บรรยายครั้งที่ ๒

ตอนที่ ๑ - ปฏิบัติการจากสัปดาห์ที่แล้ว

๑. การปรับแต่งการทำงานของโปรแกรมในระบบปฏิบัติการ Unix ดำเนินการผ่านแฟ้ม run control ซึ่งเป็น configuration file

- โปรแกรมที่ทำการปรับแต่ง
  bash - .bash_profile
  vi - .exrc
- เราควรจะสามารถปรับแต่งการทำงานของโปรแกรมได้ทุกโปรแกรมหรือไม่? เพราะเหตุใด?
- การปรับแต่งการทำงานของโปรแกรมในระบบปฏิบัติการ Microsoft Windows ทำได้หรือไม่? และทำได้อย่างไร?
  - การเลือก "แบบ" ของตัวอักษร (Font) และ "ขนาด" ของตัวอักษร (Size) ในโปรแกรม SSH ทำได้หรือไม่? อย่างไร
  - โปรแกรมในกลุ่ม Browser: Mozilla Firefox, Microsoft Internet Explorer, Google Chrome สามารถปรับแต่งการทำงานได้หรือไม่? อย่างไร?
  - Preference ในโปรแกรมประยุกต์บางตัว เป็นการปรับแต่งการทำงานของโปรแกรมหรือไม่?

๒ - Online manual page มีกี่ section

1 Executable programs or shell commands
   คำสั่งของ Unix
2 System calls (functions provided by the kernel)
   บริการของระบบปฏิบัติการที่มีให้โปรแกรมประยุกต์เรียกใช้งาน
3 Library calls (functions within program libraries)
   โปรแกรมย่อยจากคลังมาตรฐาน (standard library) ของภาษา C
4 Special files (usually found in /dev)
   รายละเอียดของแฟ้มแทนอุปกรณ์ เช่น /dev/tty - controlling terminal
5 File formats and conventions eg /etc/passwd
   โครงสร้างข้อมูลของแฟ้มสำคัญในระบบ เช่น แฟ้มเก็บรหัสผ่าน (/etc/passwd)
6 Games
7 Miscellaneous (including macro packages and conventions),
   เรื่องเบ็ดเตล็ดรวมถึง macro เช่น man(7), groff(7)
8 System administration commands (usually only for root)
   คำสั่งควบคุมการทำงานของระบบสำหรับผู้ดูแลระบบ
9 Kernel routines [Non standard]

๓ - option สำหรับการ and คำสำคัญของคำสั่ง apropos

default - inclusive or

and - option -a

ตัวอย่าง - index AND file

$ apropos index -a file

๔ - คำสั่ง chmod และความเหมาะสม

สิทธิในการใช้งานแฟ้ม -r-xr-xrw- เป็นสิทธิการใช้งานที่สมเหตุสมผลหรือไม่? เพราะเหตุใด?

คำสั่ง

-r-xr-xrw-

owner = r-x = 101 = 5

group = r-x = 101 = 5

other = rw- = 110 = 6

$ chmod 0556 <ชื่อแฟ้ม>

ความสมเหตุสมผล

เจ้าของมีสิทธิอ่าน - และให้โปรแกรมทำงาน

กลุ่มที่เจ้าของเป็นสมาชิก มีสิทธิเช่นเดียวกับเจ้าของ

ผู้ใช้ทั่วไป มีสิทธิในการอ่าน - เขียน

เจ้าของและกลุ่มแก้ไขข้อมูลในแฟ้มของตนเองไม่ได้ ผู้ใช้อื่นแก้ได้?

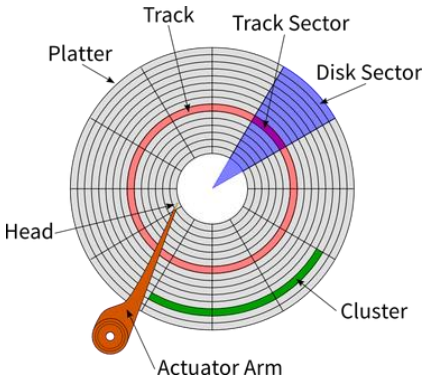
ความรู้ทั่วไปเรื่องระบบแฟ้ม (Filesystems)

การพิจารณาระบบแฟ้มทำได้ 2 ระดับ คือระดับกายภาพ และระดับมโนภาพ

กายภาพ (Physical)

- อุปกรณ์เก็บข้อมูลความจุสูง (Mass storage) ในระบบคอมพิวเตอร์ -- มีหลายชนิด โครงสร้างและวิธีการจัดเก็บข้อมูลต่างกัน

  - HDD: มีหลาย track แต่ละ track มีหลาย sector เป็นอุปกรณ์หลักสำหรับระบบคอมพิวเตอร์



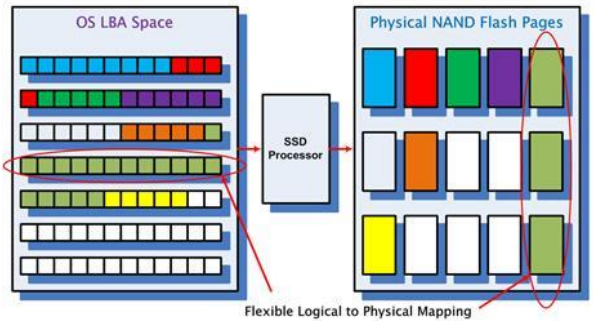
- Optical disk: CD/DVD มี track เดียวแบ่งเป็นหลาย sector
- Flash: ไม่มี track และ sector

