http://staff.informatics.buu.ac.th/~jira/886326/Lectures/week01

สัปดาห์ที่ ๑

ตอนที่ ๑ เนื้อหาของรายวิชา 886326: Unix Tools and Programming 3(2-2-5)

ระบบปฏิบัติการที่ใช้ในการเรียนการสอน:

- ระบบปฏิบัติการชนิด Multitask/Multiuser สำหรับเครื่องคอมพิวเตอร์ส่วนบุคคล - Linux

ชื่อวิชาแสดงว่ามีองค์ประกอบ 2 ส่วน คือ

Tools -- ได้แก่โปรแกรมอรรถประโยชน์ (Utilities) ช่วยอำนวยความสะดวกในการใช้งานของผู้ใช้
คือโปรแกรมใน /bin, /usr/sbin, และ /usr/bin

Utilities บางตัวมีความสามารถในการทำงานมาก เรียกว่า Power Tools เช่น

- Editor เช่น vi และ emacs,
- Compiler เช่น gcc และ g++,
- Make - ใช้ทำงานตาม script สำหรับการ compile แยกส่วน หรือ Separated compilation,
- Debugger สำหรับติดตามและหาที่ผิดในการทำงาน เช่น gdb, และ
- Revision Control System สำหรับการพัฒนาโปรแกรมที่มีหลาย version

Programming -- การเขียน Shell script โดยใช้คำสั่งควบคุมโครงสร้างและตัวแปรของ shell ประกอบกับการทำงานของ Utilities เพื่อปรับแต่งการทำงาน หรือเพื่อขยายขอบขีดความสามารถของ Utilities ให้มีประสิทธิภาพสูงขึ้นและตรงกับความต้องการ

ตอนที่ ๒ ทบทวนระบบปฏิบัติการ

ระบบปฏิบัติการเป็น "แก่น" ของระบบคอมพิวเตอร์

- มุมมองที่มีต่อระบบระบบปฏิบัติการของผู้ใช้แต่ละคนไม่เหมือนกัน อาจมองเห็นเป็นของวิเศษ, เป็นอะไรก็ไม่รู้, เป็นอะไรที่งุนงงและ สับสนวุ่นวาย และผู้คนส่วนใหญ่ "กลัว" ระบบปฏิบัติการหน้าที่หลักของระบบปฏิบัติการคือการลดความซับซ้อนในการใช้งานคอมพิวเตอร์

- หน่วยประมวลผลกลาง (Central Processing Unit - CPU) ของคอมพิวเตอร์แตกต่างกัน

หน่วยประมวลผลกลาง Pentium, Core ix, AMD, ARM, MIPS

- ปริมาณหน่วยความจำหลักแตกต่างกัน- หน่วยเก็บข้อมูลความจุสูง (Mass storage) หรือหน่วยความจำรอง อาจมี ยี่ห้อ รุ่น ความจุ ความเร็วรอบหมุน เทคโนโลยีในการเชื่อมต่อ (interface) ของ hard drive แตกต่างกัน, ...
- อุปกรณ์ประกอบ (Peripheral devices) แตกต่างกัน เช่น
    - เมาส์ มีสาย ไม่มีสาย, sound card, กล้อง, ...
- เทคโนโลยีเครือข่ายแตกต่างกัน
    - เครือข่ายมีสาย, เครือข่ายไร้สาย (wifi), DSL, ...

คำถามที่น่าสนใจ

- เมื่อผู้เขียนโปรแกรมจะต้องเขียนโปรแกรมประยุกต์โปรแกรมหนึ่ง จำเป็นหรือไม่ที่ผู้เขียนโปรแกรมนั้นจะต้องเขียนคำสั่งสำหรับควบคุมบังคับการทำงานของอุปกรณ์แต่ละชิ้นด้วยตัวเอง
- ผู้เขียนโปรแกรมจำเป็นจะต้องปรับแต่งโปรแกรมให้เข้ากับอุปกรณ์แต่ละชิ้นหรือไม่? เช่นเมื่อมีการเปลี่ยน hard drive ที่มีการเชื่อมต่อ แบบ SATA เป็น USB hard drive จำเป็นจะต้องมีการแก้ไขโปรแกรมให้สอดคล้องกับฮาร์ดแวร์ที่เปลี่ยนไปหรือไม่?
- หากมีโปรแกรมใดโปรแกรมหนึ่งทำงานผิดพลาดควรจะมีผลให้โปรแกรมทั้งหมดในระบบทั้งระบบล่มตามไปด้วยหรือไม่?

