CS222

Operating System

Lab 1: Basic Linux Tutorial

1st Semester 2021

Kasidit Chanchio

Department of Computer Science

Faculty of Science and Technology

Thammasat University

https://en.wikipedia.org/wiki/Ken_Thompson

UNIX

- เป็น OS แบบมัลติทาสกิ้งและมัลติยูเซอร์
- พัฒนาขึ้นในปี 1969 ที่ AT&T's Bell Lab โดย
  - Ken Thomson และ Dennis Ritchie (ผู้สร้างภาษา C)
- มีหลายชนิด
  - System V, Solaris, SCO UNIX, SunOS, AIX
  - BSD, FreeBSD, OpenBSD
- ในช่วง 90 มีเหตุการณ์ UNIX WARS ทำให้พัฒนาการหยุดชะงัก
  - https://en.wikipedia.org/wiki/Unix_wars

https://www.facebook.com/AssociationForComputingMachinery/photos/a.450674248151/10153632223513152/?type=3

LINUX

https://www.neowin.net/news/linus-torvalds-accepts-microsofts-linux-hyper-v-upgrades-so-both-intel-and-amd-can-benefit/

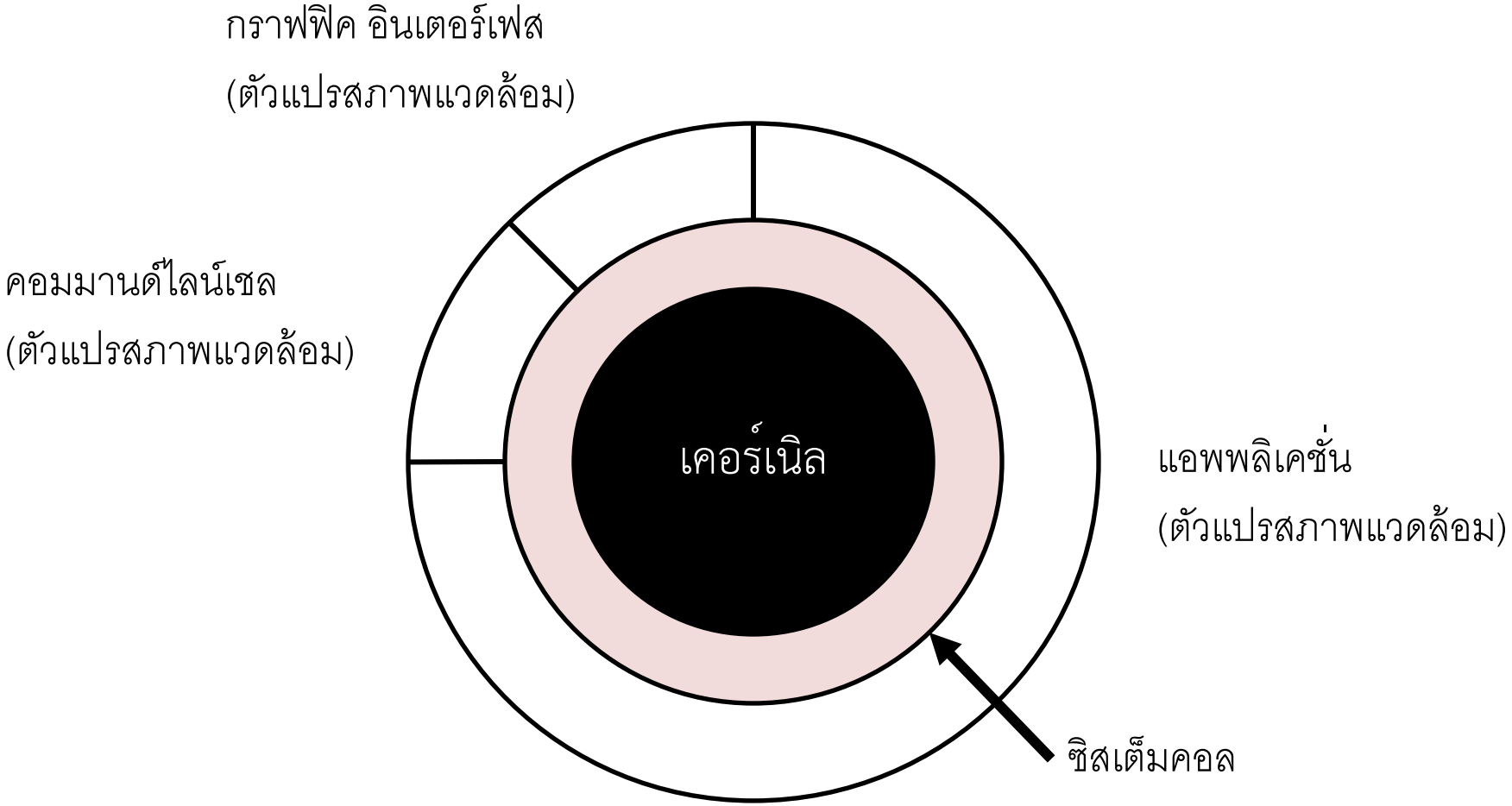
https://fossbytes.com/linus-torvaldss-famous-email-first-linux-announcement/

- เป็น Clone ของ UNIX สำหรับรันบน x_86 ISA
- พัฒนาขึ้นโดย Linus Trovalds ซึ่งเป็น นศป.เอก ในเวลานั้น ปัจจุบัน Linus เป็นซอฟต์แวร์เอนจิเนียของ Linux Foundation นอกจาก Linux และ Linus ยังเป็นผู้สร้างระบบ Git version control ซึ่งกันอย่างแพร่หลาย
- ได้รับแรงบันดาบใจจากระบบ Minix ของ Andrew Tanenbuam
- เริ่มต้นพัฒนาสำหรับ 32 bits x_86 PC
- ปัจจุบันใช้บน Supercomputer ถึง มือถือ (Android)
- มีหลาย Distribution เช่น Ubuntu, Redhat, Fedora, etc. กว่าสามร้อย

ส่วนประกอบ

- ประกอบไปด้วย (1) เคอร์เนิล (2) เชล/กุย (3) ระบบไฟล์ และ (4) สภาพแวดล้อม
- โครงสร้างของระบบไฟล์เป็นแบบ Hierarchy ประกอบไปด้วยเอ็นติตี้สองแบบได้แก่ ไดเรกทอรี่ และไฟล์
  - ไดเรกทอรี่คือที่เก็บไฟล์หรือไดเรกทอรี่
  - ไฟล์ คือจุดอ้างอิงของสิ่งที่อยู่ภายในระบบคอมพิวเตอร์ เช่นข้อมูล ฮาร์ดแวร์สำหรับประมวลผล หน่วยความจำ อุปกรณ์ไอโอ และโปรเซส เป็นต้น
- การอ้างอิงชื่อจะอ้างอิงโครงสร้างไดเรกทอรี่
- มีตำแหน่งปัจจุบันในไดเรกทอรี่สำหรับการใช้งานของผู้ใช้แต่ละครั้ง

ส่วนประกอบของลินุกซ์



ขั้นตอนการบูท

- ระบบคอมพิวเตอรรีเซตค่าในเครื่อง
- กำหนดค่าเริ่มต้นให้รีจิสเตอร์และอุปกรณ์
- ซีพียู อ่านชุดคำสั่งจาก BIOS เข้ามาประมวลผล **หรือ**โหลด UEFI โปรแกรมจาก non-volatile storage เข้าสู่หน่วยความจำ และเริ่มกระบวนการบูท
- กำหนดค่าชุดคำสั่งเพื่อเข้าถึงอุปกรณ์พื้นฐานเช่น คีบอร์ด หน้าจอ ฮาร์ดดิสค์
- โหลดโอเอสจากฮาร์ดดิสค์หรืออุปกรณ์เก็บข้อมูลอื่น และสั่งให้ซีพียูรันโอเอส
- โอเอสจะกำหนดค่าให้กับอินเตอร์รับแฮนด์เลอร์และโหลดโค๊ดของเคอร์เนิลเข้าสู่หน่วยความจำ
- โอเอสรันโปรแกรมล็อกอิน หรือกราฟิกอินเตอร์เฟสเพื่อรอรับการล็อกอินของผู้ใช้
- โอเอสรันโปรแกรมคอมมานด์ไลน์เชล หรือกราฟิกอินเตอร์เฟสเพื่อรอรับคำสั่ง

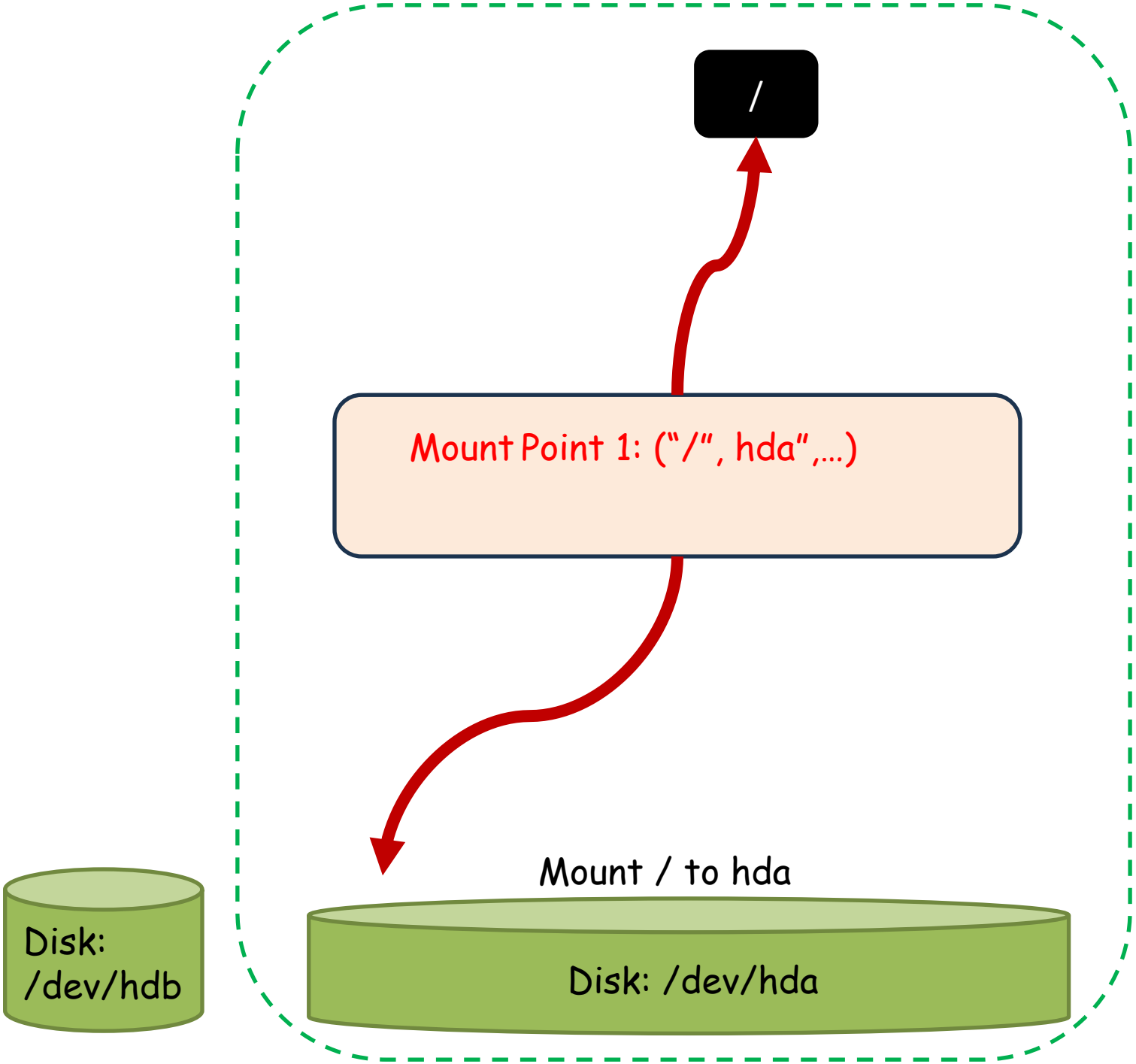
ชื่อไดเรกทอรี่และไฟล์

- ในระบบไฟล์ (File Systems) ของลินุกซ์ มีการให้ชื่อเอ็นติตี้ (entity) สองแบบ ได้แก่ absolute pathname และ relative pathname
- Absolute pathname คือการอ้างอิงถึงชื่อเต็มของเอ็นติตี้โดยอ้างอิงจาก รูท ของโครงสร้างระบบไดเรกทอรี่และไฟล์
- Relative pathname คือการอ้างอิงถึงชื่อของเอ็นติตี้โดยอ้างอิงจากตำแหน่งปัจจุบันที่ผู้ใช้ปฏิบัติงานในโครงสร้างระบบไดเรกทอรี่และไฟล์
- ชื่อแรกในระบบฯคือ รูท (root) ไดเรกทอรี่มีสัญลักษณ์คือ '/'
- ในภาพ รูทไดเรกทอรี่มี ไดเรกทอรี่ย่อย ยกตัวอย่างเช่น bin dev etc usr home lib sbin tmp และ var
- ไดเรกทอรี่เช่น /dev /sys และ /proc เรียกว่า pseudo file system ไม่ใช่ที่เก็บข้อมูลจริงแต่เป็น ช่องทางในการเรียกดูข้อมูลของ kernel และกำหนดค่าพารามิเตอร์ให้ส่วนประกอบของ kernel

ความสัมพันธ์ระหว่างระบบไฟล์กับอุปกรณ์เก็บข้อมูล 1

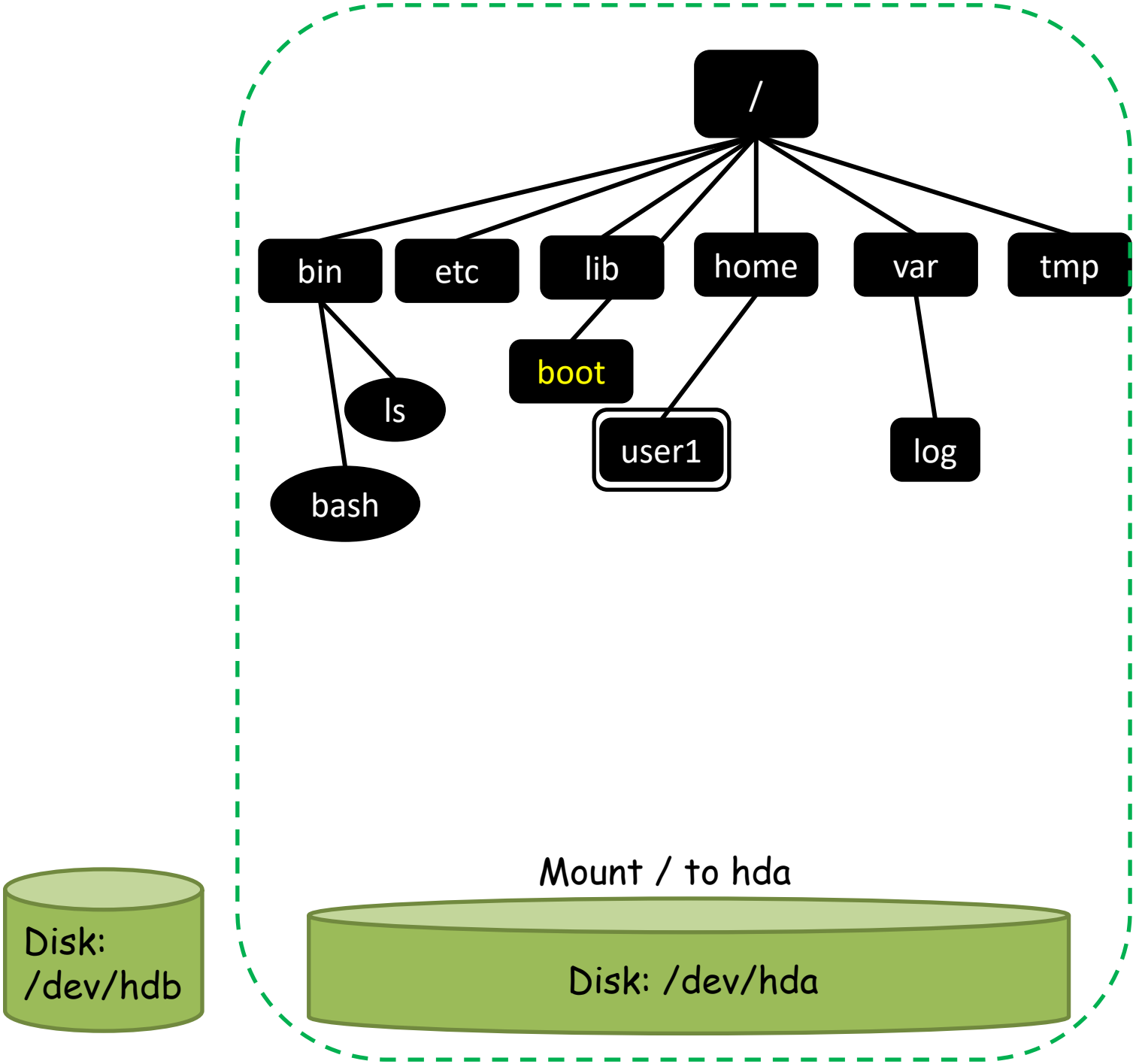
- เริ่มต้นที่สมมุติว่ามี Disk อยู่สองก้อนได้แก่ /dev/hda และ /dev/hdb
- Disk จะต้องได้รับการ format ถึงจะใช้งานได้ สมมุติว่าเรียบร้อยแล้ว
- ไฟล์ซิสเต็มเป็นจินตนาการของการจัดระเบียบการเก็บข้อมูล เริ่มต้นจากสิ่งที่เรียกว่า Root ไดเรกทอรี่หรือ “/”
- UNIX/LINUX ใช้ระบบไดเรกทอรี่แบบแผนภาพต้นไม้ที่ประกอบไปด้วยไดเรกทอรี่และไฟล์ ไดเรกทอรี่คือที่ๆใช้เก็บไฟล์หรือไดเรกทอรี่
- เราจะรู้ได้อย่างไรว่าข้อมูลในไดเรกทอรี่ถูกเก็บอยู่บน Disk /อุปกรณ์ไหน
- OS จะกำหนดว่าไดเรกทอรี่และข้อมูลของมันอยู่บน Disk ไหน โดยใช้ Mount point = (**ชื่อไดเรกทอรี่, ชื่ออุปกรณ์**, ข้อมูลอื่นๆ)

/
Disk:
/dev/hdb
Disk: /dev/hda


/
Mount Point 1: ("/", hda",...)
Mount / to hda
Disk: /dev/hda
Disk: /dev/hdb

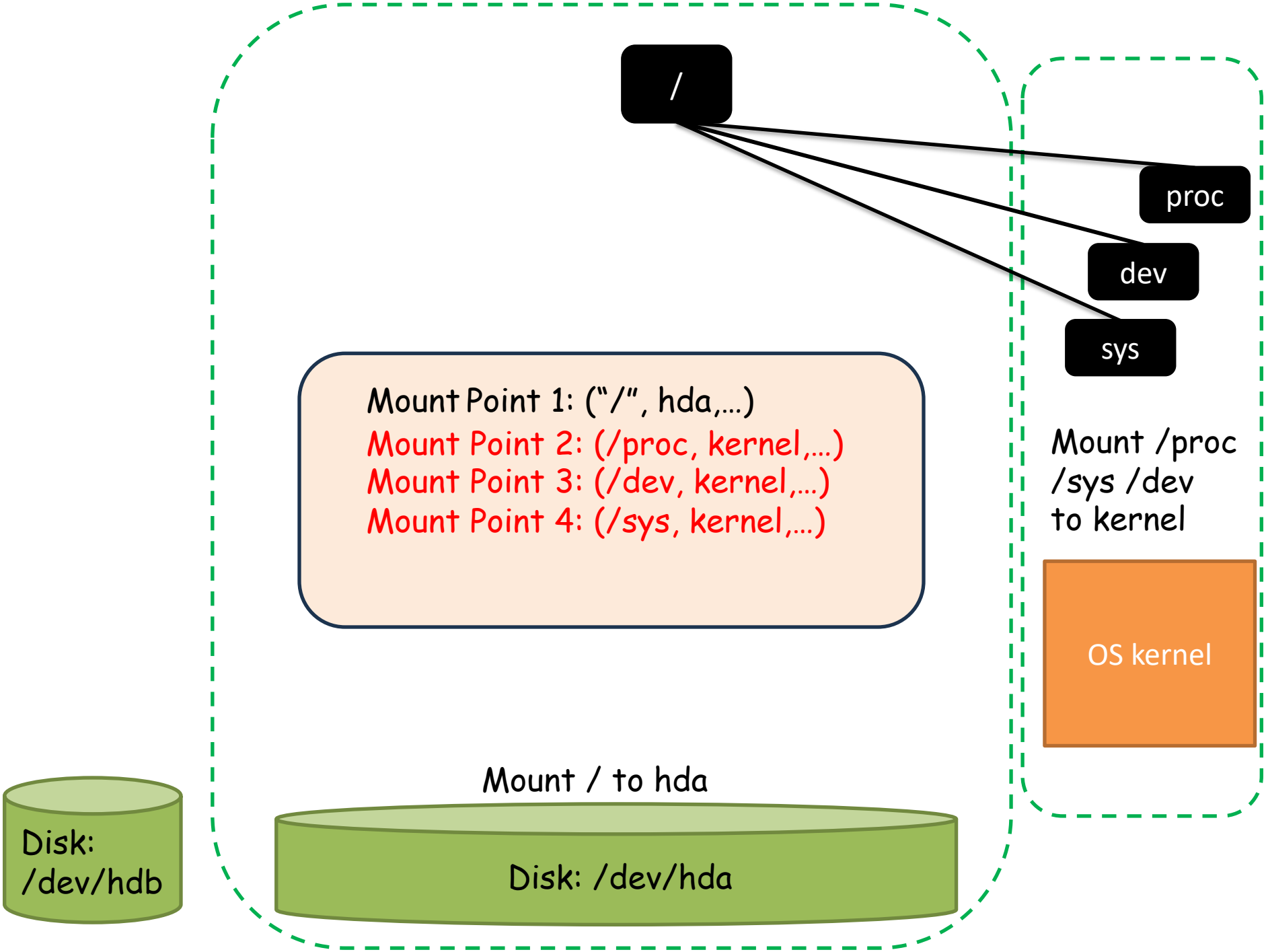
ความสัมพันธ์ระหว่างระบบไฟล์กับอุปกรณ์เก็บข้อมูล 2

