4.1 ชนิดของการเข้าถึงแฟ้มข้อมูล (Types of Access)

ความจำเป็นในการป้องกันแฟ้มข้อมูลนั้น เป็นผลมาจากการที่มีการแชร์แฟ้มข้อมูลแล้วสามารถเข้าถึงแฟ้มข้อมูลนั้นได้ เมื่อมีแฟ้มข้อมูลที่ผู้ใช้ไม่ต้องการเปิดเผยจำเป็นต้องมีการป้องกันเพื่อไม่ให้ผู้อื่นเข้ามาใช้งาน เมื่อเป็นเช่นนี้เราจึงต้องมีการป้องกันข้อมูลที่สมบูรณ์ โดยการไม่ให้เข้าถึงข้อมูลหรือก็คือให้เราเข้าถึงแฟ้มข้อมูลเพียงคนเดียว ทั้งสองวิธีนั้นเป็นวิธีการป้องกันที่ดีสำหรับการใช้งานทั่วไป และอะไรคือการควบคุมการเข้าถึงการใช้งาน

การป้องกันโดยการควบคุมการเข้าถึงของข้อมูลโดยการกำหนดสิทธ์ของผู้ใช้ว่ามีประเภทเป็นอะไร ซึ่งเราสามารถกำหนดมันได้ โดยเครื่องแม่ข่ายจะมีประเภทในการกำหนดชนิดดังนี้

- อ่าน (Read) อ่านข้อมูลจากแฟ้มข้อมูล
- เขียน (Write) เขียนหรือเขียนซ้ำแฟ้มข้อมูล
- ดำเนินการ (Execute) อ่านแฟ้มข้อมูลแล้วเอาเข้าไปในหน่วยความจำแล้วดำเนินการกับแฟ้มข้อมูลนั้น
- เพิ่ม (Append) เขียนข้อมูลใหม่จนกระทั่งจบแฟ้มข้อมูล
- ลบ (Delete) ลบแฟ้มข้อมูลออกไปทำให้เกิดพื้นที่ว่าง
- รายการ (List) แสดงชื่อและส่วนประกอบต่าง ๆ ของแฟ้มข้อมูล

การดำเนินการอื่น ๆ เช่น การคัดลอก เปลี่ยนชื่อ และแก้ไขแฟ้มข้อมูล ระบบสามารถดำเนินการได้ทั้งหมด สำหรับชุดคำสั่งที่มีจำนวนมากและซับซ้อนเหล่านี้จะทำงานในระดับที่สูงขึ้นไป โดยจะดำเนินการด้วยระบบโปรแกรมที่จะทำให้เครื่องคอมพิวเตอร์สามารถลดการใช้ทรัพยากรลงไปได้

ดังนั้นเมื่อสารสนเทศถูกเก็บเข้าไว้ในระบบคอมพิวเตอร์ สิ่งที่ต้องทำคือการป้องกันข้อมูลจากความเสียหายทางกายภาพ (ความไว้ใจได้) และการเข้าถึงข้อมูลอย่างไม่เหมาะสม (การป้องกัน)

9.4.2 รายการเข้าถึงแฟ้มข้อมูลและกลุ่ม (Access Lists and Groups)

ส่วนใหญ่วิธีในการป้องกันปัญหานั้นจะขึ้นอยู่กับข้อมูลเฉพาะของตัวผู้ใช้ ผู้ใช้ที่แตกต่างกันอาจจะต้องกำหนดสิทธิ์เพื่อให้เข้าไปใช้แฟ้มข้อมูลหรือไดเรกทอรี่ที่กำหนดไว้เท่านั้น โดยทั่วไปนั้นการเชื่อมโยงของแฟ้มข้อมูลในแต่ละไดเรกทอรี่จะใช้วิธีการเข้าถึงโดยการควบคุม (Access-control list : ACL) ซึ่งวิธีการนี้ต้องระบุถึงชื่อผู้ใช้ และประเภทของการเข้าถึงจะอนุญาตให้ใช้ได้เฉพาะรายเท่านั้น ACL นำมาใช้ในการกำหนดสิทธิ์การเข้าถึงฟังก์ชั่นการทำงานต่าง ๆ ในการกำหนดสิทธิ์นี้จะแบ่งการกำหนดค่าต่าง ๆ ออกเป็นสองส่วนหลัก ๆ คือ