หน้าที่ของระบบปฏิบัติการ

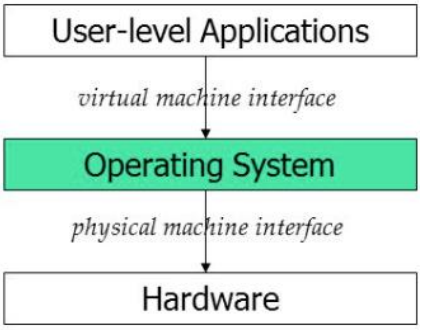
1. บริหารจัดการทรัพยากรทางกายภาพ (Physical resources)
    - ทำหน้าที่ "ขับ" หรือ "สั่งการ" ให้อุปกรณ์ต่าง ๆ ทำงาน เชน หน่วยประมวลผลกลาง หน่วยความจำ จานแม่เหล็ก แป้นพิมพ์ จอภาพ วงจรเชื่อมต่อเครือข่าย ...
    - จัดการทรัพยากรให้มีการใช้งานประสิทธิภาพ มีความเชื่อถือสูง ทนทานต่อการผิดพลาดของ (Fault tolerance) และการแยกแยะ และ จัดการกับความผิดพลาดที่เกิดขึ้น
2. สร้างสิ่งแวดล้อมที่เหมาะสมสำหรับการทำงานของโปรแกรมประยุกต์ เช่นโปรแกรมประมวลคำ และโปรแกรมตารางคำนวณ เป็นต้น
    - สร้าง virtual resource (ทรัพยากรเสมือนจริง?) และ virtual interface สำหรับให้บริการต่อโปรแกรมประยุกต์
    - ทำให้การเขียนโปรแกรมง่ายขึ้น ผ่าน high-level abstractions
    - สร้างสิ่งแวดล้อมที่มีเสถียรภาพสูง แยกแยะและจัดการความผิดพลาดได้

ระบบปฏิบัติการเป็นซอฟต์แวร์ที่ซับซ้อน

ระบบปฏิบัติการ เป็นโปรแกรมในกลุ่มที่มีความซับซ้อนสูงเป็นพิเศษ

- เป็นระบบขนาดใหญ่ มีการทำงานขนานกัน มีค่าใช้จ่ายในการพัฒนาสูงมาก และเป็นโปรแกรมที่มีผู้เข้าใจน้อย
- นอกจากระบบปฏิบัติการแล้ว โปรแกรมที่มีความซับซ้อนมาก เช่น โปรแกรมเครือข่าย เช่น TCP/IP, โปรแกรมสำหรับการทำงานของรัฐบาล, ระบบพยากรณ์อากาศ, ...
- จะพัฒนาโปรแกรมที่ซับซ้อนเช่นนี้ได้อย่างไร? -- Abstraction & layering
- เป้าหมาย: ระบบที่มีความเชื่อถือสูง สามารถเชื่อถือได้เต็มที่ เมื่อต้องดำเนินการกับข้อมูลที่เป็น sensitive data และ critical roles

ภาพโดยรวมของระบบปฏิบัติการ



Virtual Machine Abstraction

- ระบบปฏิบัติการทำหน้าที่เป็น virtual machine ซึ่งเป็นสิ่งแวดล้อมที่ง่ายกว่าและปลอดภัยกว่าสำหรับการเขียนโปรแกรมและการใช้งานมากกว่าที่จะดำเนินการกับฮาร์ดแวร์โดยตรง