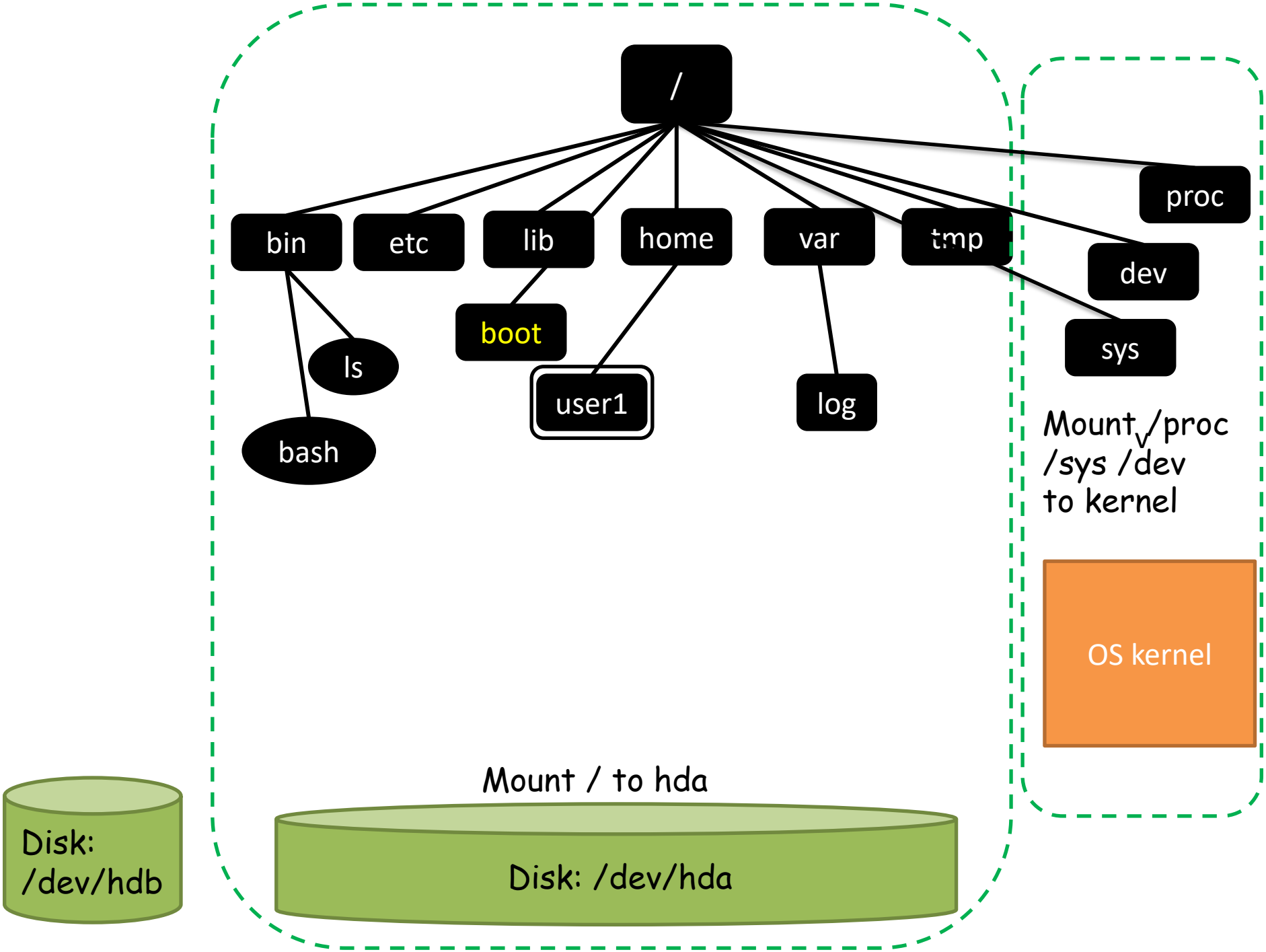
- Mount point 1 = (”/”, hda, ข้อมูลอื่นๆ)
- หลังจากนั้น เมื่อเราติดตั้ง OS โปรแกรมที่ติดตั้ง (ใน ISO disk image) จะสร้างไดเรกทอรี่และไฟล์ต่างๆที่จำเป็น
  - มันจะเก็บ linux kernel code ใน /boot ไดเรกทอรี่
- มันจะสร้าง account และพื้นที่เก็บข้อมูลของ user เริ่มต้น เช่น user1 ในไดเรกทอรี่ /home/user1


/
bin
etc
lib
home
var
tmp
boot
ls
user1
log
bash
Mount / to hda
Disk: /dev/hdb
Disk: /dev/hda

ความสัมพันธ์ระหว่างระบบไฟล์กับอุปกรณ์เก็บข้อมูล 3

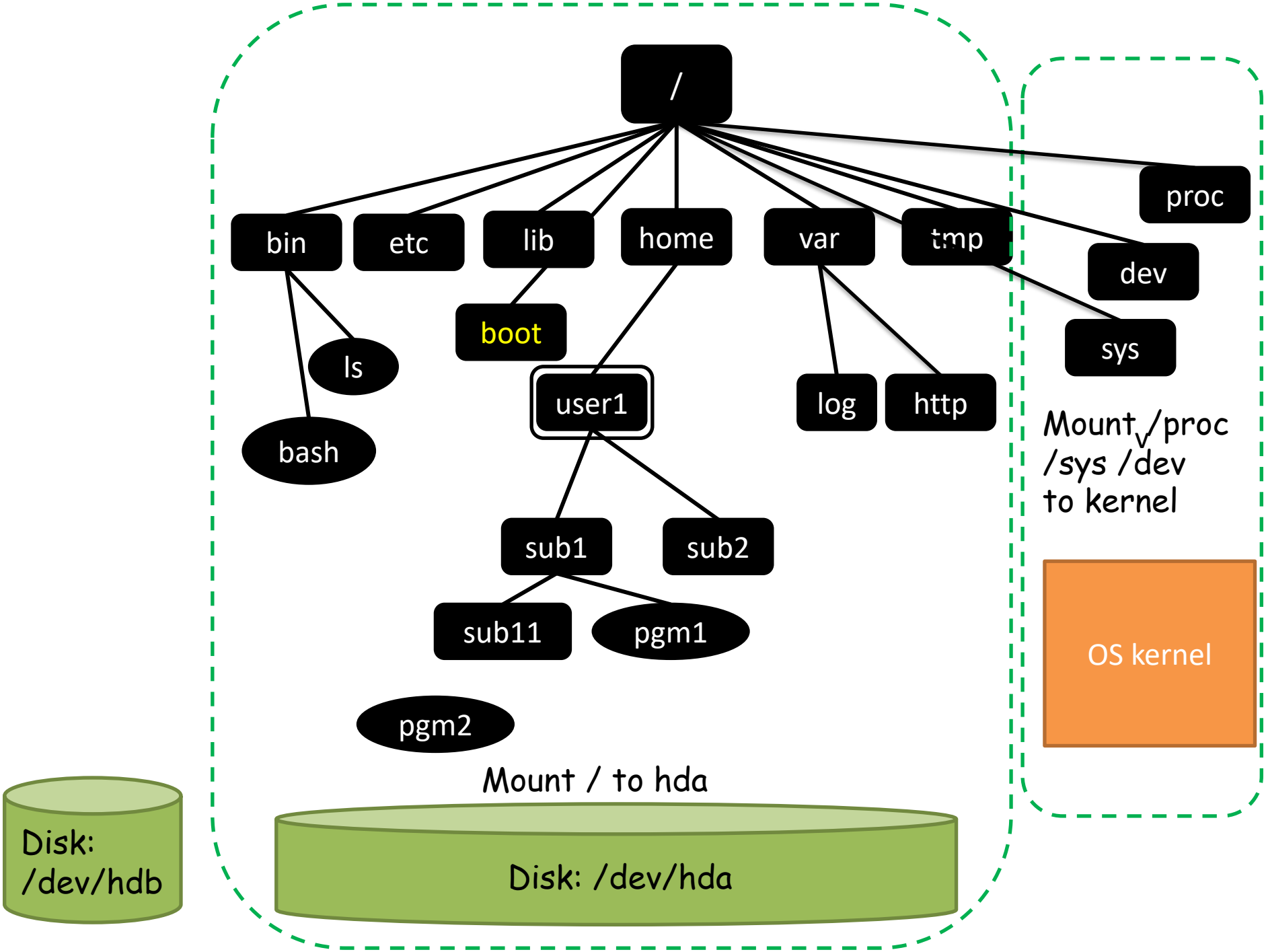
- OS kernel จะต้องเปิดช่องทางให้ผู้ใช้เข้าถึงข้อมูลและควบคุม OS kernel โดยใช้นามธรรมของไฟล์ซิสเต็ม
- Linux ใช้ Mount point เป็นเครื่องมือเพื่อทำให้ผู้ใช้ทำดังนั้นได้โดย
- เปิดช่องทางให้ผู้ใช้เข้าถึงข้อมูลของโพรเซสและควบคุมโพรเซส
  - Mount point 2 = (”/proc”, pseudo-fs-kernel, ข้อมูลอื่นๆ)
- เปิดช่องทางให้ผู้ใช้เข้าถึงข้อมูลของอุปกรณ์และควบคุมอุปกรณ์
  - Mount point 3 = (”/dev”, pseudo-fs-kernel, ข้อมูลอื่นๆ)
- เปิดช่องทางให้ผู้ใช้เข้าถึงข้อมูลค่าคอนฟิกและควบคุมค่าคอมพิกของ OS
  - Mount point 4 = (”/sys”, pseudo-fs-kernel, ข้อมูลอื่นๆ)


/
proc
dev
sys
Mount Point 1: ("/", hda,...)
Mount Point 2: (/proc, kernel,...)
Mount Point 3: (/dev, kernel,...)
Mount Point 4: (/sys, kernel,...)
Mount /proc /sys /dev to kernel
OS kernel
Mount / to hda
Disk: /dev/hdb
Disk: /dev/hda


/
bin
etc
lib
home
var
tmp
proc
dev
sys
boot
ls
bash
user1
log
Mount /proc
/sys /dev
to kernel
OS kernel
Mount / to hda
Disk:
/dev/hdb
Disk: /dev/hda

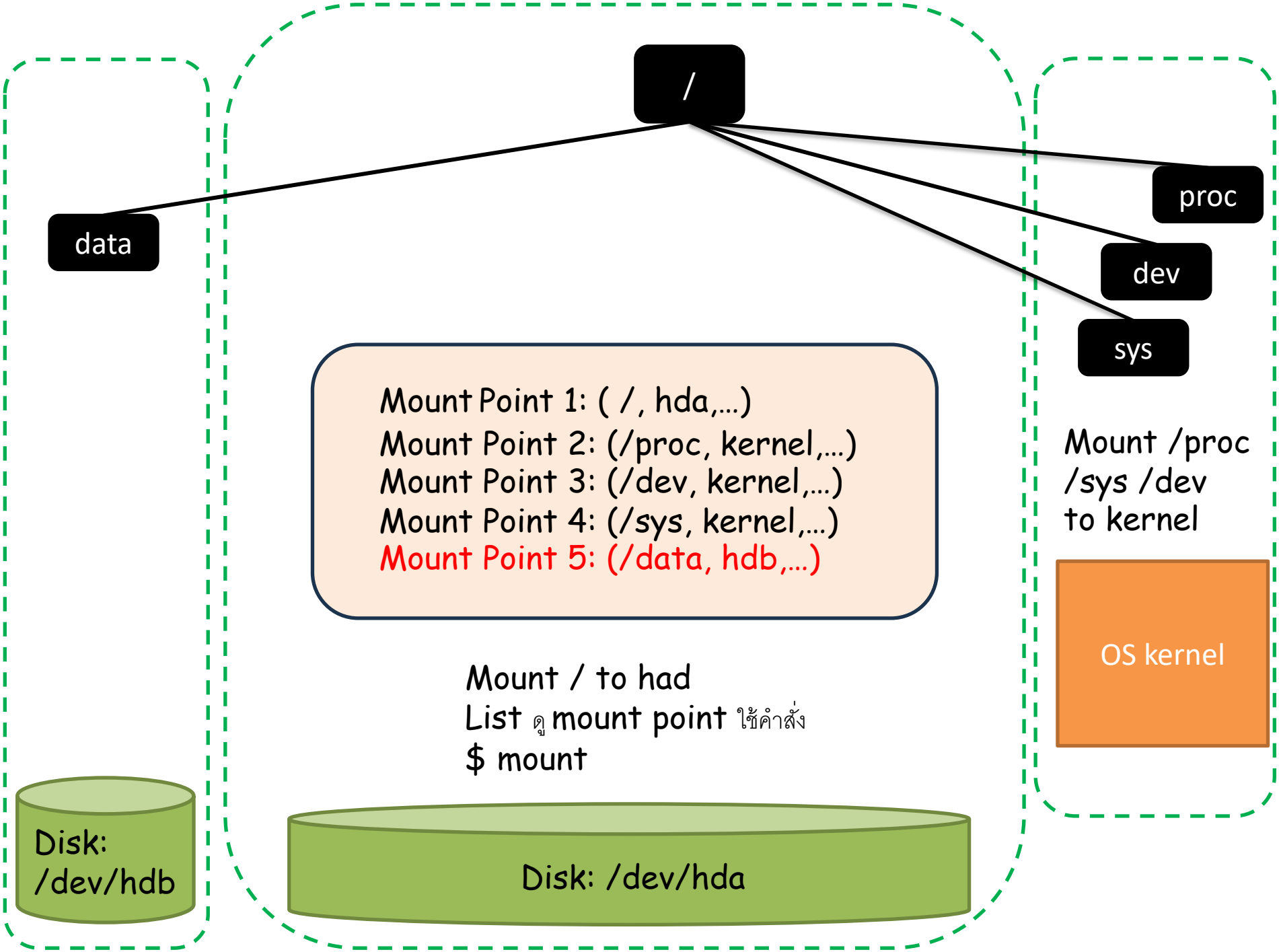
ความสัมพันธ์ระหว่างระบบไฟล์กับอุปกรณ์เก็บข้อมูล 4

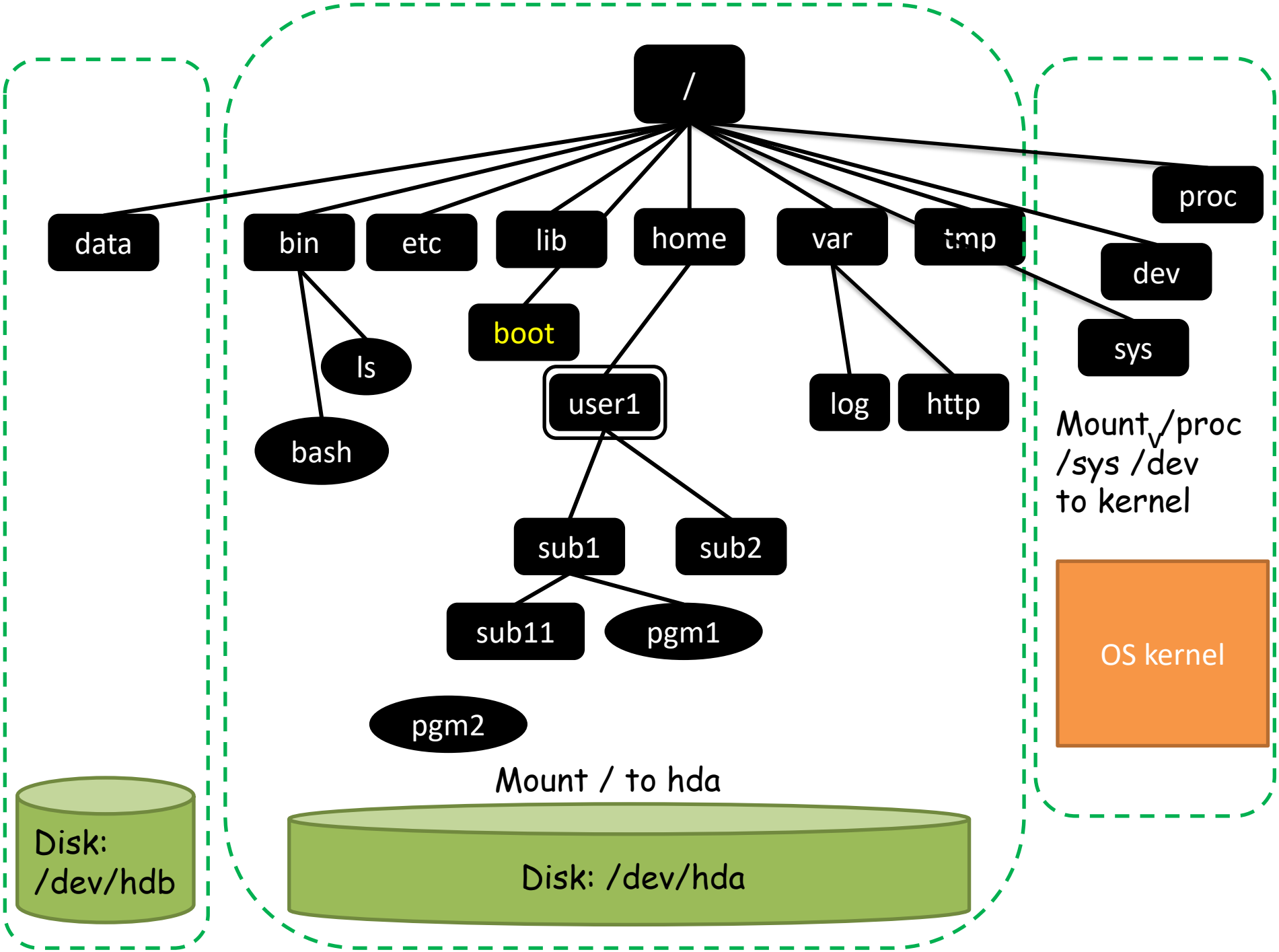
- user1 สร้างไดเรกทอรี่ย่อย sub1 sub2 sub1/sub11 sub1/sub11/pgm2 sub1/pgm1
- sys admin ติดตั้ง apache server จึงมี /var/http


/
bin
etc
lib
home
var
tmp
proc
dev
sys
boot
ls
bash
user1
log
http
sub1
sub2
sub11
pgm1
pgm2
Mount /proc
/sys /dev
to kernel
OS kernel
Mount / to hda
Disk:
/dev/hdb
Disk: /dev/hda