1. ส่วนของ AROs (Access Request Objects) ตรงนี้จะเป็นการกำหนด Object ที่ทำการร้องขอ (Request) ฟังก์ชั่นการทำงาน (Action) ซึ่งโดยส่วนใหญ่แล้วจะหมายถึง กลุ่มของผู้ใช้ หรือตัวผู้ใช้เอง สามารถกำหนดเป็น n-level ได้ หมายถึง กลุ่มของผู้ใช้หรือผู้ใช้ จะสามารถแบ่งเป็นระดับย่อย ๆ ได้หลายระดับ

2. ส่วนของ ACOs (Access Control Objects) ตรงนี้จะใช้กำหนดทรัพยากรต่าง ๆ ในระบบของเราที่ต้องการจะกำหนดสิทธิ์ให้กับผู้ใช้บางกลุ่มหรือบางคนในการเข้าถึงทรัพยากรต่าง ๆ ซึ่งส่วนนี้สามารถกำหนดเป็นระดับย่อยได้เช่นกัน ประเภทในการเข้าถึงแฟ้มข้อมูล โดยการควบคุมจำแนกสิทธิ์ของผู้ใช้ออกเป็น 3 ประเภทคือ

- เจ้าของ (Owner) เป็นเจ้าของแฟ้มข้อมูลสามารถกำหนดสิทธิ์ให้แก่ผู้ใช้งานทั่วไปได้
- กลุ่ม (Group) เป็นกลุ่มของผู้ใช้ซึ่งจะแชร์แฟ้มข้อมูลใช้งานระหว่างกันได้และต้องการ เข้าถึงที่คล้ายกันเป็นกลุ่มหรือทำงานกลุ่ม
- คนอื่นทั้งหมด (Universe) เป็นกลุ่มของผู้ใช้ทั้งหมดที่มีอยู่ในระบบการป้องกันที่เชื่อมโยงกับแฟ้มข้อมูล

ตัวอย่าง Sara กำลังเขียนหนังสือเล่มใหม่ เธอจ้างคนที่จบปริญญาตรีมา 3 คน (Jim, Dawn, Jill) มาช่วยข้อความในหนังสือเก็บไว้ในแฟ้มข้อมูลชื่อ Book การป้องกันแฟ้มข้อมูลทำได้ดังนี้

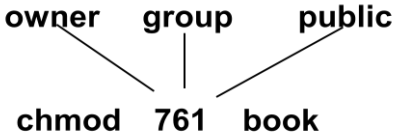
Sara ต้องสามารถทำงานบนแฟ้มข้อมูลนี้ได้ทุกอย่าง Jim, Dawn และ Jill ควรจะอ่านและเขียนแฟ้มข้อมูลนี้ได้เท่านั้น พวกเขาไม่ควรลบแฟ้มข้อมูลนี้ได้ ผู้ใช้คนอื่นควรจะอ่านแฟ้มข้อมูลนี้ได้ (โดย เธอต้องการให้คนทั่วไปได้อ่านและให้ Feedback กลับมา)

เพื่อให้การป้องกันสำเร็จ เราต้องสร้างกลุ่มใหม่ (New group) เรียกว่า Text ให้มีสมาชิกคือ Jim, Dawn และ Jill ชื่อของกลุ่ม text ต้องเกี่ยวข้องกับแฟ้มข้อมูล Book และการเข้าถึงอย่างถูกต้อง เราต้องตั้งตามนโยบายข้างต้น

ตัวอย่างในระบบ UNIX มีการกำหนดบิตขึ้น 3 บิต rwx โดย r หมายถึง ควบคุมการเข้าถึงแบบอ่าน (Read), w หมายถึง ควบคุมการเขียน (Write), x หมายถึง ควบคุมการทำงาน (Execution) ดังนั้นจากตัวอย่างของเรา การป้องกันแฟ้มข้อมูล Book ทำได้ดังนี้