องค์ประกอบ

- Kernel และ Utilities

- kernel เป็นตัวระบบปฏิบัติการ ควบคุม ดูแล และจัดสรรทรัพยากร
- utilities หรือโปรแกรมอรรถประโยชน์ เป้นโปรแกรมที่ใช้อำนวยความสะดวกในการทำงานของผู้ใช้

- Shell สำหรับเรียกใช้งาน Utilities และโปรแกรมประยุกต์ (Application programs)

- รับคำสั่งจากผู้ใช้ ตีความ และปฏิบัติ หรือทำหน้าที่เป็นตัวแปลคำสั่ง Command Interpreter

ประเภทของ Shell

- Graphic User Interface (GUI) -- X-Window

  Window manager -- 180 ชนิด ดูรายละเอียดที่ http://www.gilesorr.com/wm/table.html

- Command Line Interface (CLI) -- Bourne Again Shell (/bin/bash), TC shell (/usr/bin/tcsh)

Command Line Interface

- Interactive mode สั่งงานผ่าน Command Line

  Format: command [-option] <argument-list>

- สามารถนำคำสั่งหลายคำสั่งมาทำงานต่อกันเป็น pipelining ได้ (i/o redirection)

  เช่น a | b คือ การนำ output ของ a เป็น input ของ b- เปลี่ยนทิศทาง input/output ได้

- เป็นภาษาสำหรับเขียนโปรแกรม Shell script

  Shell script ไม่ได้ออกแบบมาสำหรับการเขียนโปรแกรมประยุกต์โดยทั่วไป แต่ออกแบบมาสำหรับเขียนโปรแกรม ช่วยในการทำงานของระบบปฏิบัติการ

- shell มีโครงสร้างควบคุมการทำงาน เนื้องานมาจาก utilities

ตอนที่ ๓ ระบบปฏิบัติการชนิด Multitask/Multiuser

Multitask - ระบบปฏิบัติการในปัจจุบันเกือบทุกตัวเป็น multitask หรือ multiprocess หมดแล้ว

- Unix, Linux, และ Android สำหรับอุปกรณ์เคลื่อนที่ (Mobile devices) เช่น smart phone, tablet, netbook
- Microsoft Windows และ Windows Phone สำหรับ smart phone
- Mac OS และ IOS สำหรับอุปกรณ์เคลื่อนที่ ได้แก่ iPhone และ iPad

หน่วยประมวลผลกลางตัวเดียว ให้บริการโพรเซส หรือ งาน (task) หลายงานในขณะเดียวกัน

- แบ่งเวลา (timesharing) ออกเป็นหน่วยย่อย (time slice หรือ time quantum) แต่ละหน่วยเวลามีเพียงโพรเซสเดียวที่ได้ใช้งาน CPU เมื่อหมดเวลาบังคับตัดตอน (preempt) เพื่อสลับให้โพรเซสอื่นทำงาน

Multiuser - มีผู้ใช้ "หลาย" คน แต่ละคนมีอาจมีสิทธิในการใช้ทรัพยากรต่างกัน

- Unix, Microsoft Windows Server, Mac OS

เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นแม่ข่าย (host) มีเพียงเครื่องเดียว มีแป้นพิมพ์ จอภาพ (และเมาส์) เพียงชุดเดียว เพื่อให้สามารถให้บริการผู้ใช้หลายคนในขณะเดียวกัน ผู้ใช้แต่ละคนจึงต้องมีแป้นพิมพ์ และจอภาพเป็นของตนเอง และมีอุปกรณ์สื่อสารแบบอนุกรมเชื่อมต่อไปยังเครื่องแม่ข่าย

- แป้นพิมพ์ จอภาพ และเมาส์ ที่เป็นของเครื่องแม่ข่าย เรียกว่า Console สามารถใช้งานได้ทั้ง Text และ Graphic mode (X-Window) ส่วนใหญ่ใช้เป็นหน้าจอสำหรับผู้ดูแลระบบ