Disk Geometry

จำนวนหัว อ่าน/เขียน (หรือจำนวนผิวจานสำหรับเก็บข้อมูล), จำนวน cylinders และ sectors แตกต่างกันไปตามรุ่นและยี่ห้อของจานแม่เหล็ก ข้อมูลนี้เรียกว่า Disk geometry

- หน่วยประมวลผลกลาง (CPU) ติดต่อสื่อสาร OS เพื่อบังคับจานแม่เหล็กผ่าน"วงจรควบคุม" (disk controller) ซึ่งออกแบบให้มีการเชื่อมต่อ (interface) เป็นมาตรฐานเดียวกัน
- ระบบคอมพิวเตอร์เพียงแต่สั่งการผ่าน interface ว่าต้องการดำเนินการใดกับข้อมูล วงจรควบคุมซึ่งออกแบบสำหรับจานแม่เหล็กนั้นจะปฏิบัติการให้เป็นไปตามนั้น เป็นการลดภาระของระบบที่จะต้อง "รู้" วิธีการติดต่อกับจานแม่เหล็กทีมี disk geometry ต่างกัน
- เมื่อมีการนำจานแม่เหล็กรุ่นใหม่มาใช้งาน จะมาพร้อมด้วยวงจรควบคุมณแพาะของจานแม่เหล็กนั้น ทำให้สามารถติดตั้งใช้งานได้ทันที

ในปัจจุบันใช้ LBA (Logical Block Addressing) ระบบปฏิบัติการจะเข้าถึงโดยการใช้ sector โดยไม่สนใจว่า sector นั้นอยู่ที่ตำแหน่งใดใน disk



นอกจากนี้จำนวนของ sector ในแต่ละ track ในปัจจุบันยังไม่เท่ากัน โดย track ที่อยู่ใกล้แกนหมุนมีจำนวน sector น้อย และมีจำนวน sector เพิ่มขึ้นเมื่อเข้าใกล้ริมแผ่น เพื่อทำให้เนื้อที่เก็บข้อมูลของ sector มีขนาดใกล้เคียงกัน จำนวน track ของ hard drive จึงไม่มีความสำคัญ

Low-level format

เมื่อออกจากสายการผลิต สื่อแม่เหล็กบนผิวหน้าของจานยังไม่มีการจัดระเบียบและกำหนดตำแหน่ง ก่อนใช้งานจึงต้องทำการ format เพื่อเขียนตำแหน่งเพื่อกำหนดขอบเขตของ track และ sector

โดยทั่วไปเป็นหน้าที่ของบริษัทผู้ผลิตเป็นผู้ดำเนินการ ทั้งนี้เนื่องจาก geometry ของจานแม่เหล็กแตกต่างกันออกไปมาก การ format จึงเป็นกระบวนการที่ต้องเรียกใช้บริการในระดับของ controller การ format ในระดับนี้เรียกว่า low-level format เพื่อให้มีความชัดเจน

ทั้งนี้เพราะระบบปฏิบัติการของบริษัท Microsoft ใช้คำว่า format ในการสร้างระบบแฟ้มบนสื่อแม่เหล็ก หรือเรียกว่า high-level format

ระหว่างการทำ Low-level format อาจพบพื้นผิวที่บกพร่อง ไม่เหมาะสมสำหรับการเก็บข้อมูล พื้นที่เช่นนี้เรียกว่า bad blocks หรือ bad sectors

ซึ่งส่วนใหญ่ ชุดจานแม่เหล็กจะดำเนินการเอง เช่น สร้างตารางแสดงตำแหน่งของ bad blocks เพื่อหลีกเลี่ยงการใช้งาน หลักจาก format แล้วการสึกหรอระหว่างการใช้งานอาจทำให้มี bad blocks เกิดเพิ่มขึ้น หากมีจำนวนมากขึ้นอาจต้องทำ Low-level format ใหม่ การตรวจสอบ bad blocks ทำได้โดยใช้คำสั่ง badblocks ซึ่งเป็นคำสั่งเฉพาะสำหรับ superuser

จานแม่เหล็กสมัยใหม่สามารถตรวจจับ bad blocks ด้วยตนเอง และเปลี่ยนไปใช้ blocks ที่ดีโดยอัตโนมัติ โดยระบบปฏิบัติการไม่รับรู้ถึงการเปลี่ยนแปลง คุณสมบัติดังกล่าวนี้ (หากมี) จะระบุไว้ในคู่มือของจานแม่เหล็กนั้น --> การหัดอ่านคู่มือจึงเป็นเรื่องสำคัญ

Partitions

hard drive หนึ่งตัวสามารถแบ่งออกได้เป็นหลาย partition

แต่ละ partition ทำหน้าที่เหมือนกับเป็น hard disk แต่ละตัว จึงสามารถติดตั้งระบบปฏิบัติการที่แตกต่างกันลงในแต่ละ partition ได้โดยไม่มีการรบกวนการทำงานซึ่งกันและกัน และจานแม่เหล็กหนึ่งตัวต้องมีอย่างน้อยหนึ่ง partition