		R W X
สำหรับเจ้าของ (Owner) Sara ทั้ง 3 บิต ถูกตั้งค่าให้เป็นค่า	7 ⇒	1 1 1
สำหรับกลุ่ม (Group) text บิต r และ w ถูกตั้งค่าให้เป็นค่า	6 ⇒	1 1 0
สำหรับคนอื่น (Universe) มีแต่บิต x ที่ตั้งค่าให้เป็นค่า	1 ⇒	0 0 1

ดังนั้นในคำสั่งในระบบ UNIX



ตัวอย่างการแสดงรายการไดเรกทอรี่จาก UNIX แสดงดังภาพที่ 9.12 ฟิลด์แรกบรรยายถึงการป้องกันแฟ้มข้อมูลหรือไดเรกทอรี่ ตัวอักษรแรก d หมายถึงไดเรกทอรี่ย่อย นอกจากนั้นยังแสดงจำนวน Link ของแฟ้มข้อมูล ชื่อเจ้าของ ชื่อของกลุ่ม ขนาดของแฟ้มข้อมูลในหน่วยของไบต์ วันที่สร้าง และชื่อของแฟ้มข้อมูล (ตามด้วยนามสกุล)

```
-rw-rw-r--   1 pbg   staff    31200  Sep 3 08:30   intro.ps
drwx------   5 pbg   staff      512  Jul 8 09.33   private/
drwxrwxr-x   2 pbg   staff      512  Jul 8 09:35   doc/
drwxrwx---   2 pbg   student    512  Aug 3 14:13   student-proj/
-rw-r--r--   1 pbg   staff     9423  Feb 24 2003   program.c
-rwxr-xr-x   1 pbg   staff    20471  Feb 24 2003   program
drwx--x--x   4 pbg   faculty    512  Jul 31 10:31  lib/
drwx------   3 pbg   staff     1024  Aug 29 06:52  mail/
drwxrwxrwx   3 pbg   staff      512  Jul 8 09:35   test/
```

ภาพที่ 9.12 ตัวอย่างการแสดงรายการของไดเรกทอรี่

9.4.3 แนวทางการป้องกันอื่น ๆ

อีกวิธีการในการป้องกันปัญหาที่จะเชื่อมโยงกับรหัสผ่านที่แฟ้มข้อมูล เช่นเดียวกับการเข้าสู่ระบบคอมพิวเตอร์นั้น มักจะควบคุมโดยรหัสผ่านเข้าใช้แฟ้มข้อมูลแต่ละแฟ้มข้อมูลสามารถควบคุม ในทำนองเดียวกัน หากรหัสผ่านจะถูกเลือกแบบสุ่มและการเปลี่ยนแปลงบ่อยจะมีประสิทธิภาพในการจำกัดการเข้าถึงแฟ้มข้อมูล อย่างไรก็ตามการใช้รหัสผ่านมีข้อเสียเหมือนกัน ขั้นแรกจำนวนของรหัสผ่านที่ผู้ใช้ต้องจำอาจมีขนาดใหญ่ สองหากรหัสผ่านหนึ่งรหัสใช้สำหรับทุกแฟ้มข้อมูลเมื่อถูกค้นพบ แฟ้มข้อมูลทั้งหมดสามารถเข้าถึงได้ บางระบบอนุญาตให้ผู้ใช้เชื่อมโยงรหัสผ่านกับไดเรกทอรี่ย่อย แทนที่จะเชื่อมโยงกับแต่ละแฟ้มข้อมูล ปัญหานี้ระบบปฏิบัติการ IBMVM / CMS ยอมให้มีสามรหัสผ่านสำหรับ minidisk โดยแต่ละระดับของรหัสผ่านใช้สำหรับอ่าน เขียน และ multiwrite