ความสัมพันธ์ระหว่างระบบไฟล์กับอุปกรณ์เก็บข้อมูล 4

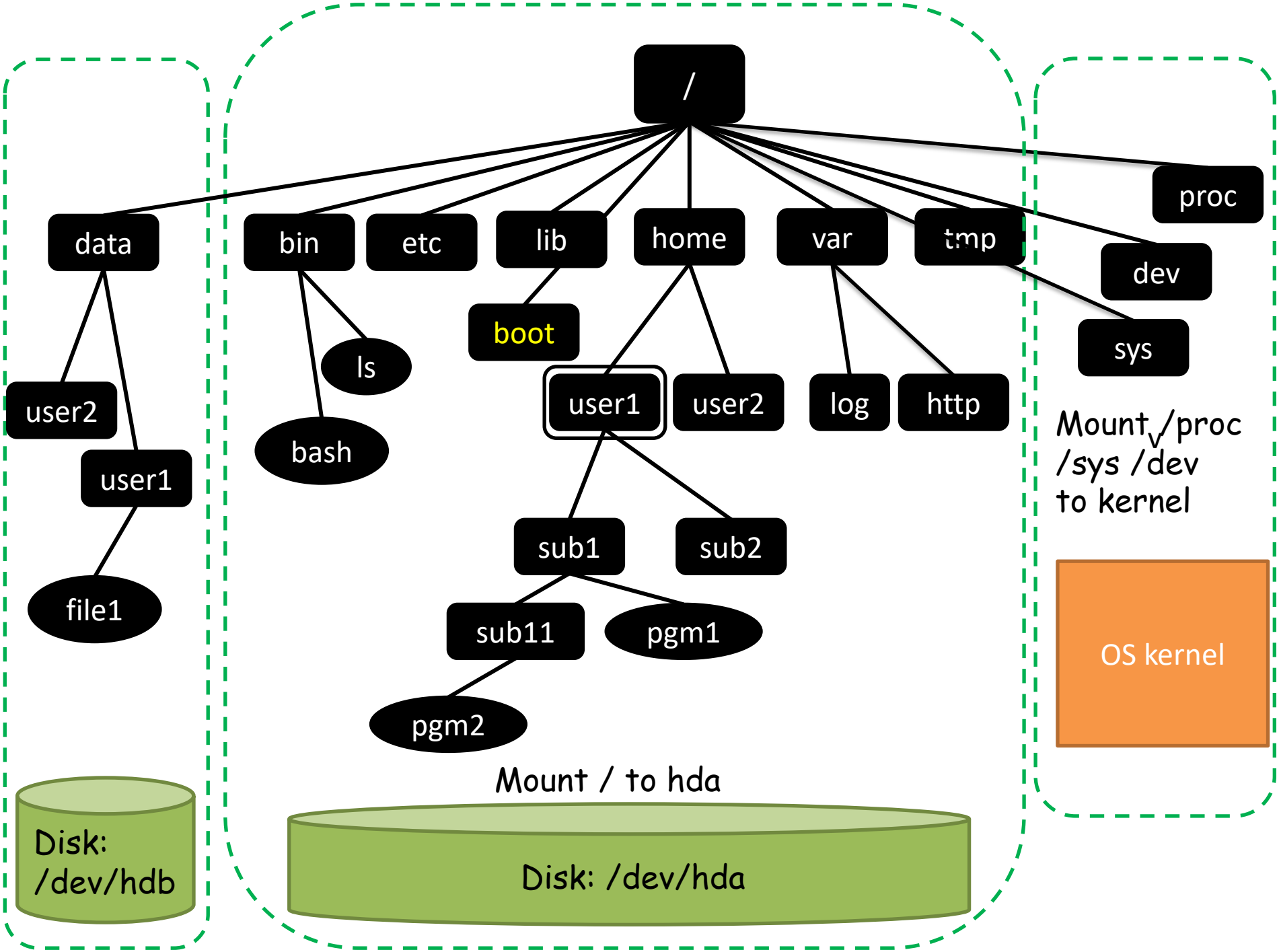
- sys admin ต้องการเพิ่ม disk hdb สำหรับเก็บข้อมูล โดยจะให้เข้าถึงได้ทาง /data ไดเรกทอรี่
- เพิ่ม Mount Point 5
  - Mount point 4 = (**"/data", hdb**, ข้อมูลอื่นๆ)
- <u>ถามว่า อยากให้ /home/user1 เป็น mount point ได้ไหม?</u> **(Advance)**
  - ได้ Admin สามารถกำหนดให้ไดเรกทอรี่ทุกอันเป็น mount point ได้
  - ถ้า mount /home/user1 บน hbd ตราบใดที่มันยัง mount อยู่ การเข้าถึง /home/user1 ก็จะเหมือนเข้าใช้ disk ใหม่ (มันจะบังพื้นที่เดิม)
  - เมื่อเลิก mount ก็จะเห็นข้อมูลเดิมใน /home/user1


/
data
proc
dev
sys
Mount Point 1: (/, hda,...)
Mount Point 2: (/proc, kernel,...)
Mount Point 3: (/dev, kernel,...)
Mount Point 4: (/sys, kernel,...)
Mount Point 5: (/data, hdb,...)
Mount /proc
/sys /dev
to kernel
OS kernel
Mount / to had
List ดู mount point ใช้คำสั่ง
$ mount
Disk:
/dev/hdb
Disk: /dev/hda


/
data
bin
etc
lib
home
var
tmp
proc
dev
sys
boot
ls
bash
user1
log
http
sub1
sub2
sub11
pgm1
pgm2
Mount /proc
/sys /dev
to kernel
OS kernel
Mount / to hda
Disk:
/dev/hdb
Disk: /dev/hda

ความสัมพันธ์ระหว่างระบบไฟล์กับอุปกรณ์เก็บข้อมูล 4

- sys admin ต้องการเพิ่ม disk hdb สำหรับเก็บข้อมูล โดยจะให้เข้าถึงได้ทาง /data ไดเรกทอรี่
- เพิ่ม Mount Point 5
  - Mount point 4 = (”/data”, hdb, ข้อมูลอื่นๆ)
- ผู้ใช้เพิ่มไดเรกทอรี่ /data/user1 /data/user2 /data/user1/file1


/
data
bin
etc
lib
home
var
tmp
proc
dev
sys
boot
ls
bash
user2
user1
file1
user1
user2
log
http
sub1
sub2
sub11
pgm1
pgm2
Mount / to hda
Disk: /dev/hdb
Disk: /dev/hda
Mount /proc /sys /dev to kernel
OS kernel

ชื่อไดเรกทอรี่และไฟล์

เอ็นติตี้	Absolute	Relative
user1	/home/user1	. (current dir)
home	/home	.. (parent dir)
/	/	../..
pgm2	/home/user1/sub1/sub11/pgm2	sub1/sub11/pgm2
log	/var/log	../../var/log
bmon		
http		

การใช้งานไดเรกทอรี่

- ในระบบไฟล์ของลินุกซ์ไดเรกทอรี่จะถูกใช้เก็บข้อมูลตามประเพณีปฏิบัติดังนี้
  - /bin เก็บ binary executable files คือไฟล์ไบนารีโปรแกรมต่างๆ
  - /etc เก็บ configuration files เช่น network configuration
  - */dev เก็บไฟล์ที่เป็นตัวแทนอ้างอิงถึงอุปกรณ์ในระบบคอมพิวเตอร์ทั้งหมดที่ลินุกซ์มองเห็น (pseudo file system)*
  - /home เก็บไดเรกทอรี่ของผู้ใช้
  - /lib เก็บไลบารี่สำหรับพัฒนาโปรแกรมหรือให้แอพพลิชั่นโหลดใช้งาน
  - */proc เก็บข้อมูลของโปรเซสที่รันในระบบทั้งหมด (pseudo file system)*
  - /var เก็บข้อมูลสนับสนุนการประมวลผลโปรแกรม /tmp เก็บข้อมูลชั่วคราว
- บน windows เราสามารถเปลี่ยน config ทาง regedit แต่บน linux ทำผ่าน pseudo file system

โฮมไดเรกทอรี่

- ในภาพ เมื่อผู้ใช้ “user1” ล็อกอิน ระบบจะกำหนดให้ผู้ใช้เริ่มทำงานที่โฮม (home) ไดเรกทอรี่ที่ระบบกำหนดให้สำหรับผู้ใช้แต่ละคน ซึ่งโดยทั่วไปลินุกซ์จะกำหนดให้เป็นไดเรกทอรี่ /home/user1
- ผู้ใช้สามารถสร้างไฟล์และไดเรกทอรี่ย่อยได้ ในภาพ สมมุติว่าผู้ใช้ “user1” ได้สร้างไดเรกทอรี่ย่อย sub1 และ sub2 และสร้างไดเรกทอรี่ย่อย sub11 ภายใต้ sub1 และสร้างไฟล์ pgm1 pgm2 และ link1

คำสั่ง cd และ pwd

- คำสั่ง cd คือ change directory เป็นคำสั่งที่ถูก build-in ในโปรแกรมเชล (shell) และทำหน้าที่เปลี่ยนตำแหน่งไดเรกทอรี่ปัจจุบันในการปฏิบัติงานของผู้ใช้ (current directory)
- cd <pathname> คือการเปลี่ยน ไดเรกทอรี่ไปยัง pathname นั้น สามารถเป็นได้ทั้ง absolute และ relative pathname
  - ถ้าใช้ cd แล้วกด enter เชลจะเปลี่ยนไดเรกทอรี่ปัจจุบันเป็นโฮมไดเรกทอรี่
- pwd (print working directory) เป็นคำสั่งสำหรับแสดง absolute pathname ของไดเรกทอรี่ปัจจุบันบนหน้าจอ

การใช้งานอูบุนตูลินุกซ์

- ขอให้ล็อกอินเข้าใช้งานด้วยบัญชีผู้ใช้หลักของเครื่อง “user1”
- เมื่อล็อกอินแล้วระบบจะเรียกคอมมานด์ไลน์เชลโปรแกรมขึ้นมาทำงาน
- โปรแกรมเชลที่ใช้คือ /bin/bash และจะรอรับคำสั่งจากผู้ใช้
- เมื่อเริ่มทำงาน Bash จะอ่านข้อกำหนดตัวแปรสภาพแวดล้อมจากไฟล์ /home/user1/.bashrc
- ชื่อไฟล์หรือไดเรกทอรี่ที่นำหน้าด้วย ”.” เรียกว่า hidden file/dir เพราะถ้าใช้คำสัง ls ธรรมดาจะไม่ถูก list ต้องใช้คำสั่ง “ls –la” จึงจะแสดงรายการ
- เมื่อล็อกอินตัวแปร HOME จะถูกกำหนดค่าให้เป็น “/home/user1” และอ้างอิงถึงได้ด้วย $HOME หรือ ${HOME} เช่นคำสั่ง “cd $HOME” จะหมายถึง ”cd /home/user1”

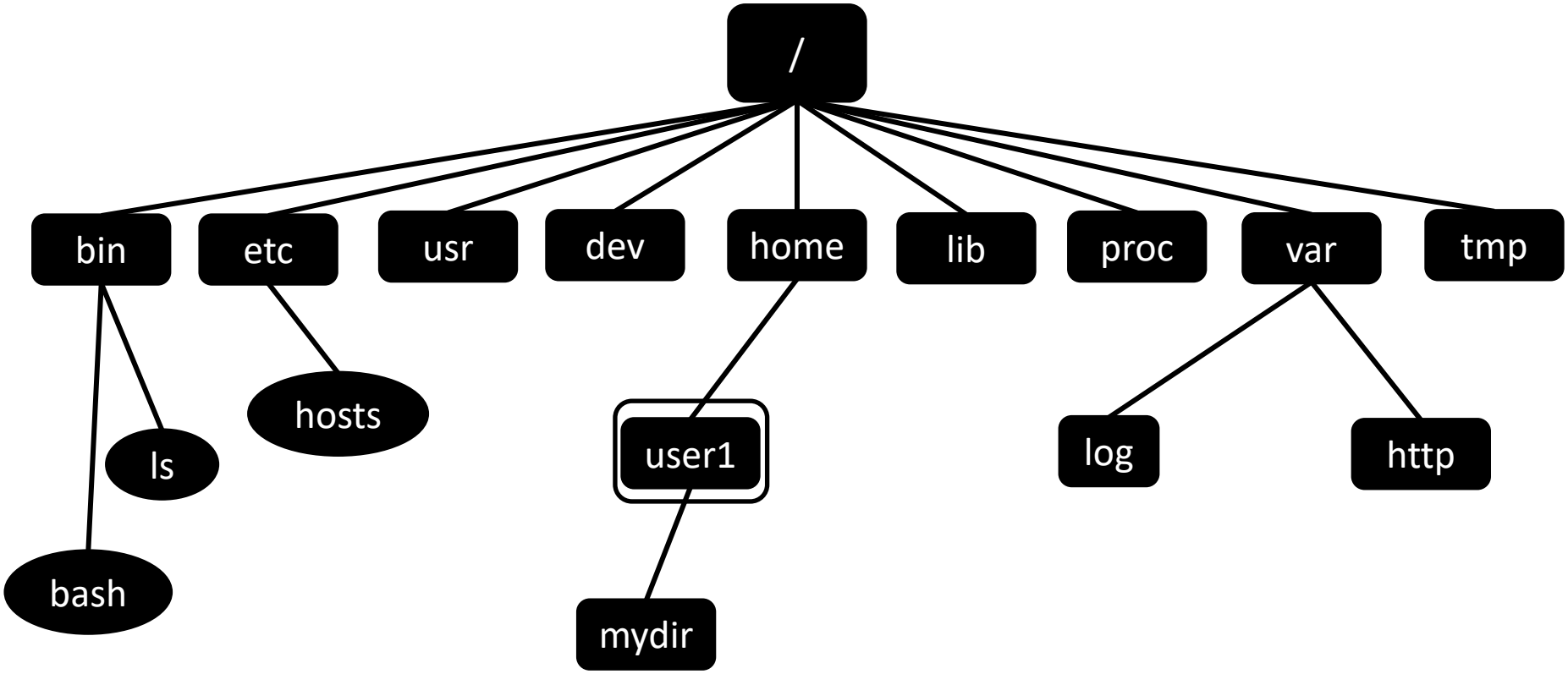
การใช้งาน Linux เบื้องต้น

- คำสั่ง cd และ pwd
- การสร้างไดเรกทอรี่และไฟล์
- การ edit และจัดการไฟล์
- ทำความรู้จักกับ permission ของไฟล์
- การสร้าง shell script
- การรัน background process
- การควบคุมและ kill process และ signal เบื้องต้น
- การสร้างและใช้งาน screen session
- การ compile program
- การใช้ make utility เพื่อพัฒนาโปรแกรม

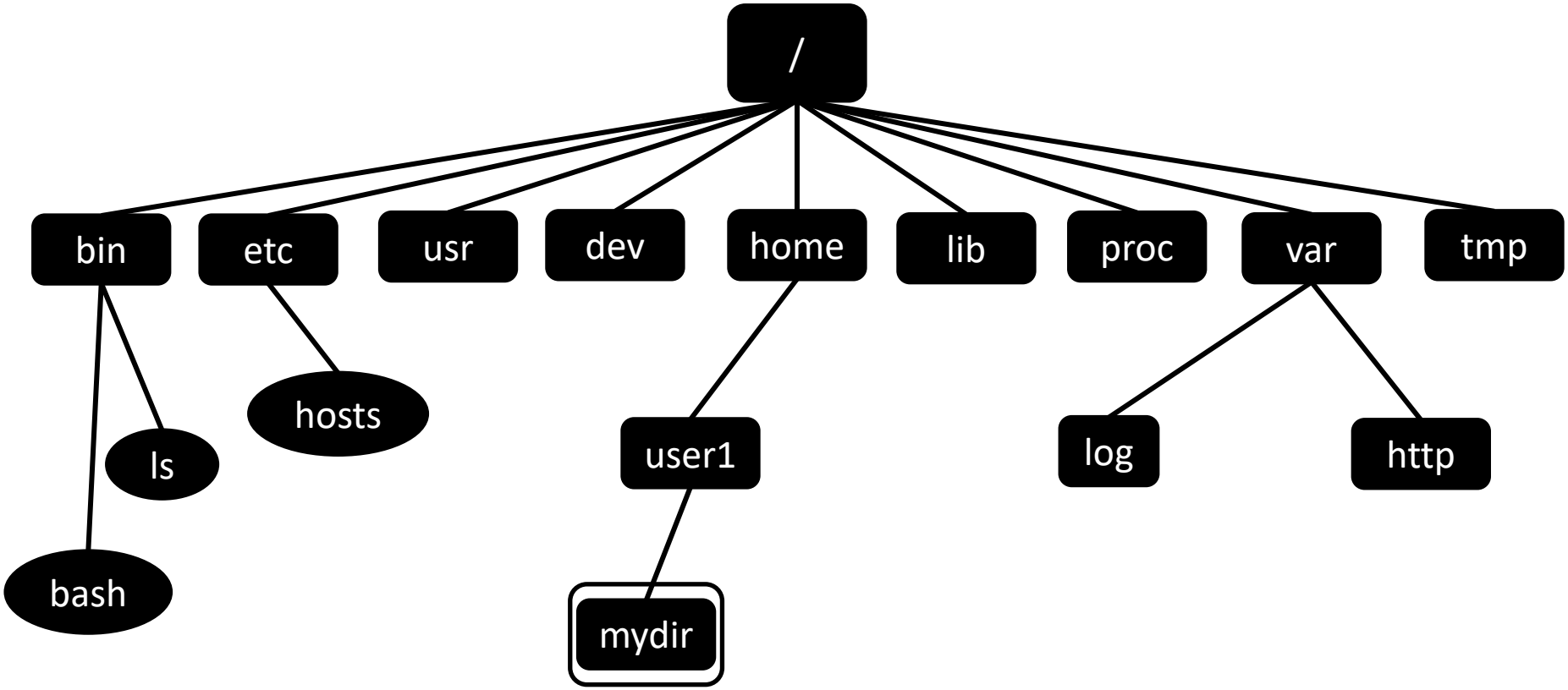
การสร้าง directory และไฟล์

- mkdir <ชื่อไดเรกทอรี่> ...
  - ... หมายถึงอาจตามด้วยชื่อหลายชื่อก็ได้
- touch <ชื่อไฟล์> ...
- cp <ไฟล์ต้นฉบับ> <ไฟล์ก๊อปปี้>
- mv <ชื่อไฟล์เดิม>. <ชื่อไฟล์ใหม่>

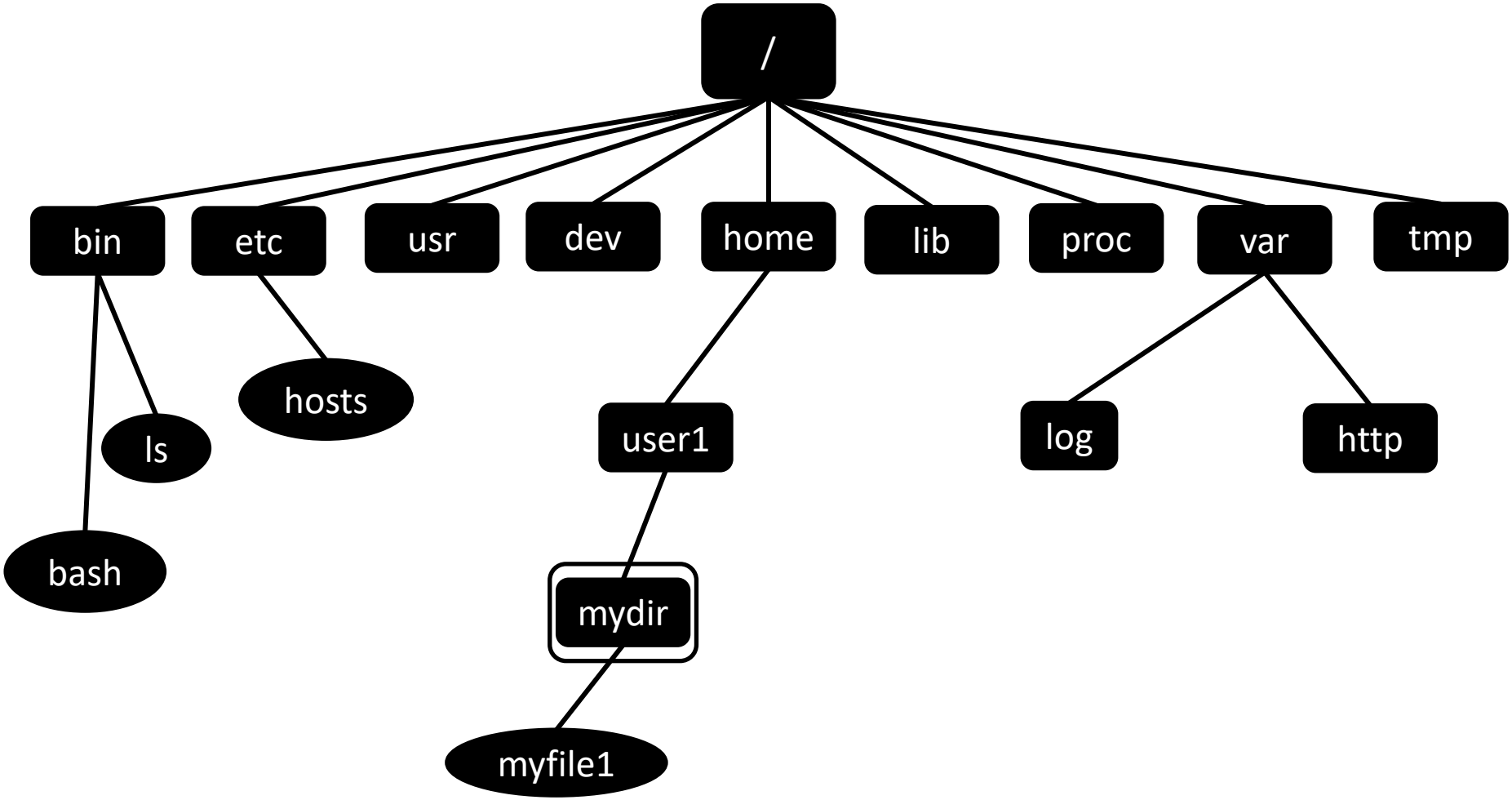
```
$ mkdir mydir
$ cd mydir
$ touch myfile1
$ touch ../myfile2
$ cp /etc/hosts myfile3
$ mv ../myfile2 /tmp/myfile2
```