ข้อมูลการแบ่ง partition ของจานแม่เหล็กเก็บอยู่ใน sector แรกของ track แรก มีชื่อเรียกเฉพาะว่า Master Boot Record (MBR) ซึ่งเป็น sector ที่ BIOS จะอ่านและเริ่มการทำงานเมื่อมีการ boot ระบบ

ใน MBR มีโปรแกรมขนาดเล็กสำหรับตรวจสอบว่า partition ใดเป็น active partition (กำหนดคุณสมบัติไว้เป็น bootable) จากนั้นจึงทำการอ่าน sector แรกของ partition นั้น ซึ่งมีชื่อเฉพาะเป็น boot sector ซึ่งเก็บ โปรแกรมที่จะทำหน้าที่อ่านส่วนแรกของระบบปฏิบัติการเข้าในหน่วยความจำและเริ่มทำงาน เพื่ออ่านส่วนที่เหลือของระบบปฏิบัติการเข้าสู่หน่วยความจำและทำงานต่อไป จนกว่าจะจะได้ระบบปฏิบัติที่สมบูรณ์พร้อมทำงาน

การแบ่ง partition ทำในขณะติดตั้งระบบปฏิบัติการ (โดยเรียกใช้คำสั่ง fdisk) การตรวจสอบและการจัดการกับ partition ทำได้โดยใช้คำสั่ง fdisk ซึ่งเป็นคำสั่งในระดับ superuser

Hard disk สำหรับเครื่องคอมพิวเตอร์ส่วนบุคคลในยุคแรกออกแบบไว้ให้มี partition ได้สูงสุดไม่เกิน 4 partitions ซึ่งในระยะต่อมาไม่เพียงพอสำหรับการใช้งาน เช่น ผู้ใช้บางรายต้องการใช้ระบบปฏิบัติการมากกว่า 4 ระบบในเครื่องเดียวกัน แต่แรงขับดันที่แท้จริงมาจากการที่ระบบปฏิบัติการแต่ละระบบต้องการ partition เฉพาะสำหรับดำเนินการบางอย่างเพื่อเพิ่มประสิทธิภาพการทำงาน เช่นระบบปฏิบัติการ Unix ต้องการ swap partition สำหรับใช้ในการสลับโพรเซสเข้าออกระหว่างหน่วยความจำหลักและจานแม่เหล็ก, ระบบปฏิบัติการของบริษัท Microsoft ต้องการแบ่ง partition ออกเป็น drive อิสระ เช่นแบ่งจานแม่เหล็กตัวหนึ่งออกเป็น drive C:, D: และ E: เป็นต้น

เพื่อแก้ปัญหาข้อจำกัดจำนวน partition จึงกำหนดให้เรียก partition เดิมแต่ละ partition ว่า primary partition ซึ่งสามารถแบ่งเป็น extended partition ได้ตามจำนวนที่ต้องการ (<= 15 partitions ต่อจานแม่เหล็กจริง 1 ตัว), extended partition จึงมีลักษณะเป็น logical partition ที่มีการทำงานเทียบเท่า primary partition เดิม

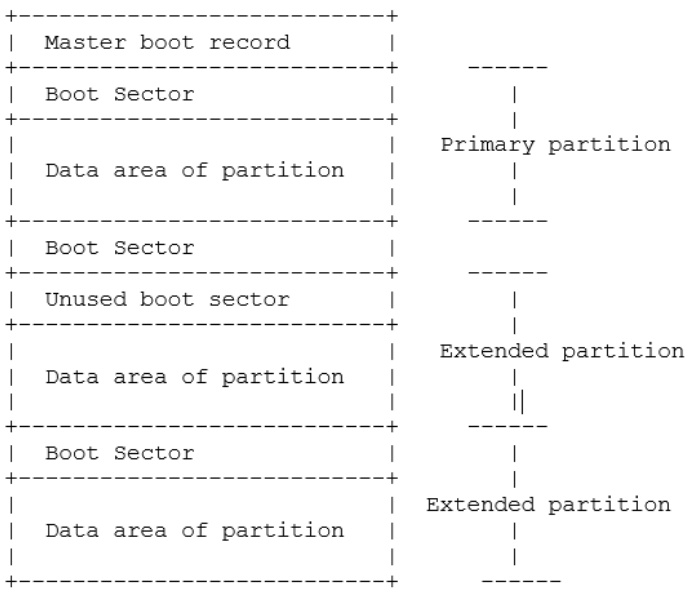
เพื่อแก้ปัญหาข้อจำกัด จึงกำหนดให้สร้าง partition หลัก หรือ Primary partition ขึ้นก่อนไม่เกิน 4 partitions จากนั้นจึงทำการแบ่ง primary partition แจ่ละตัว ออกเป็น extended partition ตามจำนวนที่ต้องการ, Extended partition ที่สร้างขึ้นเป็น logical partition ที่มีหน้าที่และการทำงานเทียบเท่า primary partition เดิม และสามารถสร้างได้สูงสุดถึง 15 partitions ต่อจานแม่เหล็กจริง 1 ตัว