- แป้นพิมพ์ จอภาพ และอุปกรณ์สื่อสาร สำหรับผู้ใช้แต่ละคนเรียกว่า Terminal ซึ่งไม่มีการผลิตจำหน่ายแล้ว เพราะสามารถใช้งาน โปรแกรมในกลุ่มที่เรียกว่า Terminal emulation สำหรับใช้จอภาพและแป้นพิมพ์ของเครื่องคอมพิวเตอร์ส่วนบุคคล จำลองเป็นหน้าจอ และเชื่อมต่อผ่านเครือข่ายอินเทอร์เน็ตไปยังแม่ข่าย Terminal ที่เกิดขึ้นจากการใช้ซอฟต์แวร์จำลองนี้เรียกว่า Pseudo-terminal

Shell สำหรับระบบปฏิบัติการ Unix มีสองแบบคือ

- Command Line Interface (CLI) เช่น Bourne Shell (sh), C Shell (csh), Korn Shell (ksh) และ Bourne Again Shell (bash)

- Graphic User Interface (GUI) ได้แก่ระบบ X-Window ซึ่งมีโปรแกรมที่ทำหน้าที่เป็น window manager จำนวนมากให้เลือกใช้

โปรแกรมในกลุ่ม Terminal จึงมีสองแบบคือ

- Terminal ทำงานได้เฉพาะกับ Command Line Interface (CLI) ไม่สามารถแสดงผลกราฟิกได้ โปรแกรม terminal emulation ในกลุ่มนี้ที่มีการใช้งานในคณะ ได้แก่ SSH Secure Shell Client, Tera Term. และ PuTTy

- Terminal ที่ทำงานได้กับ Graphic User Interface (GUI) หรือทำงานได้กับระบบ X-Window เรียกว่า X-terminal (xterm) ซึ่งมีโปรแกรมจำลองหน้าจอให้เลือก download สำหรับใช้งานเป็นจำนวนมาก

ผู้ใช้ (User) และกลุ่ม (Group)

ผู้ใช้ (user) คือ entity ที่สร้างและเป็นเจ้าของโพรเซสได้ หรือเป็น entity ที่เป็นเจ้าของแฟ้มได้

จากนิยามจะเห็นว่าผู้ใช้ หรือ user จึงไม่จำเป็นต้องเป็นตัวบุคคลเสมอไป อาจเป็นองค์ประกอบของระบบปฏิบัติการที่ทำหน้าที่ในการสร้างโพรเซสที่จำเป็นเมื่อระบบเริ่มทำงานก็ได้ เช่นผู้ใช้ daemon, bin และ sys เป็นต้น อย่างไรก็ดีผู้ใช้ส่วนใหญ่ในระบบเป็นผู้ใช้ที่เป็นบุคคล

ผู้ใช้แต่ละรายมีชื่อผู้ใช้ (นิยมเรียกว่า login name) และหมายเลขเฉพาะตัวของ ผู้ใช้ (user identification number) เรียกย่อว่า "uid" นอกจากนี้ยังประกอบด้วย สถานะของรหัสผ่าน, หมายเลขกลุ่มหลัก, ข้อมูลรายละเอียดของผู้ใช้, ไดเรกทอรีเริ่มต้น (home directory),และเชลล์เริ่มต้น

ระบบปฏิบัติการชนิด Multiuser "ต้อง" สนับสนุนการทำงานและการใช้ทรัพยากรร่วมกันระหว่างผู้ใช้ โดยการจัดให้ผู้ใช้ที่มีลักษณะการทำงานคล้ายกันเข้าเป็น "กลุ่ม" เดียวกัน เพื่อความสะดวกในการกำหนดสิทธิในการใช้ทรัพยากร

กลุ่ม (group) เป็นผู้ใช้จำนวนหนึ่งที่มีลักษณะการทำงานคล้ายกัน และมีการใช้ทรัพยากรร่วมกัน