/
bin
etc
usr
dev
home
lib
proc
var
tmp
hosts
ls
bash
user1
mydir
log
http

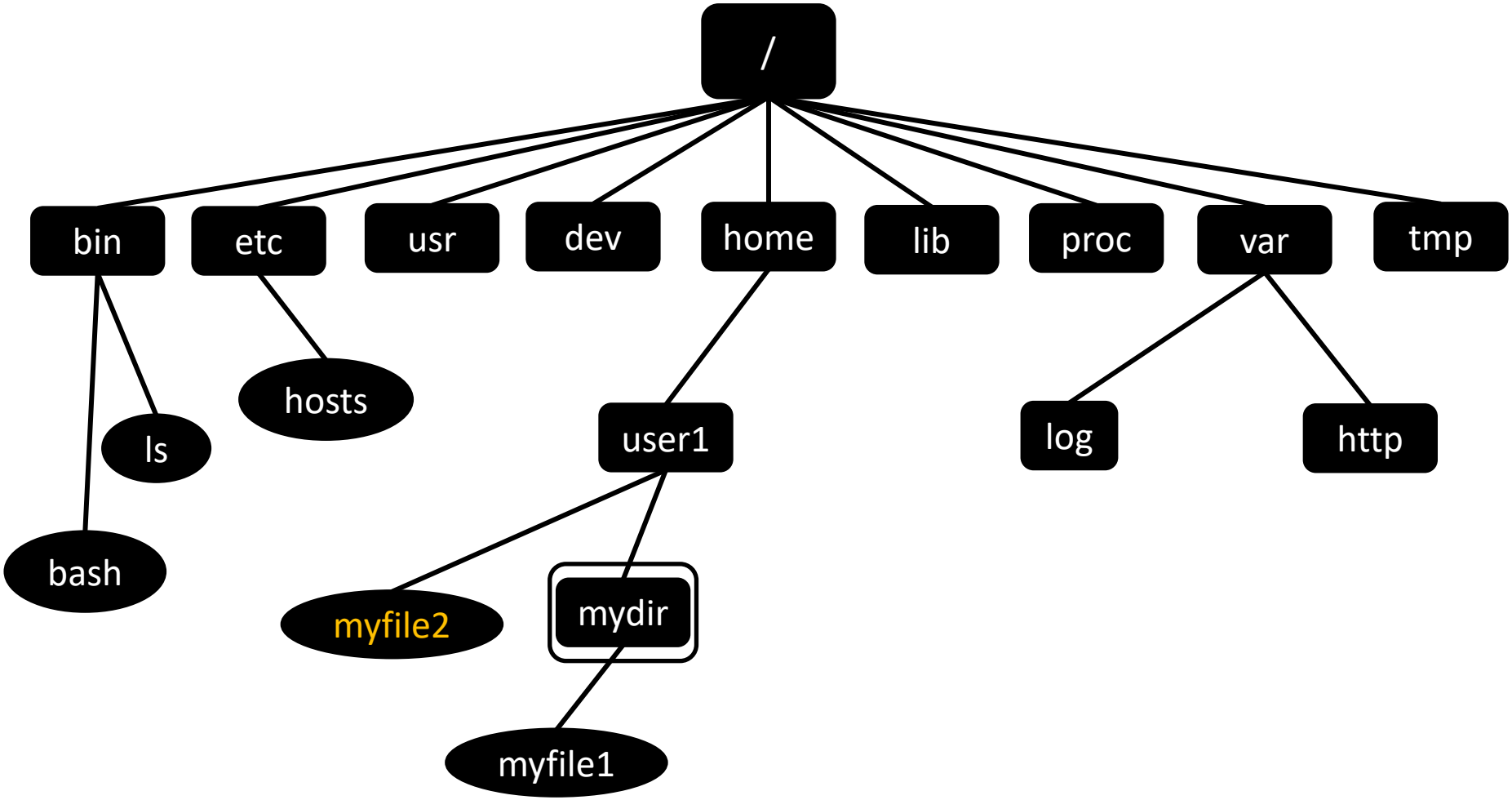
```
$ mkdir mydir
```


/
bin
etc
usr
dev
home
lib
proc
var
tmp
hosts
ls
bash
user1
mydir
log
http

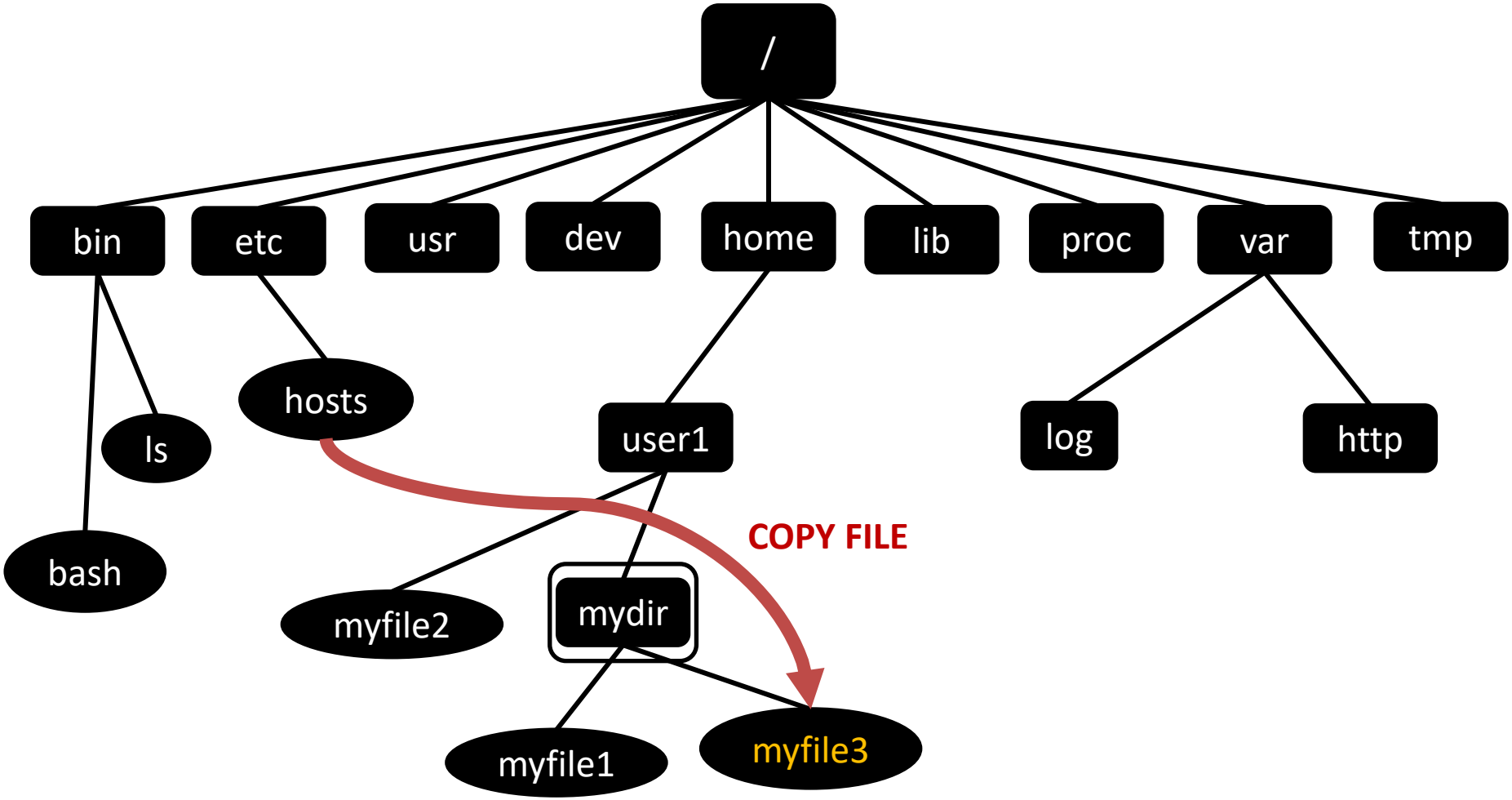
```
$ cd mydir
$
```



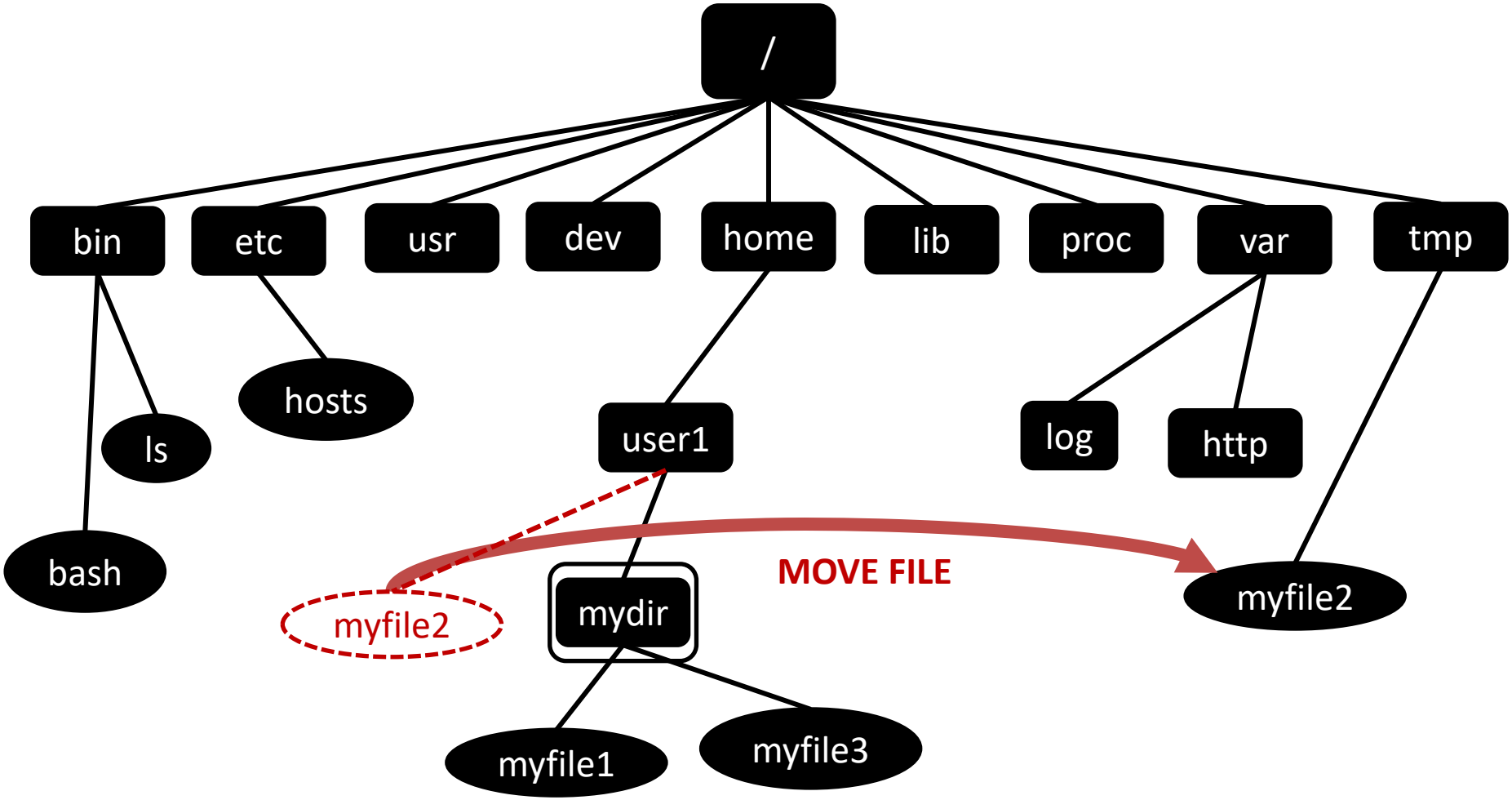

```
$ touch myfile1
$
```



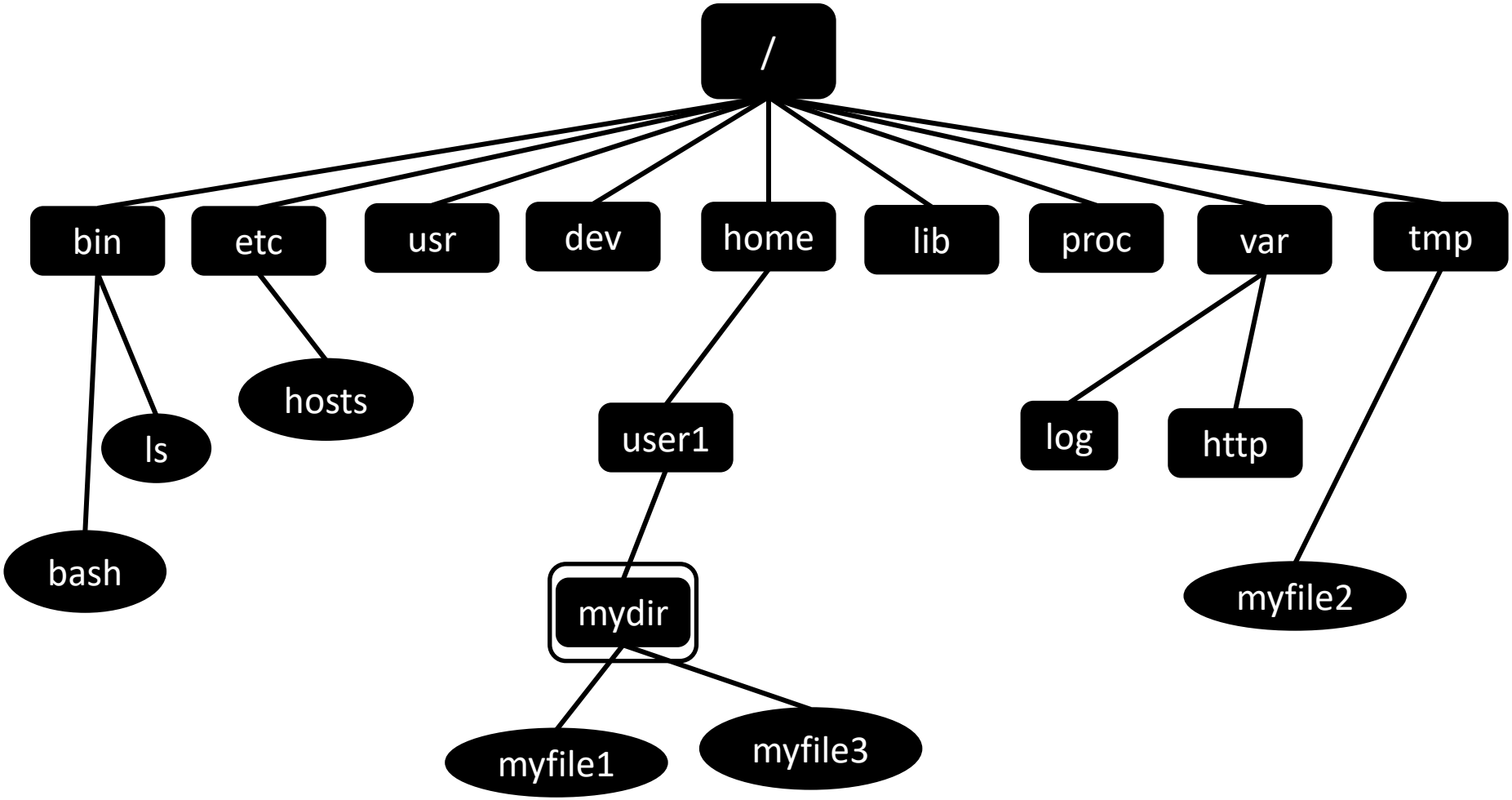

```
$ touch ../myfile2
$
```




```
$ cp /etc/hosts myfile3
$
```



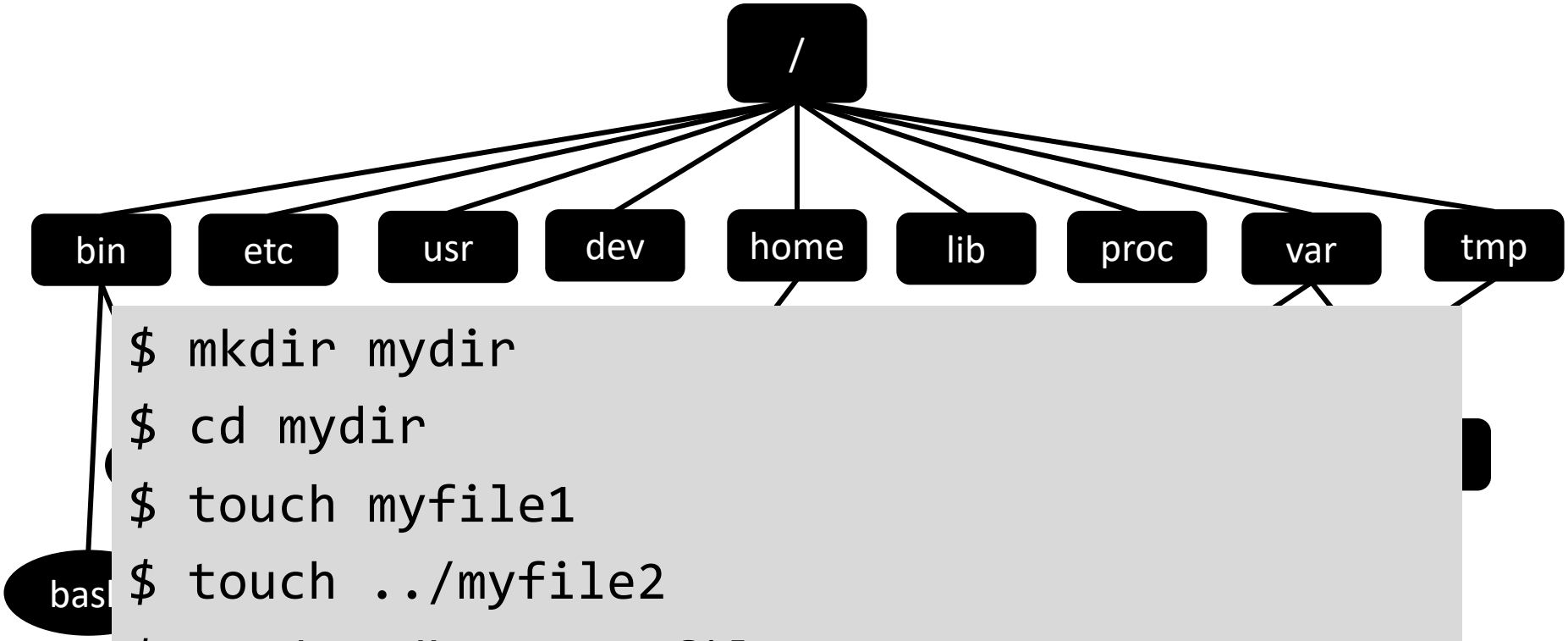

```
$ mv ../myfile2 /tmp/myfile2
$
```




```
$ mv ../myfile2 /tmp/myfile2
$
```

ทดลองทำ

- ให้ login เข้าสู่เครื่อง Ubuntu linux
- Username: Room107
- Password: cstu107
- ผู้ใช้จะมี account ของตนและจะเริ่มทำงานใน home directory
  - สมมุติว่า home ของ นศ คือ Room107
  - ให้สมมุติว่าไดเรอทอรี่ user1 ใน Slide คือ Room107 ของเครื่องของ นศ
- ปฏิบัติ 5 นาที: ให้ออกคำสั่งตามตัวอย่างที่ผ่านมา

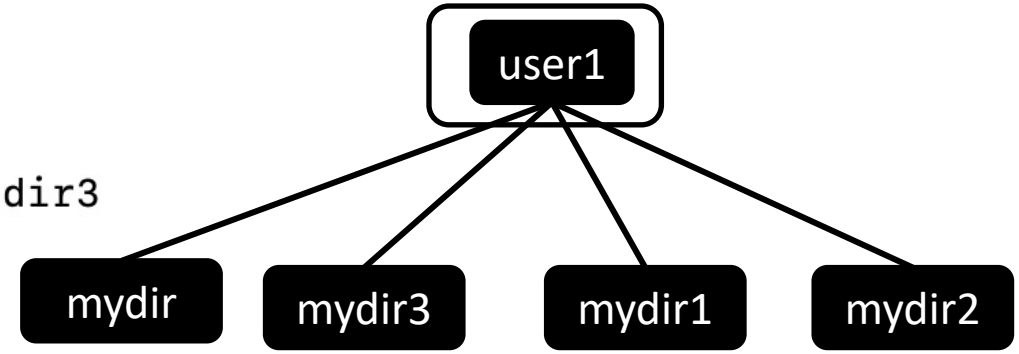



```
$ mkdir mydir
$ cd mydir
$ touch myfile1
$ touch ../myfile2
$ cp /etc/hosts myfile3
$ mv ../myfile2 /tmp/myfile2
$ ls -l      ➔ ทำอะไร
$ cat myfile3 ➔ ทำอะไร
$ ls .. ➔ ทำอะไร
$ ถ้าจะย้าย myfile3 ไปไว้ที่ .. จะต้องออกคำสั่งอะไร
```

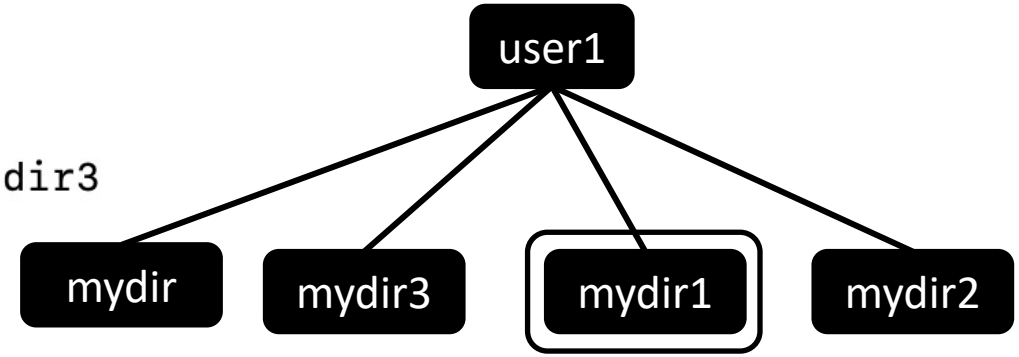
คำสั่ง Linux CLI พื้นฐาน 1

- นศ ทดลองทำ พร้อมกับการสอนของอาจารย์
- หลัง login linux จะเรียกโปรแกรม shell ขึ้นมารอรับคำสั่งจากผู้ใช้
- ผู้ใช้จะมี account ของตนและจะเริ่มทำงานใน home directory
  - สมมุติว่า home ของ นศ คือ Room107
- คำสั่ง pwd ดูตำแหน่ง directory ที่ใช้งานปัจจุบัน
- คำสั่ง cd เปลี่ยนตำแหน่ง directory
- คำสั่ง mkdir สร้าง directory ย่อย
- คำสั่ง touch สร้าง ไฟล์เปล่า
- คำสั่ง cp ทำสำเนาไฟล์
- คำสั่ง mv เปลี่ยนชื่อไฟล์หรือย้ายไฟล์
- คำสั่ง ls แสดงรายการชื่อไฟล์และชื่อ directory ที่เก็บอยู่ใน directory
- คำสั่ง rmdir ลบไดเรกทอรี่ และคำสั่ง rm ลบไฟล์