แผนภาพแสดงตัวอย่างการแบ่ง partition

จานแม่เหล็กในแผนภาพนี้มี 2 primary partitions

- partition แรก เป็น primary partition มี boot sector และพื้นที่เก็บข้อมูล สำหรับติดตั้งระบบปฏิบัติการหนึ่ง
- partition ที่สอง เป็น primary partition สำหรับติดตั้งระบบปฏิบัติการอีกระบบหนึ่ง โดยแบ่งออกเป็น 2 extended partitions

เนื่องจาก partition ทั้งสองใช้ระบบปฏิบัติการเดียวกัน จึงใช้เฉพาะ boot sector ของ primary partition ไม่ได้ใช้ boot sector ของ extended partition



คำถาม

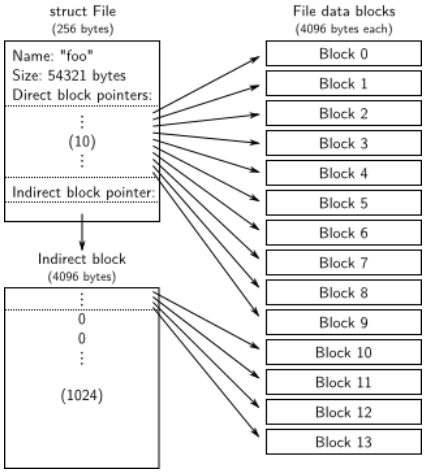
---

1. Master boot record (MBR) คืออะไร? ใช้สำหรับเก็บข้อมูลใด?
2. จากแผนภาพและคำอธิบาย เราสามารถติดตั้งระบบปฏิบัติการสองระบบที่ต่างกันลงใน extended partition ได้หรือไม่? เพราะเหตุใด?

File System

ระบบแฟ้มคือ data structure และ algorithm ของระบบปฏิบัติการในการจัดการ ตรวจสอบ และติดตามแฟ้มที่อยู่ใน partition

กล่าวอีกนัยหนึ่งคือ เป็นโครงสร้างและวิธีการในการจัดเก็บแฟ้ม



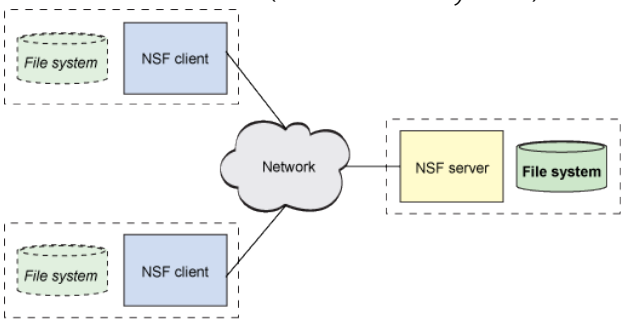
ความแตกต่างระหว่าง partition และระบบแฟ้มเป็นประเด็นที่สำคัญ
มี Utility จำนวนน้อยมากที่ทำงานโดยตรงกับ sector ของ partition ตัวอย่างที่ชัดเจนที่สุดคือโปรแกรมที่ทำหน้าที่สร้าง filesystem (เช่น mkfs - make file system)

โปรแกรมเกือบทั้งหมดทำงานผ่านระบบแฟ้ม ดังนั้นก่อนที่ partition ใดจะสามารถใช้เก็บแฟ้มได้ ต้องทำการสร้าง file system คือการกำหนดโครงสร้างข้อมูลให้เรียบร้อยเสียก่อน

เนื่องจากระบบปฏิบัติการ Unix (และ Linux) สนับสนุนการใช้งานระบบแฟ้มหลายชนิด เช่น ext2, ext3, minix, nfs, msdos, และ ntfs เป็นต้น

ผู้สร้างจึงต้องตัดสินใจว่าจะเลือกใช้ระบบแฟ้มใด ระบบแฟ้มที่เหมาะสมสำหรับการใช้งานระบบปฏิบัติการ Unix คือ ext2 และ ext3

สำหรับระบบแฟ้มอื่นมีไว้สำหรับให้ระบบปฏิบัติการ Unix เป็นผู้ให้บริการแฟ้มสำหรับระบบปฏิบัติการอื่น เช่น nfs สำหรับใช้ระบบแฟ้มจากเครื่องอื่นในเครือข่าย (Network file system)



msdos และ ntfs สำหรับระบบปฏิบัติการของบริษัท Microsoft

ระบบแฟ้มทั้งหมดที่ Unix สนับสนุนสามารถศึกษาได้จาก Online manual page ในหมวดที่ 5 โดยใช้คำสั่ง