บางระบบปฏิบัติการเช่น MS-DOS และระบบปฏิบัติการก่อนหน้า Mac OS X ด้านการป้องกันแฟ้มข้อมูลนั้นมีเพียงเล็กน้อย ในสถานการณ์ที่ระบบเก่าเหล่านี้จะมีการวางแฟ้มข้อมูลไว้ในเครือข่ายที่แชร์แฟ้มข้อมูลและการสื่อสารที่จำเป็นต้องมีการป้องกัน การออกแบบระบบปฏิบัติการใหม่จะทำได้ง่ายกว่าการเพิ่มคุณสมบัติให้ระบบปฏิบัติการที่มีอยู่ การปรับปรุงดังกล่าวมักจะทำให้มีประสิทธิภาพลดน้อยลง

ในระบบ Multilevel โครงสร้างไดเรกทอรี่ต้องป้องกันไม่เพียงแต่ละแฟ้มข้อมูลแต่ยังต้องป้องกันชุดของแฟ้มข้อมูลในไดเรกทอรี่ย่อย การทำงานของไดเรกทอรี่ต้องมีการป้องกันที่ค่อนข้างแตกต่างจากการดำเนินงานแฟ้มข้อมูล และจำเป็นต้องควบคุมการสร้างและลบแฟ้มข้อมูลในไดเรกทอรี่ นอกจากนี้อาจต้องการควบคุมว่าผู้ใช้สามารถตรวจสอบการทำงานของแฟ้มข้อมูลในไดเรกทอรี่ บางครั้งการรู้จักชื่อของแฟ้มข้อมูลก็มีความสำคัญในตัวเอง ดังนั้นรายการเนื้อหาในไดเร็กทอรี่ต้องป้องกันการดำเนินงานในทำนองเดียวกัน หากเส้นทางดูที่ชื่อแฟ้มข้อมูลในไดเรกทอรี่ของผู้ใช้จะต้องได้รับอนุญาตให้เข้าถึงทั้ง ไดเรกทอรี่และแฟ้มข้อมูลในระบบที่มีหลายแฟ้มข้อมูลและหลายชื่อ การกำหนดเส้นทางที่ทำให้ผู้ใช้อาจมีสิทธิในการเข้าใช้ที่แตกต่างกันหนึ่งแฟ้มข้อมูลขึ้นอยู่กับการใช้ชื่อพาธ

9.5 สรุป

ในการจัดเก็บข้อมูลลงในสื่อเก็บข้อมูลจะต้องมีการตั้งชื่อกลุ่มข้อมูล ชื่อที่ว่านี้เรียกว่าแฟ้มข้อมูล ในการเข้าถึงแฟ้มข้อมูล ไม่จำเป็นต้องทราบตำแหน่งที่อยู่ของแฟ้มข้อมูล ระบบปฏิบัติการจะจัดการให้ผ่านทางการดำเนินการจากโปรแกรมของระบบที่เรียกว่า System call ซึ่ง System call เหล่านี้จะช่วยในการสร้าง ลบ อ่าน และเขียนแฟ้มข้อมูลต่าง ๆ ได้ ระบบปฏิบัติการยังช่วยเก็บไฟล์ในกลุ่มเดียวกันไว้ในไดเรกทอรี่ ซึ่งไดเรกทอรี่เป็นแฟ้มข้อมูลประเภทหนึ่ง ที่ช่วยแบ่งกลุ่มแฟ้มข้อมูลที่จัดเก็บในสื่อที่เป็นกลุ่มเพื่อให้การใช้งานมีประสิทธิภาพ ทั้งโครงสร้างไดเรกทอรี่มีทั้งแบบไดเรกทอรี่เดี่ยว ไดเรกทอรี่สองระดับ และไดเรกทอรี่หลายระดับ