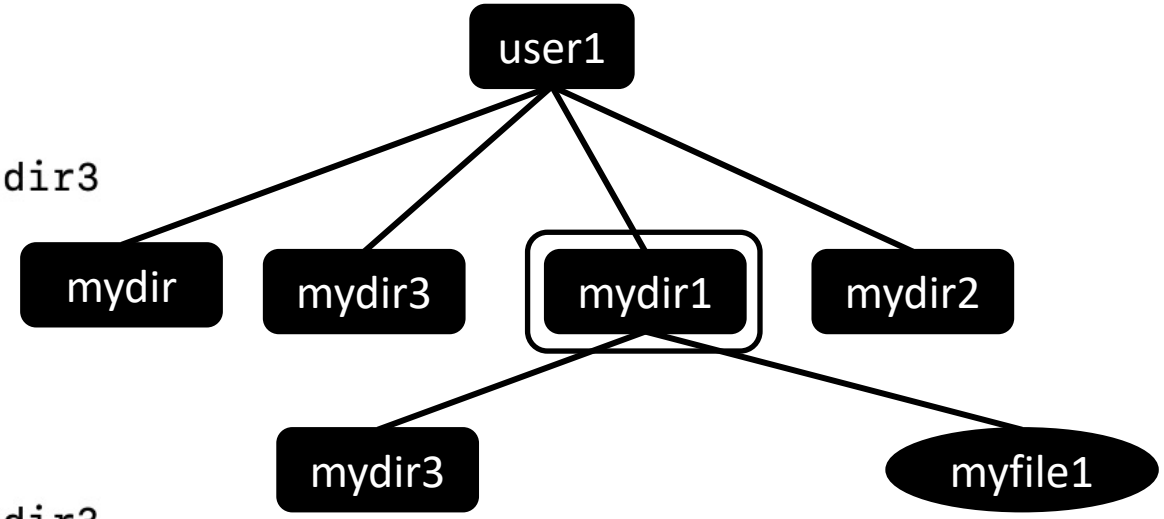
```
user1@vm2:~$ mkdir mydir1
user1@vm2:~$ mkdir mydir2 mydir3
user1@vm2:~$
user1@vm2:~$ pwd
/home/user1
user1@vm2:~$
```



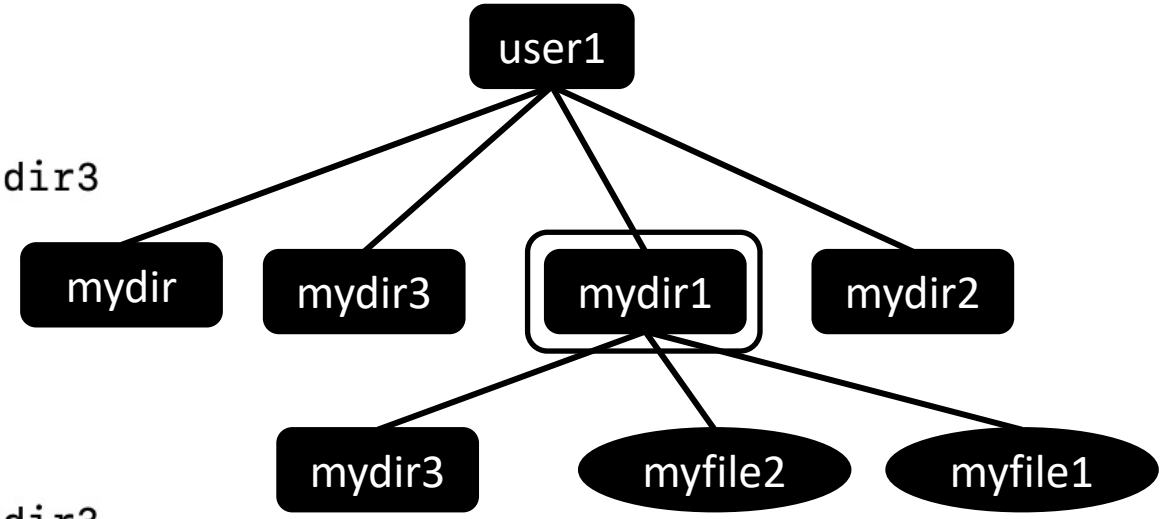
```
user1@vm2:~$ mkdir mydir1
user1@vm2:~$ mkdir mydir2 mydir3
user1@vm2:~$
user1@vm2:~$ pwd
 /home/user1
user1@vm2:~$
user1@vm2:~$ cd mydir1
user1@vm2:~/mydir1$
```



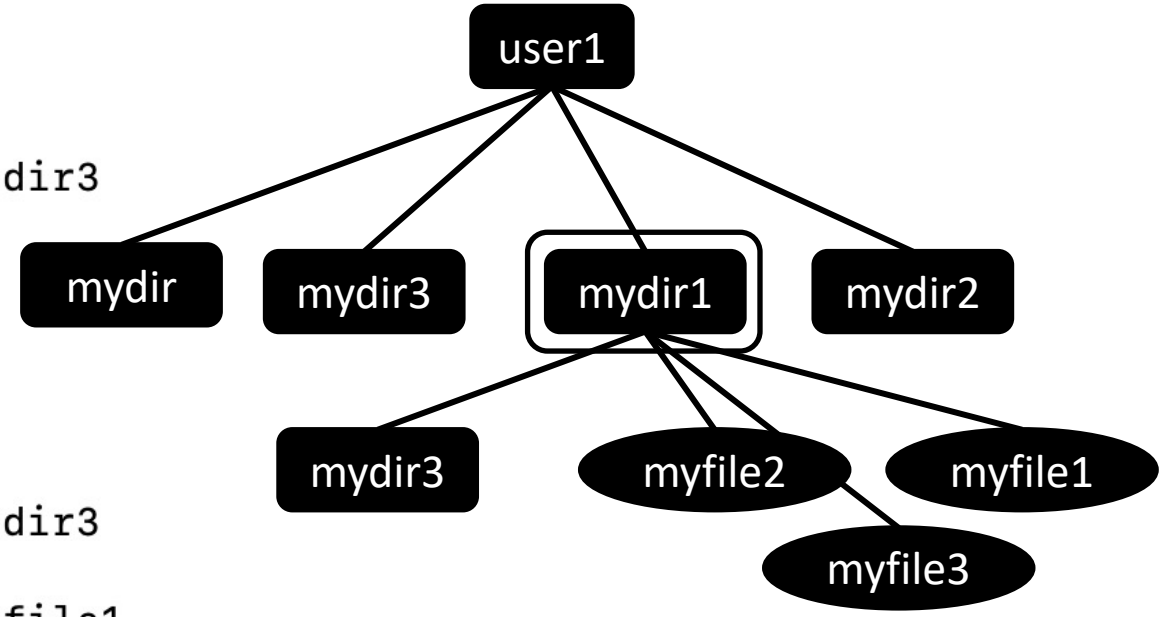
```
user1@vm2:~$ mkdir mydir1
user1@vm2:~$ mkdir mydir2 mydir3
user1@vm2:~$
user1@vm2:~$ pwd
/home/user1
user1@vm2:~$
user1@vm2:~$ cd mydir1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mkdir mydir3
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ touch myfile1
user1@vm2:~/mydir1$ ls -l
total 4
drwxrwxr-x 2 user1 user1 4096 Aug 24 10:14 mydir3
-rw-rw-r-- 1 user1 user1    0 Aug 24 10:14 myfile1
user1@vm2:~/mydir1$
```



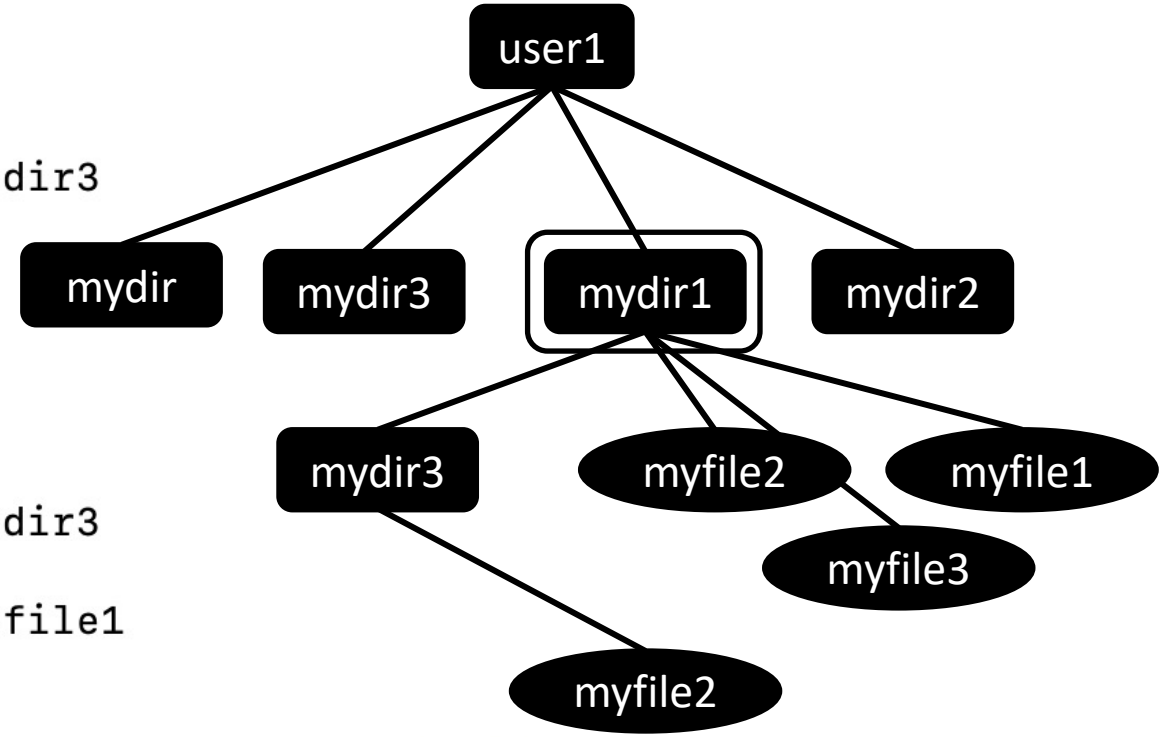
```
user1@vm2:~$ mkdir mydir1
user1@vm2:~$ mkdir mydir2 mydir3
user1@vm2:~$
user1@vm2:~$ pwd
/home/user1
user1@vm2:~$
user1@vm2:~$ cd mydir1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mkdir mydir3
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ touch myfile1
user1@vm2:~/mydir1$ ls -l
total 4
drwxrwxr-x 2 user1 user1 4096 Aug 24 10:14 mydir3
-rw-rw-r-- 1 user1 user1    0 Aug 24 10:14 myfile1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cp /etc/hosts myfile2
user1@vm2:~/mydir1$
```



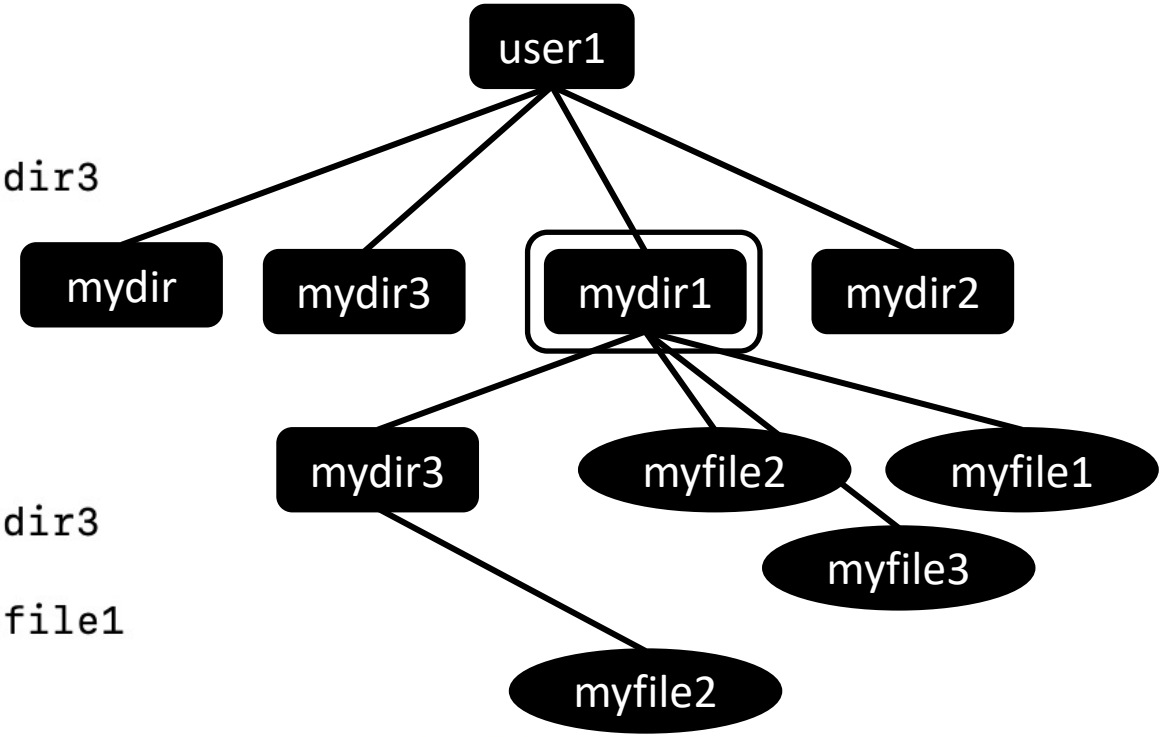
```
user1@vm2:~$ mkdir mydir1
user1@vm2:~$ mkdir mydir2 mydir3
user1@vm2:~$
user1@vm2:~$ pwd
/home/user1
user1@vm2:~$
user1@vm2:~$ cd mydir1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mkdir mydir3
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ touch myfile1
user1@vm2:~/mydir1$ ls -l
total 4
drwxrwxr-x 2 user1 user1 4096 Aug 24 10:14 mydir3
-rw-rw-r-- 1 user1 user1    0 Aug 24 10:14 myfile1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cp /etc/hosts myfile2
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cp myfile2 myfile3
```



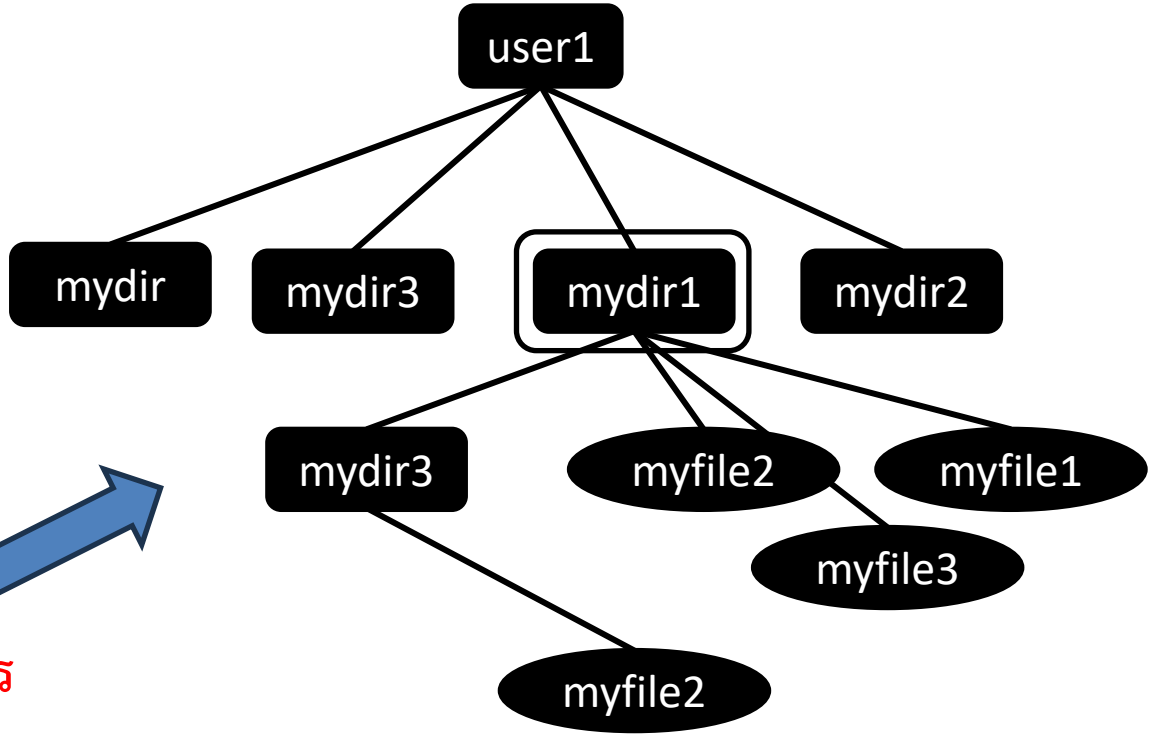
```
user1@vm2:~$ mkdir mydir1
user1@vm2:~$ mkdir mydir2 mydir3
user1@vm2:~$
user1@vm2:~$ pwd
/home/user1
user1@vm2:~$
user1@vm2:~$ cd mydir1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mkdir mydir3
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ touch myfile1
user1@vm2:~/mydir1$ ls -l
total 4
drwxrwxr-x 2 user1 user1 4096 Aug 24 10:14 mydir3
-rw-rw-r-- 1 user1 user1    0 Aug 24 10:14 myfile1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cp /etc/hosts myfile2
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cp myfile2 myfile3
user1@vm2:~/mydir1$ cp myfile2 mydir3
```



```
user1@vm2:~$ mkdir mydir1
user1@vm2:~$ mkdir mydir2 mydir3
user1@vm2:~$
user1@vm2:~$ pwd
/home/user1
user1@vm2:~$
user1@vm2:~$ cd mydir1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mkdir mydir3
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ touch myfile1
user1@vm2:~/mydir1$ ls -l
total 4
drwxrwxr-x 2 user1 user1 4096 Aug 24 10:14 mydir3
-rw-rw-r-- 1 user1 user1    0 Aug 24 10:14 myfile1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cp /etc/hosts myfile2
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cp myfile2 myfile3
user1@vm2:~/mydir1$ cp myfile2 mydir3
user1@vm2:~/mydir1$ ls -l mydir3
total 4
-rw-r--r-- 1 user1 user1 218 Aug 24 10:15 myfile2
user1@vm2:~/mydir1$
```



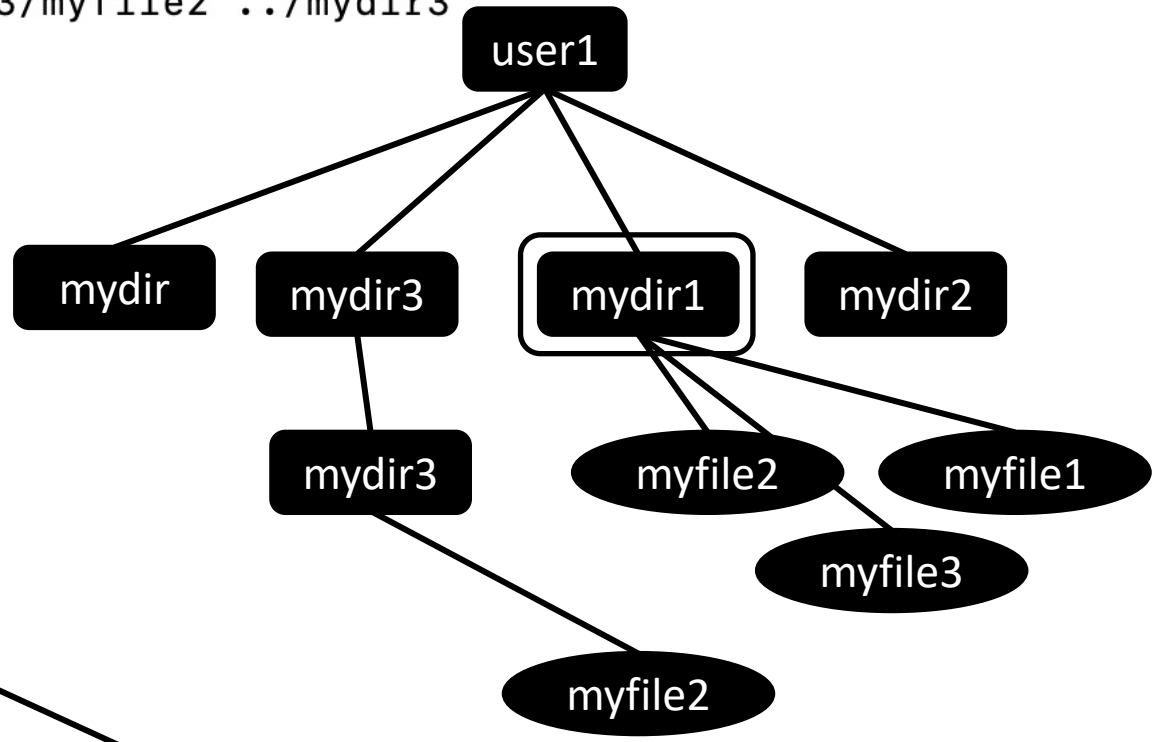
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
```

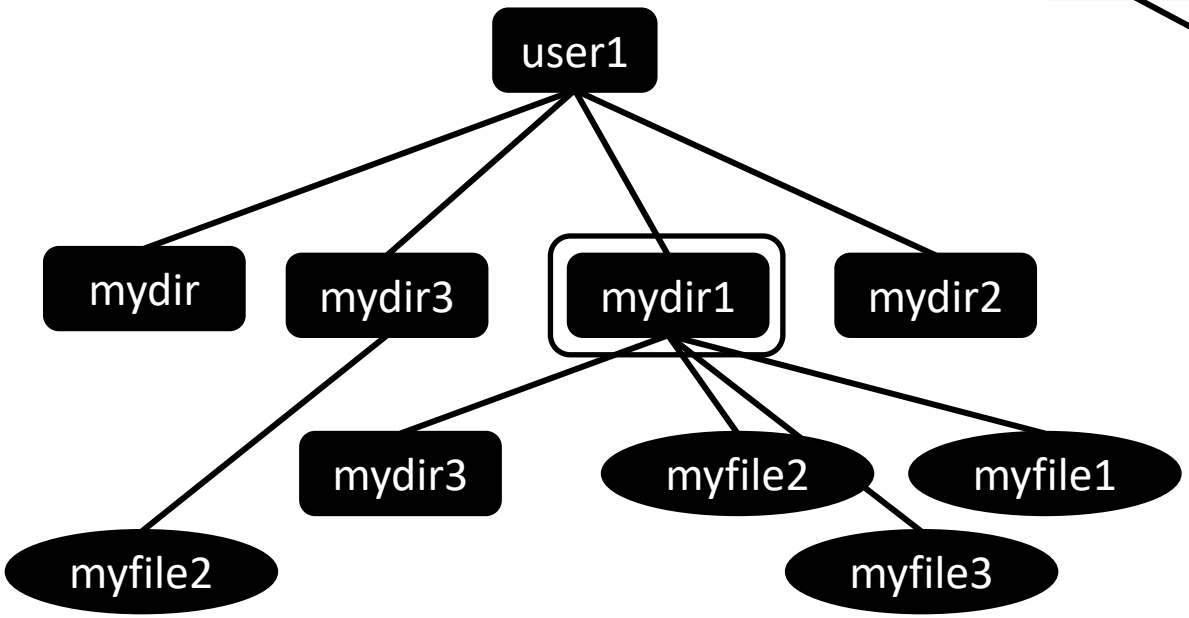


จะเกิดการ
เปลี่ยนแปลงอย่างไร