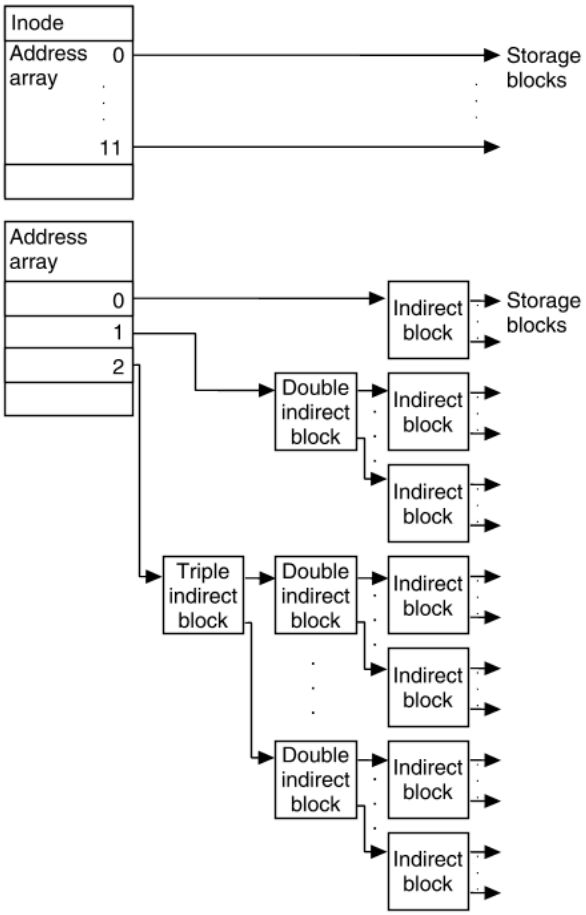
$ man -s5 fs

เมื่อเลือกระบบแฟ้มได้แล้ว การสร้างใช้คำสั่งในตระกูล mkfs (make file system) เช่นต้องการสร้างระบบแฟ้มชนิด ext2, ext3, และ ext4

ทำได้โดยใช้คำสั่ง mk2efs - create an ext2/ext3/ext4 filesystem คำสั่งนี้มี option สำหรับกำหนด parameters มากมาย ควรศึกษารายละเอียดต่าง ๆ ให้ชัดเจนก่อนใช้งาน

โครงสร้างทางกายภาพของระบบแฟ้มแต่ละ partition

- Boot block: พื้นที่เก็บโปรแกรมสำหรับการเริ่มทำงาน (Bootstrapping)
- Superblock: เก็บข้อมูลของระบบแฟ้มทั้งหมด เช่น ชนิด ขนาด และสถานะของระบบแฟ้ม, ขนาดของ block, รายการ block ที่ใช้งานแล้วและ block ที่ว่างอยู่, ขนาดและตำแหน่งของตาราง i-node
- directory block: เป็น data block ที่ใช้สำหรับเก็บรายชื่อแฟ้มที่อยู่ในไดเรกทอรี ได้แก่ชื่อแฟ้มและหมายเลข i-node
- i-list: เป็น list (ตาราง) ของ i-node ซึ่งเป็นโครงสร้างข้อมูลที่ใช้เก็บรายละเอียดของแต่ละแฟ้ม ยกเว้นชื่อแฟ้ม
- data block: พื้นที่สำหรับเก็บข้อมูลทีอยู่ในแฟ้ม
- indirection block: พื้นที่สำหรับเก็บหมายเลขของ data block สำหรับแฟ้มที่มีขนาดใหญ่



แฟ้มแต่ละแฟ้มมี i-node เฉพาะตัว หรือ แฟ้มสองแฟ้มที่มี i-node เดียวกัน เป็นแฟ้มเดียวกัน

ชื่อแฟ้มและหมายเลข i-node จัดเก็บใน directory block ซึ่งเป็น data block ที่ใช้เก็บ "รายชื่อแฟ้ม" แทนที่จะเป็น "ข้อมูลในแฟ้ม" ข้อมูลอย่างหนึ่งของ i-node คือหมายเลขของ data block ที่ใช้เก็บข้อมูลจริง ดังนั้นการอ่านข้อมูลจากแฟ้มจึงต้องอ่านหมายเลขของ data block จาก i-node ก่อน เพื่อไปยัง

data block ที่กำหนดเพื่ออ่านข้อมูล ในกรณีที่แฟ้มมีขนาดใหญ่ ใช้จำนวน data block มากกว่าที่จะเก็บได้ใน i-node จะทำการสร้าง indirection block เพื่อเก็บหมายเลข data block และกำหนด pointer ไปยัง indirection block ลงใน i-node

Mount และ Unmount

ระบบแฟ้มของ Unix เป็น single directory tree มี root (/) เป็นจุดเริ่มต้น ระบบแฟ้มที่สร้างเสร็จแล้ว ต้องนำมา mount เข้ากับระบบแฟ้มที่มีอยู่
partition แรก mount เข้ากับ /, partition ต่อไปสามารถเลือกได้ว่าจะ mount ที่ subdirectory ใด

CD-ROM, DVD-ROM และ Flash drive เป็นสื่อเก็บข้อมูลและมีระบบแฟ้มเช่นเดียวกัน แต่เป็นสื่อชนิดถอดเข้าออกได้ (Removable media)

ก่อนจะใช้งานได้ ต้องทำการ mount เช่นเดียวกัน โดยทั่วไปจะจัด /media ไว้ให้เป็น mount point เช่น

/media/cdrom	# mount point สำหรับ CD-ROM
/media/floppy	# mount point สำหรับ diskette หรือ floppy disk

และเมื่อเลิกใช้งาน ก่อนจะถอดออกจากระบบต้องทำการ unmount

การ mount และ unmount ดำเนินการผ่านคำสั่ง

$ mount - mount a filesystem

$ umount - unmount filesystem

มโนภาพ (Logical)