จากนิยามการจัดให้ผู้ใช้ที่มีลักษณะการทำงานคล้ายกันเข้าเป็น "กลุ่ม" เดียวกัน ทำให้สะดวกในการกำหนดสิทธิในการใช้ทรัพยากรโดยเมื่อกำหนดสิทธิให้กับกลุ่ม ผู้ใช้ทุกคนในกลุ่มจะได้รับสิทธินั้นด้วย ตัวอย่างเช่น กลุ่มของโปรแกรมเมอร์ที่อยู่ในโครงการพัฒนาซอฟต์แวร์เดียวกัน จำเป็นต้องใช้แฟ้มโปรแกรมร่วมกัน

ผู้ใช้แต่ละคนต้องสังกัดกลุ่มหลัก 1 กลุ่ม เรียกว่า Primary group และอาจสังกัดกลุ่มอื่นๆ อีกหลายกลุ่ม ได้ตามความจำเป็นของงาน การเปลี่ยนกลุ่ม การเพิ่มและลดสมาชิกในกลุ่ม เป็นความรับผิดชอบของผู้ดูและระบบ

กลุ่มแต่ละ group มีชื่อกลุ่ม และหมายเลขกลุ่ม (group identification number) เรียกย่อว่า "gid" ระบบปฏิบัติการนำ uid และ gid ไปใช้ในการพิจารณาสิทธิของผู้ใช้สำหรับการใช้งานแฟ้ม และบริการของระบบ

การเข้าใช้งานและการพิสูจน์ตัวจริง

เพื่อความปลอดภัยในการใช้งาน ผู้ที่จะสามารถใช้งานได้ ต้องมีการลงทะเบียนเพื่อใช้ใช้งานในระบบ แต่ละรายการเรียกว่า user's account ประกอบด้วย ชื่อผู้ใช้ (login name) และรหัสผ่าน (password) เมื่อจะเข้าใช้งาน ผู้ใช้ต้องป้อนชื่อผู้ใช้และรหัสผ่านเพื่อเป็นการพิสูจน์ยืนยันว่าเป็นผู้ใช้ที่อยู่ในบัญชีรายชื่อจริง กระบวนการนี้เรียกว่า "การพิสูจน์ตัวจริง" (Authentication)

- เมื่อผ่าน "การพิสูจน์ตัวจริง" แล้ว ผู้ใช้จะเข้าสู่พื้นที่ทำงานส่วนตัว เรียกว่า Home directory (~) ผู้ใช้แต่ละคนมี home directory ต่างกัน

- home directory ของผู้ใช้ใดเป็นสิทธิโดยสมบูรณ์ของผู้ใช้นั้น และสามารถอนุญาตให้ผู้ใช้อื่นใช้งานได้ โดยใช้คำสั่ง chmod (change permission mode) - เพื่อให้ผู้ใช้สามารถใช้ทรัพยากรร่วมกันได้ เช่น การใช้แฟ้มโปรแกรมร่วมกันในระหว่างทีมงานพัฒนาโปรแกรม

- เมื่อผู้ใช้จะใช้ทรัพยากรใด ระบบจะทำการตรวจสอบว่าผู้ใช้มีสิทธิในการใช้งานทรัพยากรนั้นหรือไม่ และเป็นสิทธิใด

- ทรัพยากรแต่ละอย่างมีเจ้าของ (owner) - ความเป็นเจ้าของ (ownership) ทำให้ผู้ใช้มี "ความเป็นส่วนตัว" มีความปลอดภัย (Security) ในการใช้งาน

ผู้ใช้พิเศษในระบบปฏิบัติการ Unix

ระบบปฏิบัติการ Unix มีผู้ใช้พิเศษเรียกว่า Superuser ทำหน้าที่เป็นผู้ดูแลระบบ (system administration) มีชื่อผู้ใช้เป็น root มีสิทธิในการใช้ทรัพยากรทุกอย่างในระบบ เป็นผู้สร้างและกำหนดรายละเอียดการทำงานของผู้ใช้อื่น