ธรรมชาติลักษณะทั่วไปของไดเรกทอรี่สองระดับ คือโครงสร้างแบบต้นไม้ โครงสร้างแบบต้นไม้ของไดเรกทอรี่อนุญาตให้ผู้ใช้สร้างไดเรกทอรี่ย่อย ในการจัดระเบียบไดเรกทอรี่กราฟแบบไม่เป็นวงจร โครงสร้างของไดเรกทอรี่ผู้ใช้จะใช้ในการแบ่งปันแฟ้มข้อมูลและไดเรกทอรี่ย่อย แต่มีความซับซ้อนในการค้นหาและลบโครงสร้างทั่วไปของกราฟ จะช่วยให้เกิดความยืดหยุ่นในการใช้งานร่วมกันของแฟ้มข้อมูลและไดเรกทอรี่ แต่บางครั้งต้องเก็บรวบรวมขยะเพื่อกู้คืนไว้ในส่วนที่ไม่ได้ใช้ในพื้นที่ของดิสก์

เนื่องจากแฟ้มข้อมูลเป็นข้อมูลหลักในเกือบทุกระบบของคอมพิวเตอร์ การป้องกันแฟ้มข้อมูลจึงมีความต้องการมาก การเข้าถึงแฟ้มข้อมูลสามารถควบคุมแยกกันสำหรับแต่ละประเภท การเข้าถึง อ่าน เขียน ทำงาน ลบรายการไดเรกทอรี่ และการใช้ไดเรกทอรี่ร่วมกัน การป้องกันแฟ้มข้อมูลสามารถป้องกันโดยรหัสผ่านก่อนเข้าถึงรายการ หรือเทคนิคอื่น ๆ ในแต่ละระบบปฏิบัติการ

แบบฝึกหัดท้ายบทที่ 9

1. จงอธิบายคุณลักษณะของแฟ้มข้อมูลคืออะไร มีรายละเอียดอย่างไรบ้าง
2. จงอธิบายถึงวัตถุประสงค์ของการดำเนินการใช้ฟังก์ชั่นเพื่อใช้ในการเปิดแฟ้มข้อมูล และปิดแฟ้มข้อมูล
3. จงอธิบายประเภทของแฟ้มข้อมูลที่คุณรู้จักและบอกว่าประเภทของแฟ้มข้อมูลนี้ใช้โปรแกรมเกี่ยวกับอะไรหรือดำเนินการอย่างไร
4. จงยกตัวอย่างของโปรแกรมที่ใช้วิธีการเข้าถึงแฟ้มข้อมูลแบบ Sequential และ แบบ Random
5. จงอธิบายโครงสร้างข้อมูล (แฟ้มข้อมูล Structure) หมายถึงอะไร ลักษณะการจัดแบ่งพิกัดต่าง ๆ ของข้อมูลในแฟ้มข้อมูล เพื่อให้คอมพิวเตอร์สามารถรับไปประมวลผลได้มีอะไรบ้าง
6. จงอธิบายข้อแตกต่างระหว่างไดเรกทอรี่แบบกราฟโดยทั่วไปกับแบบ ไดเรกทอรี่กราฟแบบไม่เป็นวงจร
7. จงอธิบายการกำหนดชื่อพาธมีไว้เพื่ออะไร และยกตัวอย่างการกำหนดชื่อพาธของระบบปฏิบัติการวินโดว์ และระบบปฏิบัติการลินุกส์ เพื่อให้สามารถเรียกใช้แฟ้มข้อมูลนั้นจาก Root
8. จงอธิบายประเภทในการเข้าถึงแฟ้มข้อมูลโดยการควบคุมจำแนกสิทธิ์ของผู้ใช้ การป้องกันและกำหนดสิทธิ์ในระบบปฏิบัติการยูนิกส์
9. ให้นักศึกษาค้นคว้าเพิ่มเติมว่า ในระบบปฏิบัติการในปัจจุบันที่กำหนดแนวทางป้องกันการเข้าถึง

แฟ้มข้อมูล และการเข้าถึงไดเรกทอรี่และไดเรกทอรี่ย่อยว่าใช้หลักการอย่างไร โดยยกตัวอย่างมาอย่างน้อย 1 ระบบปฏิบัติการ

บรรณานุกรมประจำบทที่ 9

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

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

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

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

Andrew S. Tanenbaum, Herbert Bos. (2014). **Modern Operating Systems. 4th ed.** Prentice Hall, Pearson Education International.