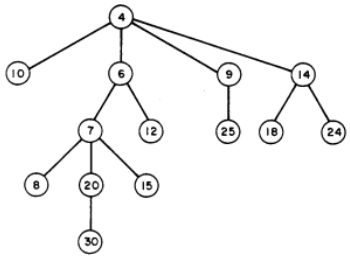
อุปกรณ์เก็บข้อมูลความจุสูงไม่ว่าจะเป็น HDD, CD/DVD หรือ Flash ใช้สำหรับเก็บ "แฟ้ม" เหมือนกัน ผู้ใช้รู้เฉพาะความแตกต่างระหว่างอุปกรณ์เฉพาะที่จำเป็น

CD/DVD -- อ่านได้อย่างเดียว หากต้องการเขียน (บันทึก) ข้อมูล ต้องใช้โปรแกรมพิเศษในการดำเนินการ

Flash -- เป็นอุปกรณ์ที่สามารถถอดออกจากระบบได้ (Removable media) การใส่เข้า/ถอดออกต้องแจ้งให้ระบบรับรู้และดำเนินการ (mount/unmount)

ภาพโดยรวม

ระบบแฟ้มใช้จัดเก็บและดำเนินการกับระบบแฟ้ม แฟ้มแต่ละแฟ้มมี "ตำแหน่ง" ที่แน่นอนในระบบแฟ้ม ในทัศนะนี้ ระบบแฟ้มเป็นโครงสร้างข้อมูล -- **Multiway tree** คือ tree ที่ parent node มีจำนวน child node ได้ไม่จำกัด โดยทั่วไปนิยมเรียกว่าโครงสร้างตามลำดับชั้น (hierarchical structure)



ระบบแฟ้มมีจุดเริ่มต้น เรียกว่า root ตามโครงสร้าง tree
root เป็นจุดเริ่มต้นสำหรับการนำแฟ้มมาเชื่อมต่อเข้าเป็น tree หรือเป็น mount point

โครงสร้าง tree	ระบบแฟ้ม
root	mount point สำหรับระบบแฟ้ม
intermediate node	directory หรือ folder
leaf node	directory หรือ แฟ้ม

ระบบแฟ้มของระบบปฏิบัติการ Unix มีโครงสร้างแบบ tree เพียงต้นเดียว มีจุดยอดหรือ **root เพียงจุดเดียว** ซึ่งต่างจากระบบปฏิบัติการของบริษัทไมโครซอฟต์ซึ่งแยก tree ออกไปตามสื่อเก็บข้อมูล กล่าวคือ Mass storage แต่ละตัว จะมีโครงสร้าง tree และมีจุดยอดเป็นของตนเอง การใช้งานจึงต้องระบุชื่ออุปกรณ์และระบบแฟ้ม

เนื่องจากระบบแฟ้มมีโครงสร้างแบบ tree การอ้างอิงจึงใช้ทิศทางแบบ tree คือ "ขึ้น" หมายถึงเปลี่ยนระดับเข้าใกล้ root, "ลง" เปลี่ยนระดับออกห่างจาก root นอกจากนี้ความสัมพันธ์ระหว่าง node เป็นไปเช่นเดียวกับโครงสร้าง tree คือเป็น Parent-child relationship

การอ้างอิงแฟ้ม

ทำได้โดยการระบุตำแหน่งใน tree เป็นเส้นทาง (path) ซึ่งอาจ
เริ่มจาก root node จนถึงตำแหน่งที่ต้องการ เรียกว่า **เส้นทางสัมบูรณ์ (Absolute path)** หรือ
เริ่มจากจุดอ้างอิง เช่น current directory (.), parent directory (..) หรือ home directory (~)
เรียกว่า**เส้นทางสัมพัทธ์ (Relative path)**

โดยคั่นเส้นทางในแต่ละระดับด้วย / (slash) ดังนี้ pathname/<ชื่อแฟ้ม>

ข้อสังเกต
---
1. เนื่องจาก tree เป็นโครงสร้างข้อมูลชนิด recursive -- การดำเนินการระบบแฟ้มจึงสามารถกำหนดให้ดำเนินการแบบ recursive ได้โดยใช้ option -r
2. directory คือแฟ้มที่ใช้เก็บรายชื่อแฟ้ม

ระบบปฏิบัติการ Unix ส่วนใหญ่ สามารถตั้งชื่อแฟ้มได้สูงสุดถึง 255 ตัวอักขระ ประกอบด้วย
- อักษรตัวใหญ่ (Uppercase, A - Z) และอักษรตัวเล็ก (Lowercase, a - z) -- อักษรตัวใหญ่และตัวเล็กแตกต่างกัน
- digits 0 - 9
- เครื่องหมาย underscore (_)
- เครื่องหมายจุด (.), และ
- เครื่องหมายลูกน้ำ (,)

แฟ้มสองแฟ้มที่อยู่ใน directory เดียวกันจะมีชื่อเดียวกันไม่ได้ แต่แฟ้มสองแฟ้มที่อยู่ต่าง directory กันสามารถใช้ชื่อเดียวกันได้

คำถาม ลูกสองคนที่เกิดจากบิดามารดาเดียวจะมีชื่อเดียวกันได้หรือไม่? เพราะเหตุใด?