ชื่อ root สันนิษฐานว่ามาจาก superuser เป็นผู้ใช้เพียงรายเดียวที่สามารถเปลี่ยนแปลงแก้ไขระบบแฟ้มใน root directory ได้ และมี home directory เป็น root directory (/)

แฟ้มและคำสั่งที่เกี่ยวข้อง

/etc/passwd	แฟ้มรายชื่อและรายละเอียดผู้ใช้, ผู้ใช้ทั่วไปมีสิทธิอ่านข้อมูลในแฟ้มได้
/etc/shadow	แฟ้มเก็บรหัสผ่านที่เข้ารหัสแล้ว, เฉพาะ root และผู้ใช้กลุ่ม shadow เท่านั้นที่มีสิทธิอ่านแฟ้ม
/etc/group	แฟ้มเก็บรายชื่อและรายละเอียดของกลุ่ม, ผู้ใช้ทั่วไปมีสิทธิอ่านข้อมูลในแฟ้มได้

อ่านรายละเอียดของแฟ้มได้ในหมวด 5: File Formats and Conversions เช่น

```
$ man -s5 passwd                # online manual page ในหมวด (section) 5 สำหรับแฟ้ม passwd
```

คำสั่งสำหรับตรวจสอบหมายเลขผู้ใช้และหมายเลขกลุ่ม

id – แสดงหมายเลขผู้ใช้และหมายเลขกลุ่ม

คำสั่งสำหรับเปลี่ยนแปลงสิทธิ เจ้าของ และกลุ่ม

chmod	คำสั่งเปลี่ยนสิทธิการใช้งานแฟ้มหรือไดเรกทอรี (change file mode bits)
chown	คำสั่งเปลี่ยนเจ้าของแฟ้มหรือไดเรกทอรี (change file owner and group)
chgrp	คำสั่งเปลี่ยนกลุ่มของแฟ้มหรือไดเรกทอรี (change group ownership)

ข้อควรระวัง

หากใช้คำสั่ง chown เพื่อเปลี่ยนเจ้าของแฟ้มจากตนเองเป็นของผู้อื่นแล้ว จะไม่สามารถเปลี่ยนกลับคืนมาเป็นของตนเองได้ต้องให้เจ้าของใหม่ หรือ superuser ดำเนินการให้

ความสัมพันธ์ระหว่างผู้ใช้และแฟ้ม

แฟ้มแต่ละแฟ้มมีผู้ใช้ที่เป็น "เจ้าของ" (owner) เพียงคนเดียว และมี "กลุ่ม" (group) ที่เป็น "เจ้าของ" เพียงกลุ่มเดียว, ผู้ใช้ที่เป็น "เจ้าของ" แฟ้ม เป็นผู้เดียวที่สามารถเปลี่ยนแสิทธิในการใช้แฟ้มในได้ เปลี่ยนเจ้าของและกลุ่มของแฟ้มได้ การที่ระบบปฏิบัติการกำหนดไว้เช่นนี้ ช่วยทำให้ผู้ใช้มั่นใจได้ว่าจะไม่มีผู้ใช้อื่นเข้ามาใช้งานแฟ้มของตนโดยไม่ได้รับอนุญาต

การกำหนดสิทธิ์ (permission) ในการใช้งานแฟ้ม (และไดเรกทอรี) ให้กับผู้ใช้ ระบบปฏิบัติการ Unix แบ่งผู้ใช้เป็น 3 กลุ่ม คือ

- สิทธิของเจ้าของ (owner permission) เป็นสิทธิที่เจ้าของแฟ้มจะสามารถดำเนินการกับแฟ้มนั้นได้
- สิทธิของกลุ่ม (group permission) เป็นสิทธิที่ผู้ใช้ที่อยู่ในกลุ่มเดียวกับเจ้าของแฟ้มนั้น จะสามารถดำเนินการกับแฟ้มนั้นได้
- สิทธิของผู้ใช้ทั่วไป (other permission) เป็นสิทธิที่ผู้ใช้ทุกคนที่อยู่ในระบบนั้น จะสามารถดำเนินการกับแฟ้มนั้นได้