```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
```

ก. ย้าย mydir3 และสิ่งที่อยู่ใต้มันมาใต้ ../mydir3





ก. ย้าย myfile2 ใน mydir3 มาใต้ ../mydir3 (ถูก)

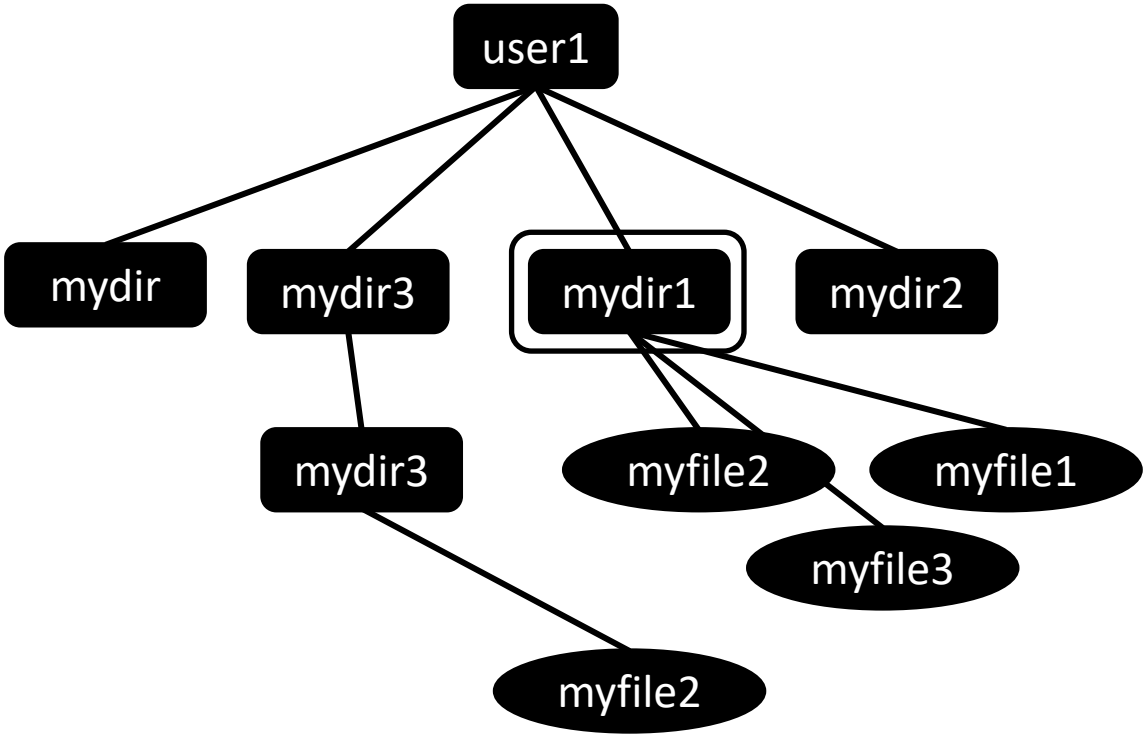
จะให้เป็นข้อ ก ต้องเปลี่ยนคำสั่งจาก

$ mv mydir3/mydir2 ../mydir3

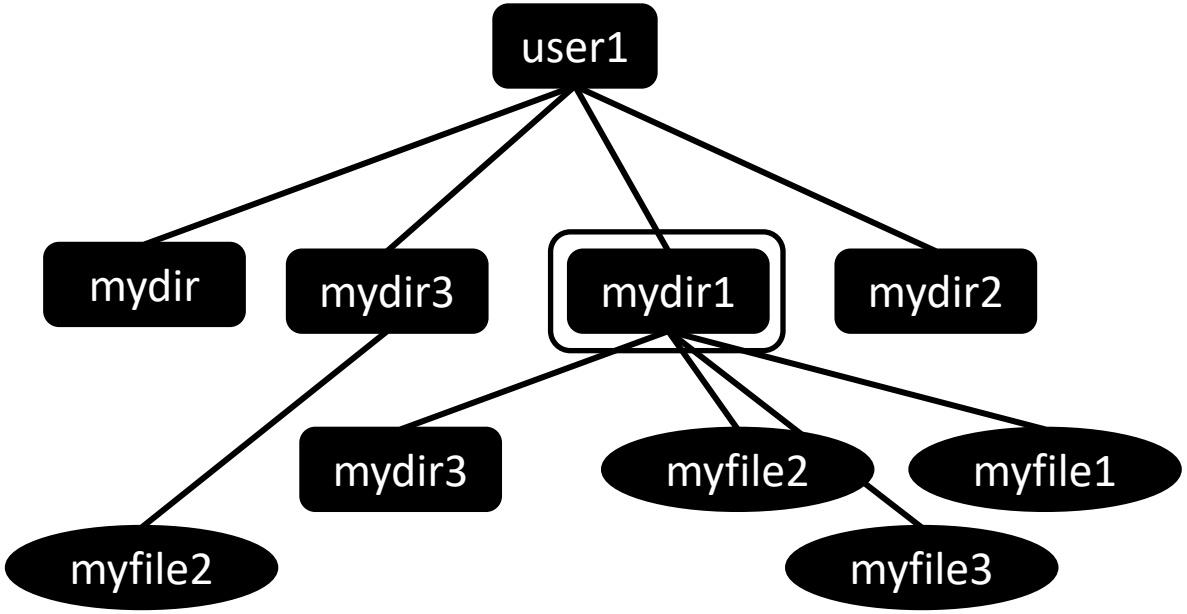
เป็น

$ mv mydir3 ../mydir3

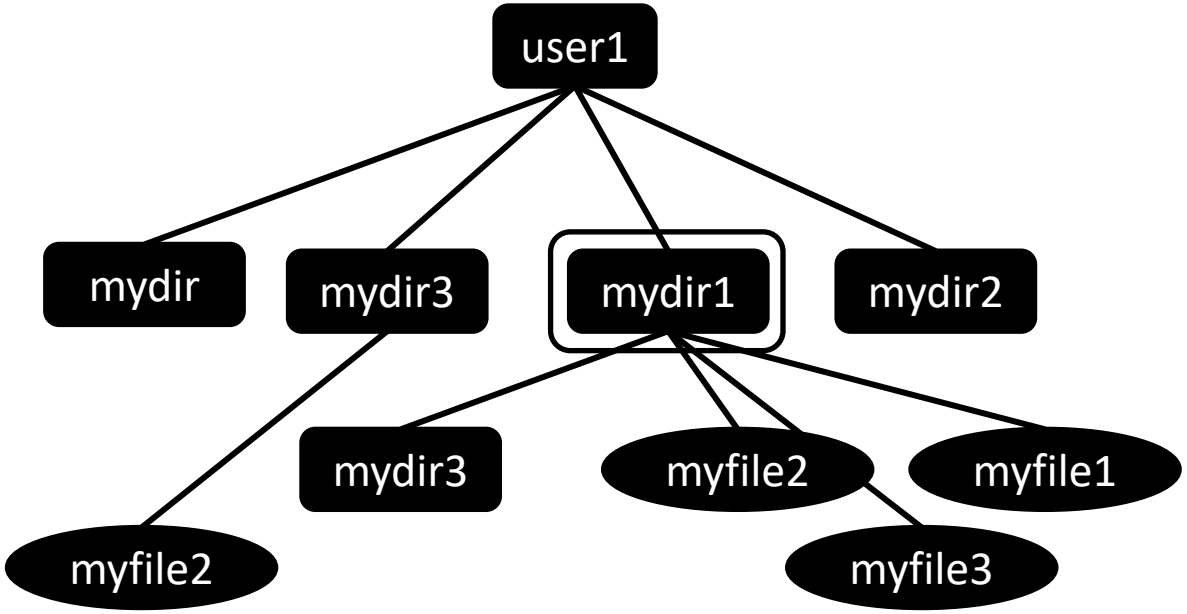
ก. ย้าย mydir3 และสิ่งที่อยู่ใต้มันมาใต้ ../mydir3



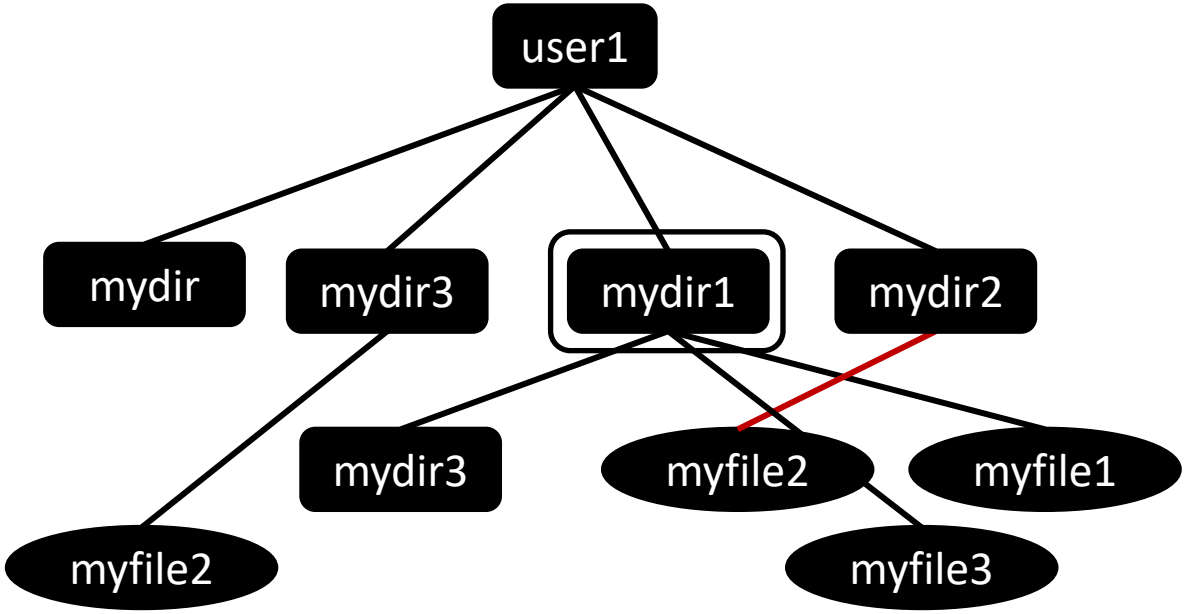
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir3
myfile2
user1@vm2:~/mydir1$ pwd
/home/user1/mydir1
```



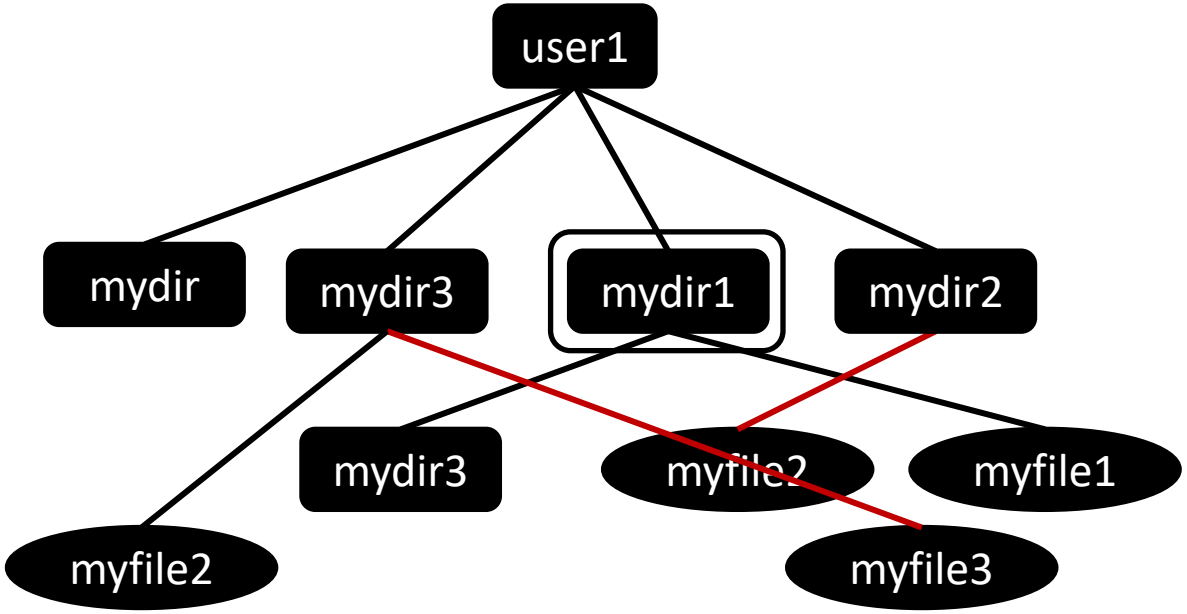
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir3
myfile2
user1@vm2:~/mydir1$ pwd
/home/user1/mydir1
user1@vm2:~/mydir1$ ls
mydir3  myfile1  myfile2  myfile3
```



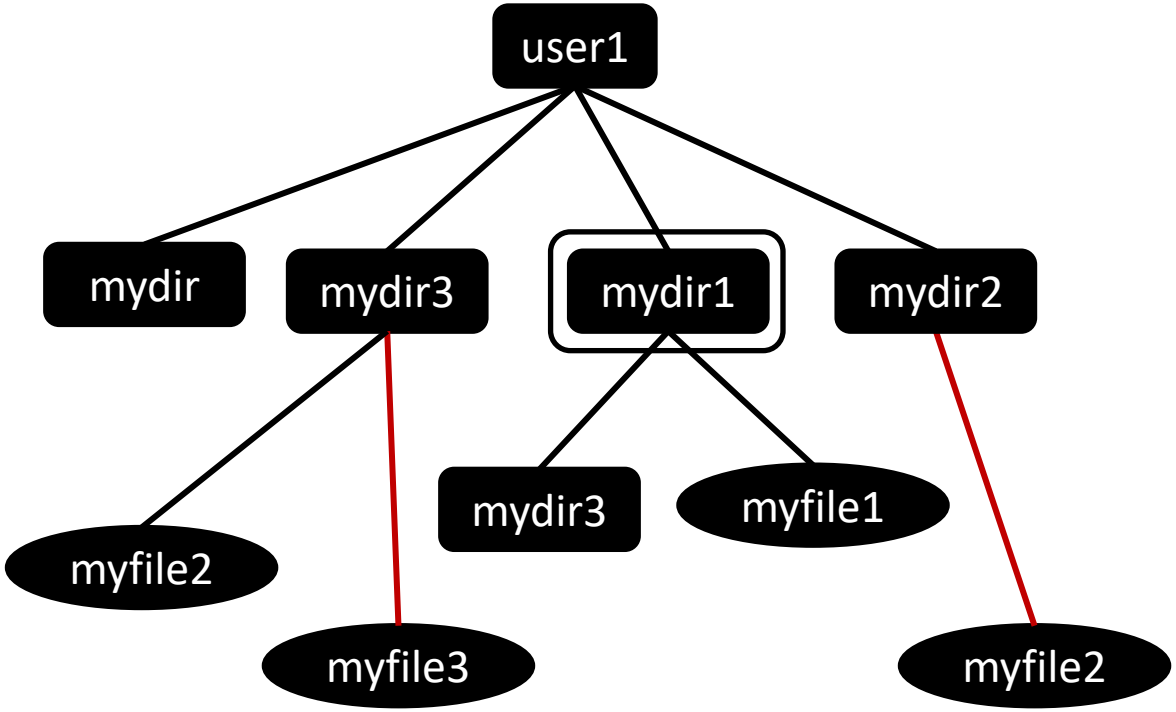
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir3
myfile2
user1@vm2:~/mydir1$ pwd
/home/user1/mydir1
user1@vm2:~/mydir1$ ls
mydir3  myfile1  myfile2  myfile3
user1@vm2:~/mydir1$ mv myfile2 ../mydir2
```



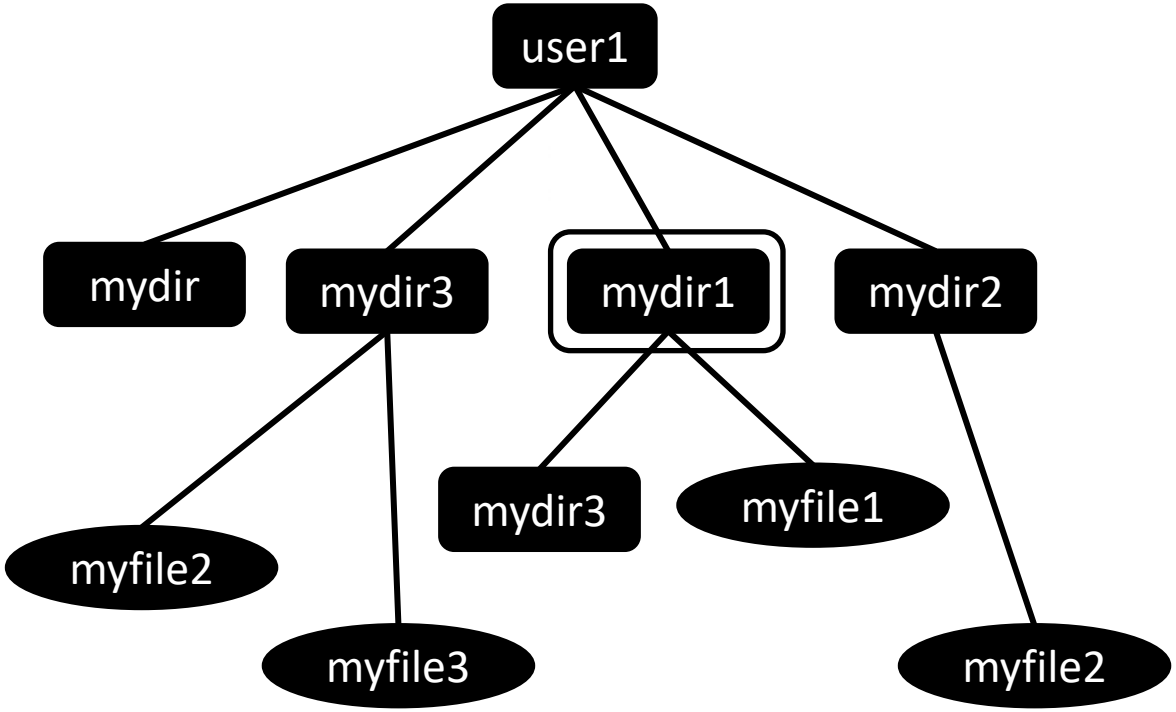
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir3
myfile2
user1@vm2:~/mydir1$ pwd
/home/user1/mydir1
user1@vm2:~/mydir1$ ls
mydir3  myfile1  myfile2  myfile3
user1@vm2:~/mydir1$ mv myfile2 ../mydir2
user1@vm2:~/mydir1$ mv myfile3 ../mydir3
```



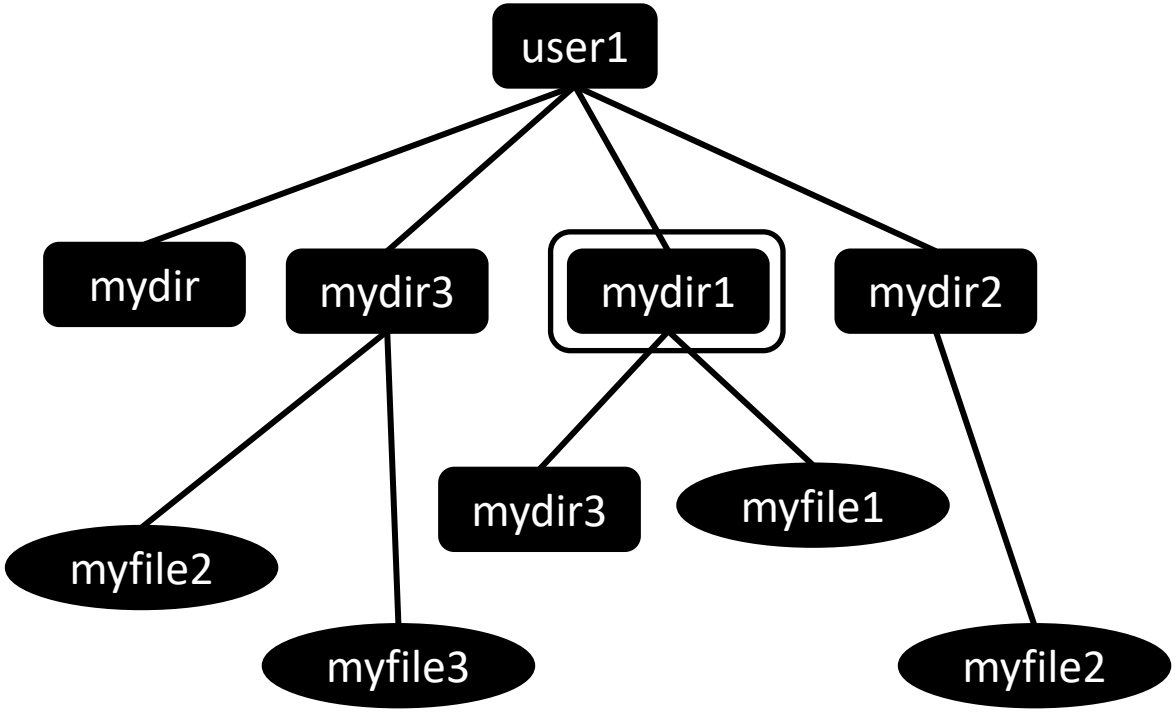
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir3
myfile2
user1@vm2:~/mydir1$ pwd
/home/user1/mydir1
user1@vm2:~/mydir1$ ls
mydir3  myfile1  myfile2  myfile3
user1@vm2:~/mydir1$ mv myfile2 ../mydir2
user1@vm2:~/mydir1$ mv myfile3 ../mydir3
```