การตั้งชื่อแฟ้ม ควรเป็นชื่อที่สื่อความหมาย เพราะเมื่อมีจำนวนแฟ้มเป็นจำนวนมากในไดเรกทอรีและเป็นแฟ้มที่ชื่อไม่สื่อความหมายใด เมื่อต้องใช้แฟ้มใดแฟ้มหนึ่งและจำชื่อแฟ้มนั้นไม่ได้ จะทำให้เจ้าของแฟ้มเสียเวลาในการเปิดแฟ้มเพื่อดูข้อมูลจนกว่าจะพบแฟ้มที่ต้องการ

Tips:
---
1. ในกรณีที่นึกชื่อแฟ้มภาษาอังกฤษไม่ออก ให้ป้อนชื่อภาษาไทยเข้าใน Web dictionary เช่น http://dict.longdo.com
ข้อมูลในแฟ้มเป็นบัญชีเงินเดือน ป้อนคำค้นเป็น "บัญชีเงินเดือน" จะได้คำภาษาอังกฤษเป็น payroll ซึ่งสามารถนำไปตั้งชื่อแฟ้มได้ และยังเป็นการเพิ่มพูนทักษะภาษาอังกฤษให้แก่ตนเองด้วย

Filename extensions

Filename extension หรือส่วนขยายของแฟ้ม เป็นส่วนของชื่อแฟ้มที่อยู่หลังเครื่องหมายจุด โดยปกติใช้แสดงประเภทของข้อมูลในแฟ้ม เช่น hello.c เป็นแฟ้มโปรแกรมต้นฉบับภาษา C, helio.cpp เป็นแฟ้มโปรแกรมต้นฉบับภาษา C++ ในระบบปฏิบัติการ unix ส่วนขยายของแฟ้มมีฐานะเป็นเพียง "ตัวเลือก" จะเลือกใช้หรือไม่ก็ได้ และอาจมีได้หลายชุด เช่น file.tar.gz

นำแฟ้มหลายแฟ้มมาต่อกันเป็นการเตรียมการสำรองข้อมูลด้วยเทป (Tape Archive)
|
file.tar.gz
|
บีบอัดด้วยโปรแกรม gzip

Hidden filename

แฟ้มที่ชื่อขึ้นต้นด้วยเครื่องหมายจุด (period) เรียกว่าแฟ้มซ่อน (hidden file) เพราะไม่แสดงผลด้วยคำสั่งแสดงรายชื่อแฟ้ม (ls) ตามปกติ หากต้องการให้แสดงต้องระบุด้วย option -a (All) ซึ่งนอกจากจะมีการแสดงผลแฟ้มตามปกติและแฟ้มซ่อนแล้ว คำสั่งนี้จะแสดงไดเรกทอรี

อีกสองไดเรกทอรีซึ่งมีชื่อเป็น . และ .. ซึ่งเป็นสัญลักษณ์ที่ใช้แทน current และ parent directory ตามลำดับ

Working directory

เมื่อผู้ใช้ Login เข้าทำงาน ระบบปฏิบัติการ Unix จะ**เชื่อมโยงชื่อผู้ใช้เข้ากับไดเรกทอรีอันใดอันหนึ่งเสมอ** กล่าวอีกนัยหนึ่งคือชื่อผู้ใช้มีความสัมพันธ์กับตำแหน่งใดตำแหน่งหนึ่งในระบบแฟ้มเสมอ ผู้ใช้สามารถเปลี่ยนไปจากไดเรกทอรีหนึ่งไปยังอีกไดเรกทอรีหนึ่งได้ตามความต้องการ (ซึ่งต้องเป็นไดเรกทอรีที่ผู้ใช้นั้นมีสิทธิในการใช้งาน) ไดเรกทอรีที่ผู้ใช้ใช้งานในปัจจุบันเรียกว่า working directory หรือ current directory ต่อไปจะใช้คำว่า "ไดเรกทอรีปัจจุบัน" แทน ผู้ใช้สามารถสร้าง ลบ และเรียกใช้งานแฟ้มในไดเรกทอรีปัจจุบันได้โดยเพียงแต่ระบุชื่อแฟ้ม

การตรวจสอบไดเรกทอรีปัจจุบันทำได้โดยใช้คำสั่ง pwd (print working directory) ซึ่งจะแสดงผลเป็นเส้นทางในระบบแฟ้มตั้งแต่ Root directory จนถึงไดเรกทอรีปัจจุบัน การอ้างอิงไดเรกทอรีปัจจุบันใช้เครื่องหมาย . เป็นตัวแทน

Home directory

เมื่อผู้ใช้ Login เข้าทำงาน ระบบปฏิบัติการ Unix จะเชื่อมโยงชื่อผู้ใช้เข้ากับไดเรกทอรีเริ่มต้นที่กำหนดไว้สำหรับผู้ใช้นั้น ไดเรกทอรีนี้เรียกว่า

Home directory มีสัญลักษณ์แทนในระบบแฟ้มเป็น ~ (tilde) ไดเรกทอรีนี้เป็นจุดเริ่มต้นของระบบแฟ้มของผู้ใช้แต่ละราย ผู้ใช้แต่ละคนมีจุดเริ่มต้นต่างกัน สามารถตรวจสอบตำแหน่งของ home directory โดยใช้คำสั่ง pwd เมื่อ login เรียบร้อยแล้ว เช่น

```
$ pwd
/home/staff/jira                # Home directory ของผู้สอน
```