และกำหนดสิทธิในการใช้งานแฟ้ม (และไดเรกทอรี) ของผู้ใช้แต่ละกลุ่มไว้สามแบบคือ

- read (r) สิทธิในการอ่าน
- write (w) สิทธิในการเขียน
- execute (x) สิทธิในการ run แฟ้มโปรแกรม หรือสิทธิในการใช้ไดเรกทอรีสิทธิทั้งสามแบบนี้มีความหมายต่างกัน เมื่อใช้กับแฟ้มและไดเรกทอรี โดยมีรายละเอียดดังนี้

สิทธิสำหรับการใช้งานแฟ้ม (file access permission)

- read (r) : สามารถอ่านข้อมูลในแฟ้มนั้นได้ เช่น ใช้คำสั่ง cat กับแฟ้มนั้นได้
- write (w) : สามารถเขียน เปลี่ยนแปลง และลบข้อมูลในแฟ้มนั้นได้
- execute (x) : สามารถ run โปรแกรมในแฟ้มนั้นได้ มีผลเฉพาะแฟ้มคำสั่งทั้งแฟ้มคำสั่งภาษาเครื่อง, shell script, และแฟ้มภาษา script อื่นๆ

หมายเหตุ หากถอนสิทธิ x จากแฟ้ม a.out จะไม่สามารถเรียกให้ทำงานได้อีก โดยระบบจะแจ้งให้ทราบว่าผู้ใช้ไม่มีสิทธิ

สิทธิสำหรับการใช้งานไดเรกทอรี (directory access permission)

- read (r) : สามารถอ่านรายชื่อแฟ้มในไดเรกทอรีได้ เช่น สามารถใช้คำสั่ง ls ในไดเรกทอรีนั้นได้

- write (w) : สามารถสร้างแฟ้มใหม่ หรือลบแฟ้มในไดเรกทอรีได้ ซึ่งหมายความว่าหากผู้ใช้มีสิทธิ w ในระดับไดเรกทอรี ผู้ใช้นั้นสามารถลบแฟ้มในไดเรกทอรีนั้นได้ โดยไม่จำเป็นต้องมีสิทธิ w ในระดับแฟ้ม

- execute (x) : สามารถเข้าในไดเรกทอรีนั้นได้ เช่น สามารถใช้คำสั่ง cd เข้าในไดเรกทอรีนั้นได้

การกำหนดสิทธิในการใช้งาน

สิทธิการใช้งานของผู้ใช้แต่ละกลุ่มเรียงลำดับ read, write, execute และเรียงลำดับจาก เจ้าของ, กลุ่ม, และผู้ใช้อื่นในระบบเสมอ สิทธิการใช้งานแต่ละอย่างแทนด้วยเลขฐานสองขนาด 1 บิต จึงมี 2 สถานะ คือ "มีสิทธิ" (แทนด้วย 1) หรือ "ไม่มีสิทธิ" (แทนด้วย 0) สิทธิของผู้ใช้แต่ละกลุ่มจึงแทนด้วยเลขฐานสองขนาด 3 บิต ซึ่งสมนัยกับเลขฐานแปด (Octal) 1 หลัก

เลขฐานสอง (Binary)	เลขฐานแปด (Octal)	สิทธิการใช้งาน (Permission)
000	0	---
001	1	--x
010	2	-w-
011	3	-wx-
100	4	r--
101	5	r-x
110	6	rw-
111	7	rwx

คำถาม จากสิทธิทั้งหมดของผู้ใช้แต่ละกลุ่มที่เป็นไปได้ทั้งหมด 8 แบบ เป็นการกำหนดที่สมเหตุสมผลทุกแบบหรือไม่? จงอภิปรายพร้อมให้เหตุผลประกอบที่ชัดเจน