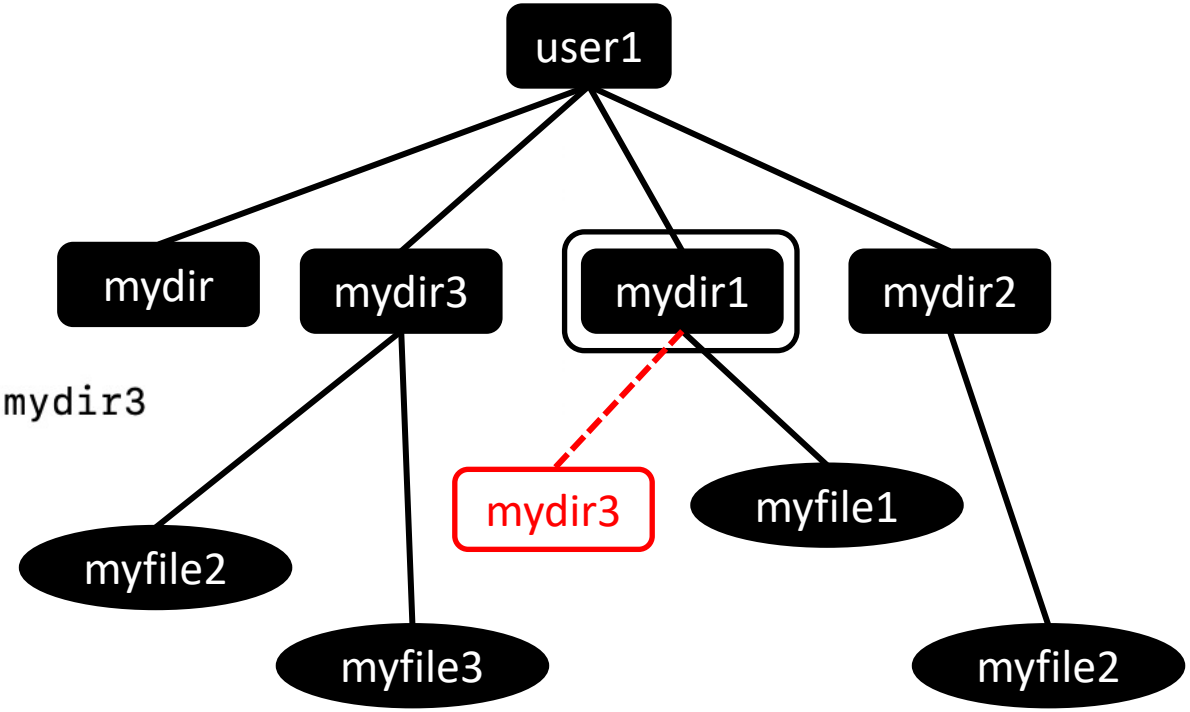
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir3
myfile2
user1@vm2:~/mydir1$ pwd
/home/user1/mydir1
user1@vm2:~/mydir1$ ls
mydir3  myfile1  myfile2  myfile3
user1@vm2:~/mydir1$ mv myfile2 ../mydir2
user1@vm2:~/mydir1$ mv myfile3 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir2 ../mydir3
../mydir2:
myfile2

../mydir3:
myfile2  myfile3
```



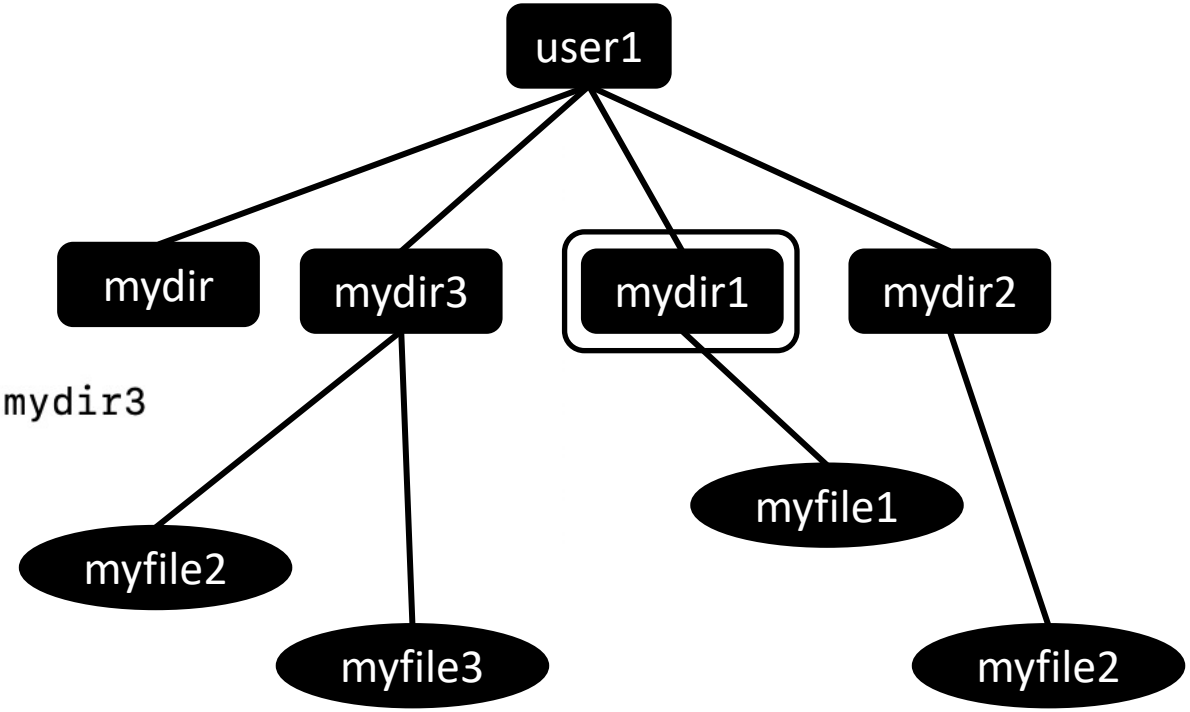
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir3
myfile2
user1@vm2:~/mydir1$ pwd
/home/user1/mydir1
user1@vm2:~/mydir1$ ls
mydir3  myfile1  myfile2  myfile3
user1@vm2:~/mydir1$ mv myfile2 ../mydir2
user1@vm2:~/mydir1$ mv myfile3 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir2 ../mydir3
../mydir2:
myfile2

../mydir3:
myfile2  myfile3
user1@vm2:~/mydir1$ ls
mydir3  myfile1
user1@vm2:~/mydir1$
```



```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir3
myfile2
user1@vm2:~/mydir1$ pwd
/home/user1/mydir1
user1@vm2:~/mydir1$ ls
mydir3  myfile1  myfile2  myfile3
user1@vm2:~/mydir1$ mv myfile2 ../mydir2
user1@vm2:~/mydir1$ mv myfile3 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir2 ../mydir3
../mydir2:
myfile2

../mydir3:
myfile2  myfile3
user1@vm2:~/mydir1$ ls
mydir3  myfile1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ rmdir mydir3
```



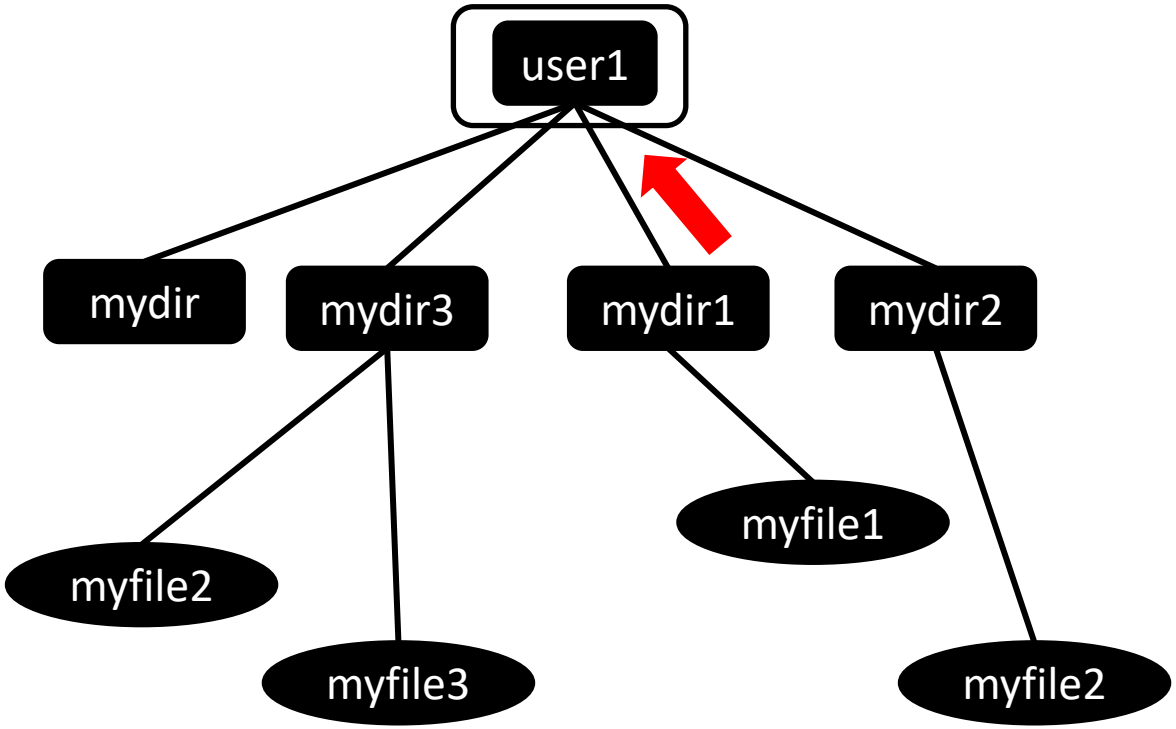
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ mv mydir3/myfile2 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir3
myfile2
user1@vm2:~/mydir1$ pwd
/home/user1/mydir1
user1@vm2:~/mydir1$ ls
mydir3  myfile1  myfile2  myfile3
user1@vm2:~/mydir1$ mv myfile2 ../mydir2
user1@vm2:~/mydir1$ mv myfile3 ../mydir3
user1@vm2:~/mydir1$ ls ../mydir2 ../mydir3
../mydir2:
myfile2

../mydir3:
myfile2  myfile3
user1@vm2:~/mydir1$ ls
mydir3  myfile1
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ rmdir mydir3
user1@vm2:~/mydir1$ ls
myfile1
user1@vm2:~/mydir1$
```



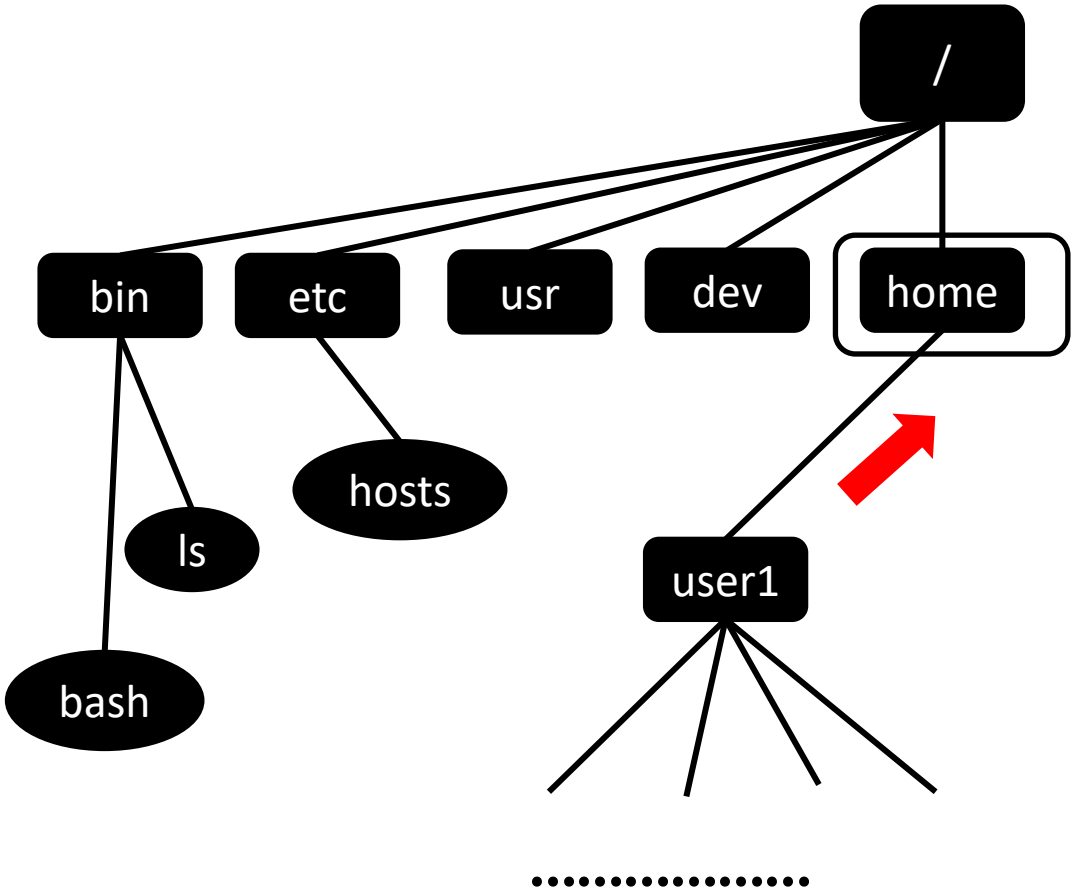
24/1/67

```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cd ..
```

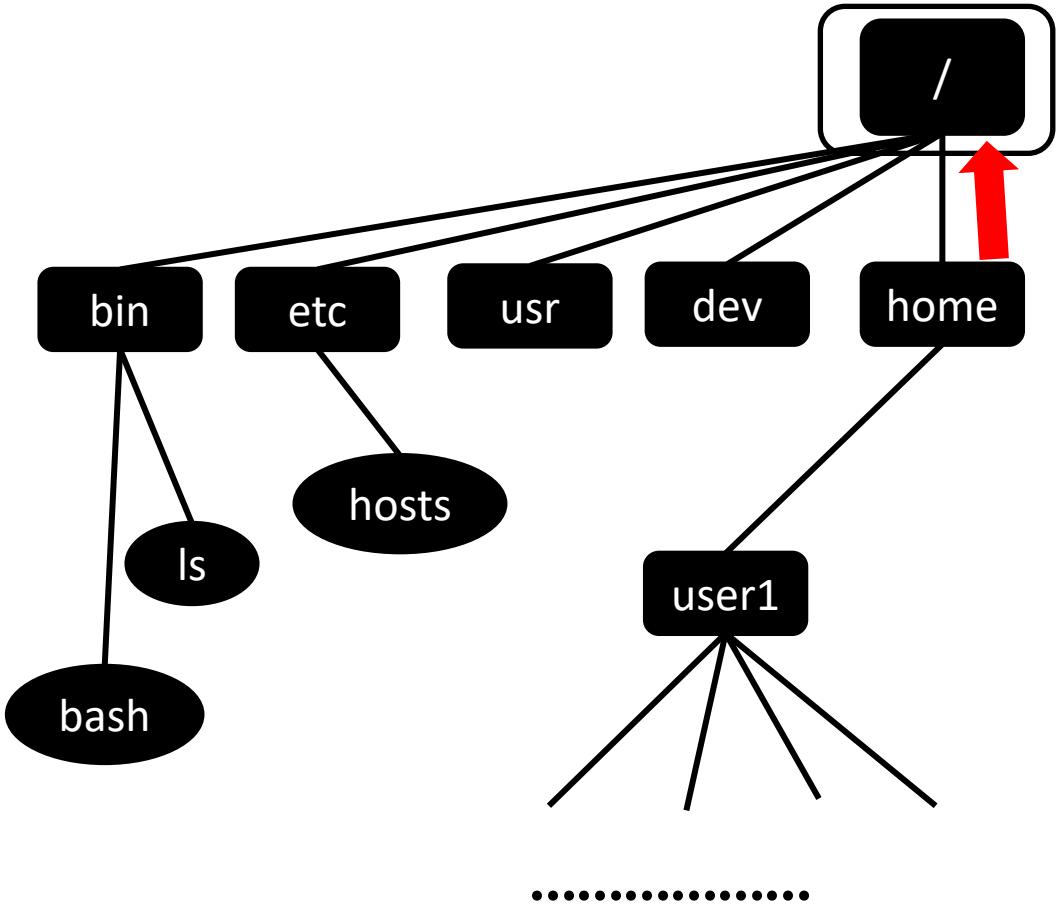
ENTER



```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cd ..
user1@vm2:~$ cd .. ENTER
```

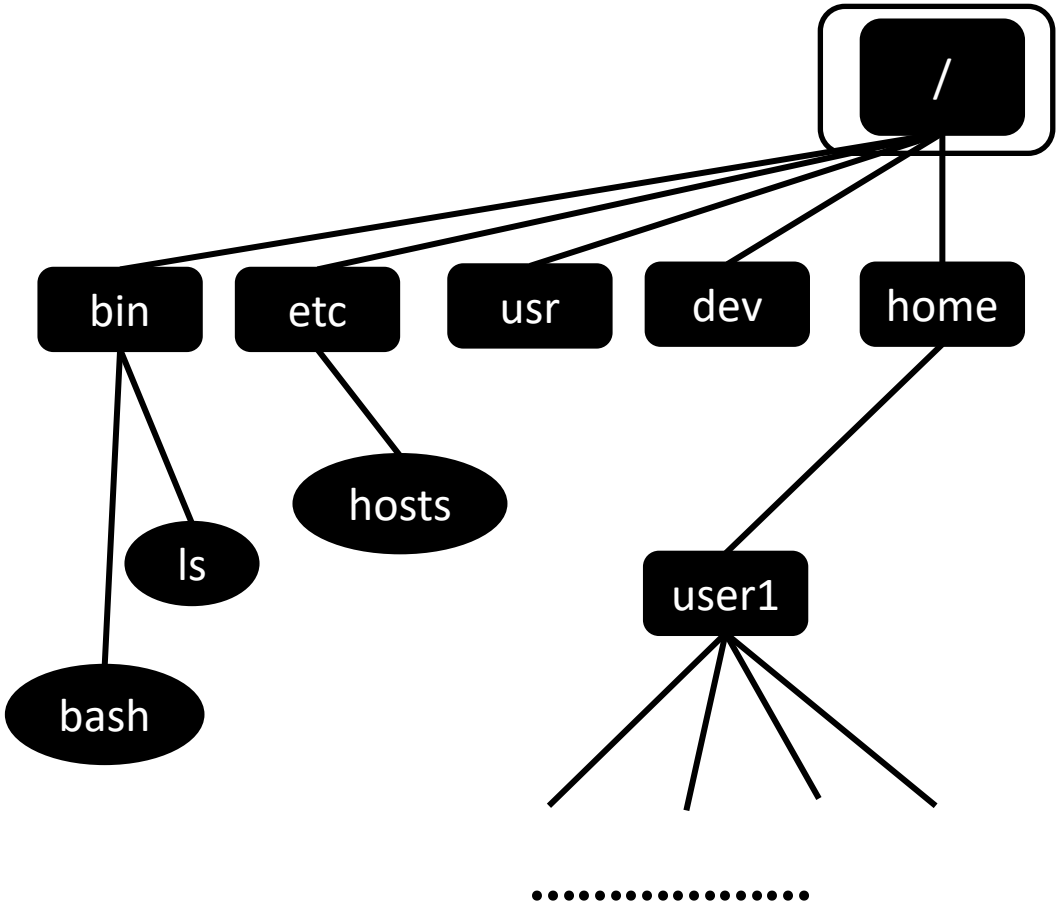