Home directory เป็นที่เก็บ startup file หรือ configuration file สำหรับ shell และโปรแกรมประยุกต์

Pathname

แฟ้มแต่ละแฟ้มมี pathname ซึ่งได้แก่เส้นทางจากไดเรกทอรีต้นทางไปตามโครงสร้างของระบบแฟ้ม จนถึง "แฟ้ม" หรือ "ไดเรกทอรี" ที่ต้องการ ชื่อที่อยู่ด้านซ้ายมือของ / เป็นชื่อไดเรกทอรีเสมอ ส่วนชื่อที่อยู่ด้านขวามืออาจเป็นชื่อไดเรกทอรีหรือแฟ้มก็ได้ pathname อย่างง่ายที่สุดคือชื่อแฟ้มในไดเรกทอรีปัจจุบัน

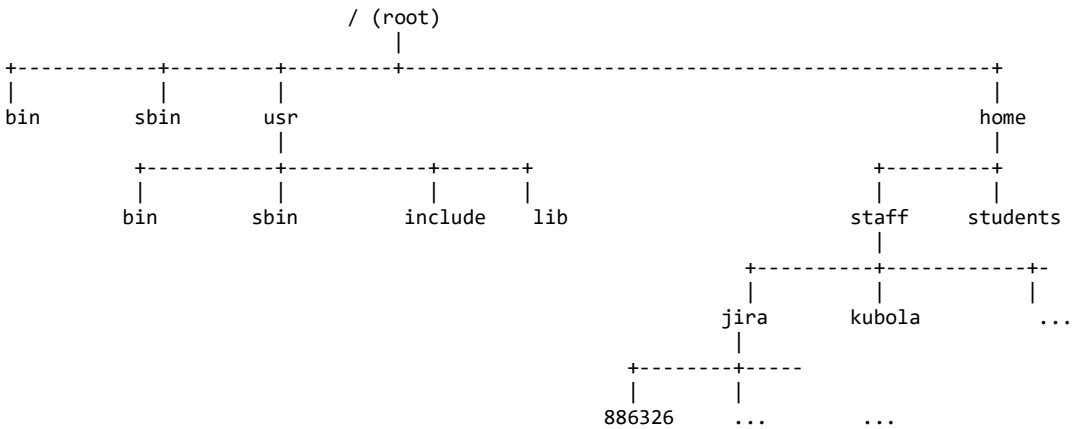
เส้นทางสัมบูรณ์ (Absolute pathname)

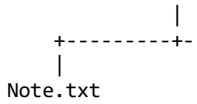
Root directory ของระบบแฟ้มเป็นไดเรกทอรีที่ไม่มีชื่อเฉพาะ แทนด้วยสัญลักษณ์ / เพียงตัวเดียว หรือ / ซึ่งเป็นอักขระตัวแรกด้านซ้ายมือสุดของ pathname เส้นทางสัมบูรณ์ไปยังแฟ้มคือ ชื่อเส้นทางซึ่งอักขระตัวแรกด้านซ้ายสุดเป็น / หรือ root directory ไปยังแฟ้มที่ต้องการ

นอกจากนี้แล้วยังสามารถเริ่มต้นด้วย ~/ ซึ่ง shell จะทำการขยายเป็น absolute pathname ของ Home directory ของผู้ใช้โดยอัตโนมัติ เมื่อก่อนจะเริ่มทำงานตามคำสั่ง

เส้นทางสัมพัทธ์ (Relative pathname)

เป็นเส้นทางจากไดเรกทอรีปัจจุบัน (current หรือ working directory) ไปยังแฟ้มที่ต้องการ หรือเป็นการกำหนดเส้นทางไปยังแฟ้มนั้นโดยอ้างอิงจากไดเรกทอรีปัจจุบัน





Absolute path ของแฟ้ม Note.txt
/home/staff/jira/886326/Note.txt

เฉพาะกรณีของผู้ใช้ jira
~/886326/Note.txt

การดำเนินการกับแฟ้มและไดเรกทอรี

Directory	สร้าง	-- mkdir (make directory) ชื่อต้องไม่ซ้ำกับไดเรกทอรีที่มีอยู่แล้ว
	ลบ	-- rmdir (remove directory) **ไดเรกที่จะลบได้ต้อง "ว่าง"**
	เปลี่ยน	-- cd (change directory)
	...	
	การย้าย และ/หรือ เปลี่ยนชื่อแฟ้ม	-- mv (move) ใช้ได้ทั้งแฟ้มและไดเรกทอรี
	...	
File	สร้าง	-- Application program เช่น vi, emacs, ...
		-- touch เปลี่ยนแปลงเวลาในการเข้าถึงแฟ้ม หากแฟ้มที่กำหนดไม่มีในระบบ จะทำการสร้างให้
	ลบ	-- rm (remove)
	คัดลอก	-- cp (copy)
	เปลี่ยนสิทธิการใช้งาน	-- chmod

คำสั่งจำเป็นต้องใช้งานร่วมกับ pathname
สัญลักษณ์แทน directory

~	home directory
.	current (working) directory
..	parent directory