```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cd ..
user1@vm2:~$ cd ..
user1@vm2:/home$ cd .. ENTER
```



```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cd ..
user1@vm2:~$ cd ..
user1@vm2:/home$ cd ..
user1@vm2:/$ cd ..
```

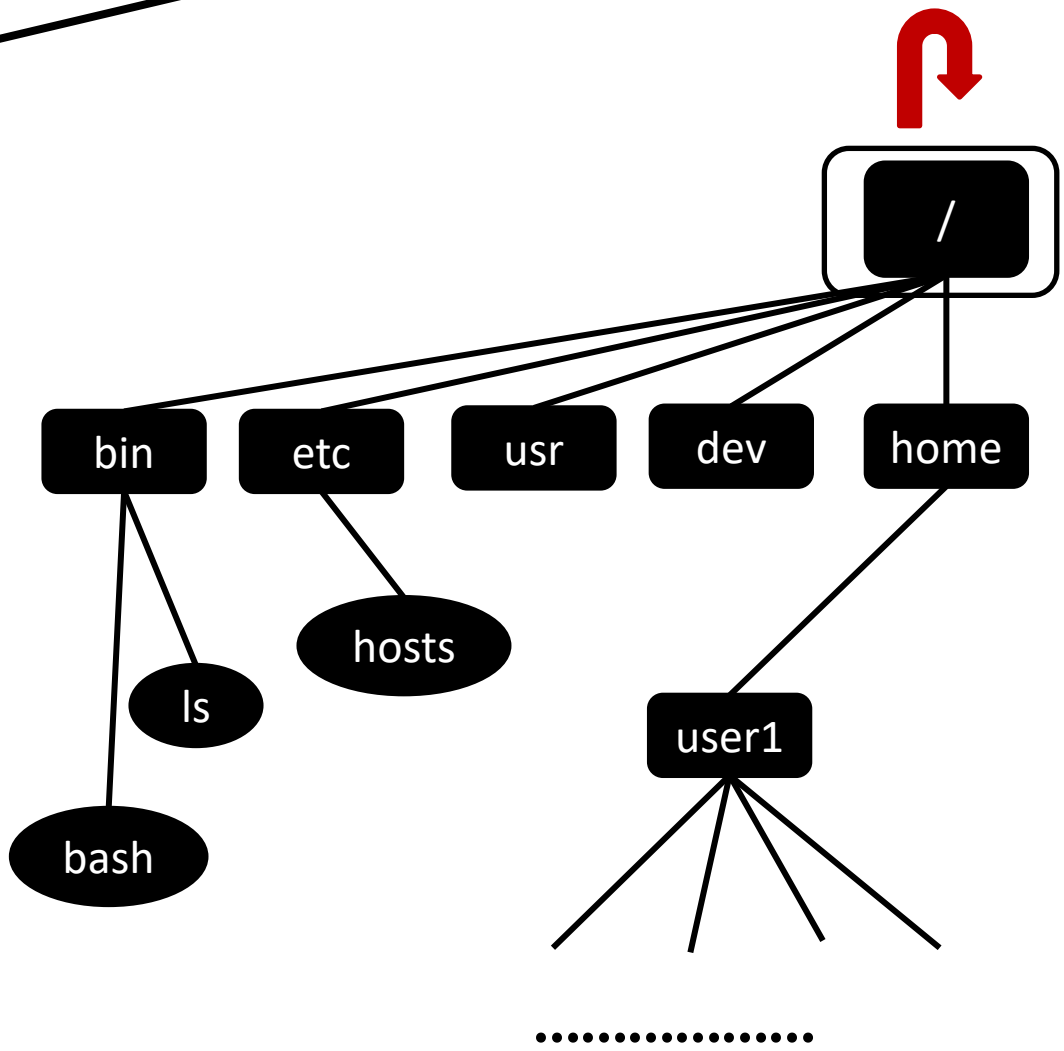
ENTER



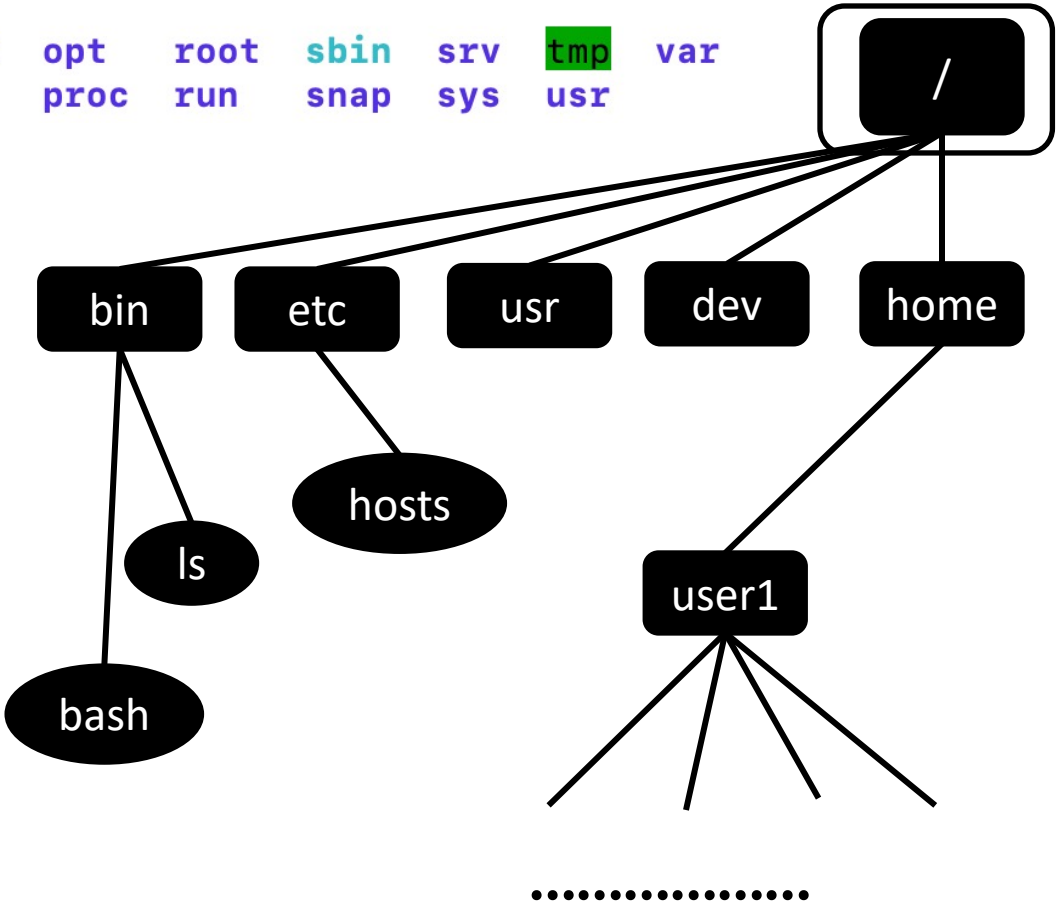
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cd ..
user1@vm2:~$ cd ..
user1@vm2:/home$ cd ..
user1@vm2:/$ cd ..
user1@vm2:/$ cd ..
user1@vm2:/$ pwd
/
```

ENTER

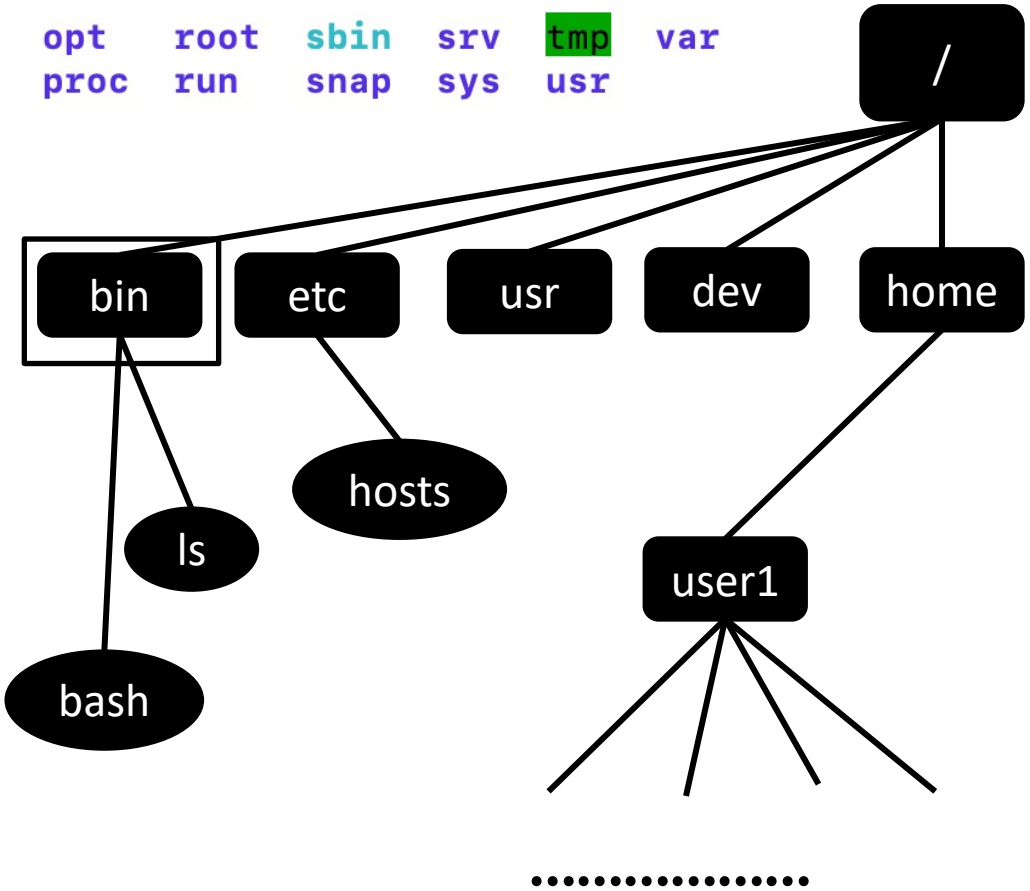
ไดเรกทอรี่พ่อแม่ของรูทก็คือรูท



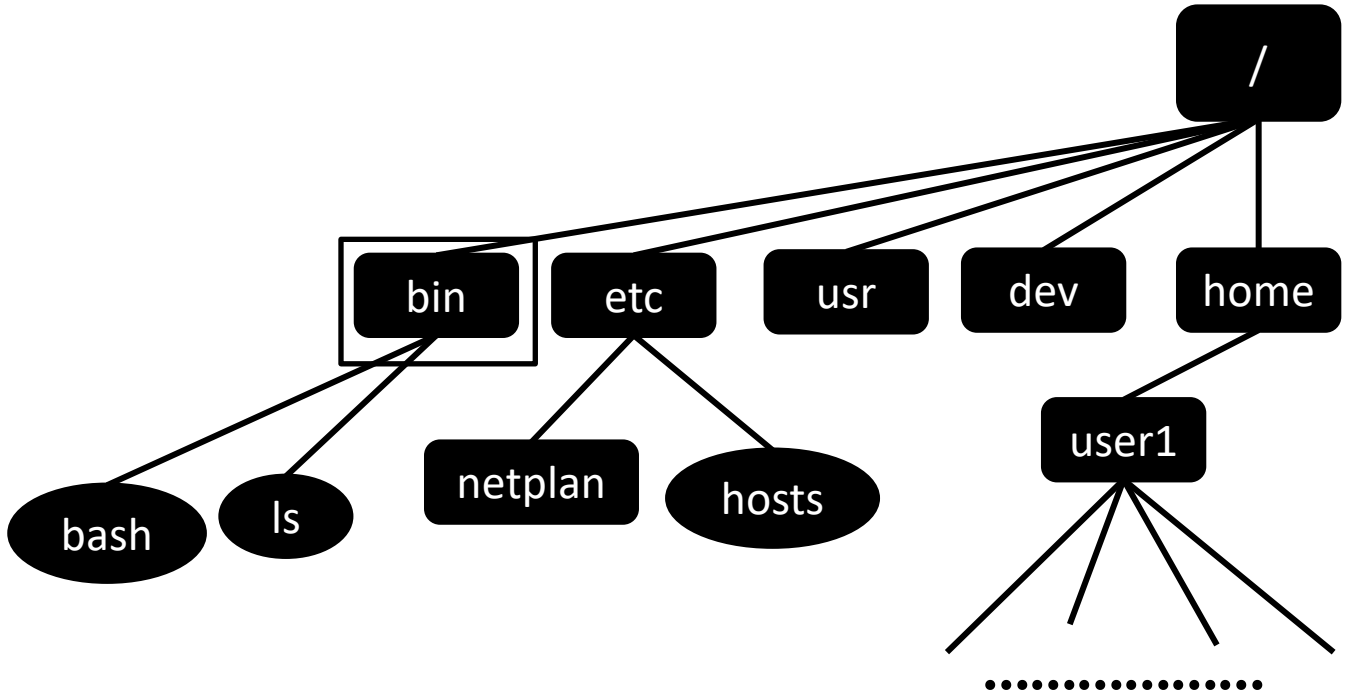
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cd ..
user1@vm2:~$ cd ..
user1@vm2:/home$ cd ..
user1@vm2:/$ cd ..
user1@vm2:/$ cd ..
user1@vm2:/$ pwd
/
user1@vm2:/$ ls
bin   cdrom  etc   lib         media  opt   root  sbin  srv  tmp  var
boot  dev    home  lost+found  mnt    proc  run   snap  sys  usr
```



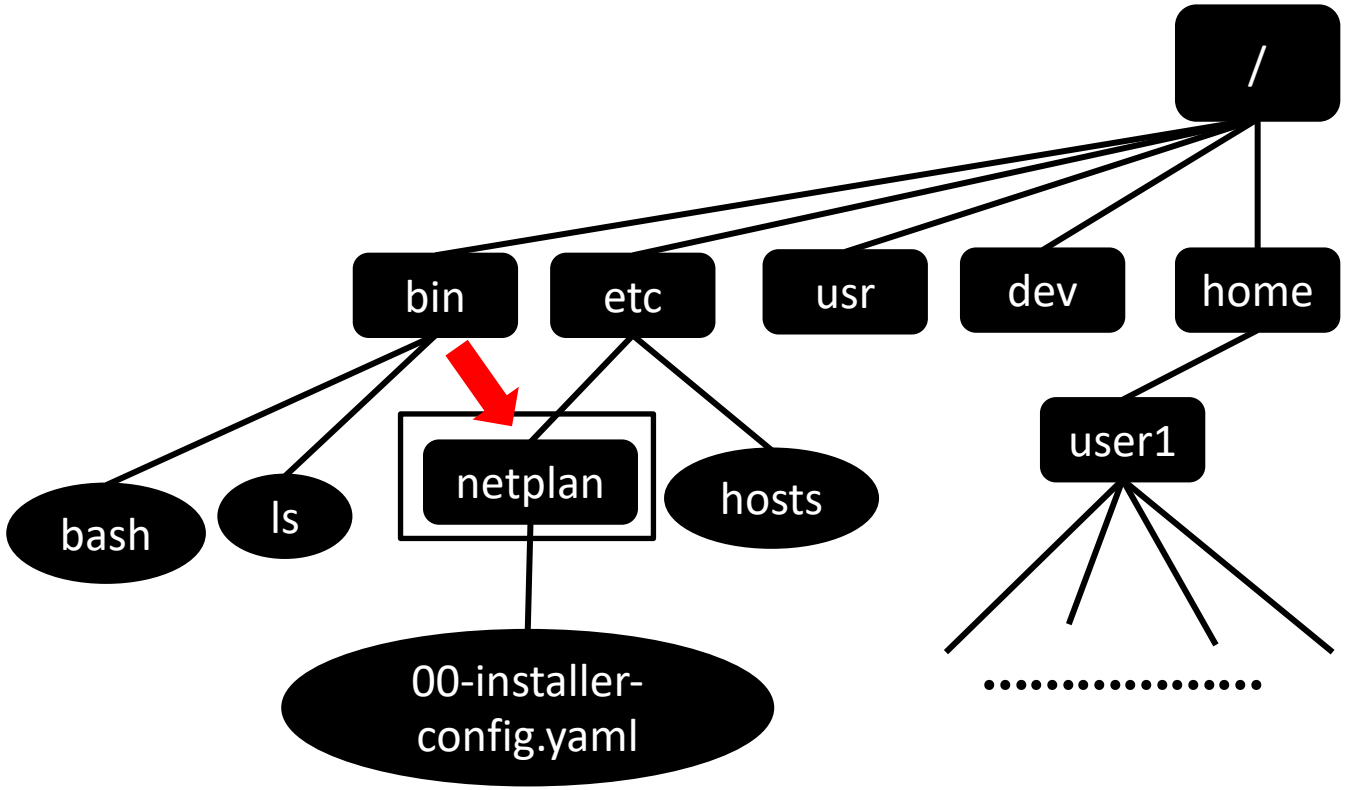
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cd ..
user1@vm2:~$ cd ..
user1@vm2:/home$ cd ..
user1@vm2:/$ cd ..
user1@vm2:/$ cd ..
user1@vm2:/$ pwd
/
user1@vm2:/$ ls
bin   cdrom  etc   lib         media  opt   root  sbin  srv  tmp  var
boot  dev    home  lost+found  mnt    proc  run   snap  sys  usr
user1@vm2:/$ cd bin
```



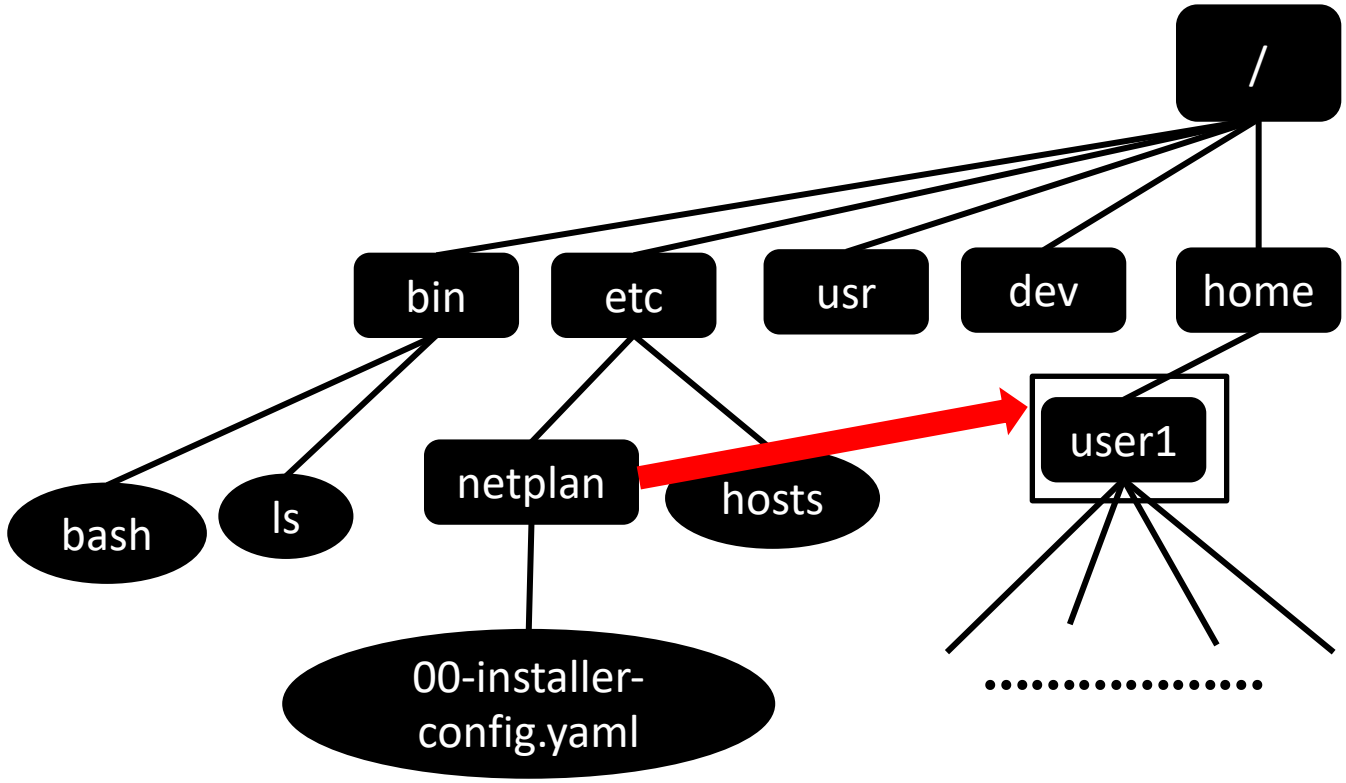
```
user1@vm2:~/mydir1$
user1@vm2:~/mydir1$ cd ..
user1@vm2:~$ cd ..
user1@vm2:/home$ cd ..
user1@vm2:/$ cd ..
user1@vm2:/$ cd ..
user1@vm2:/$ pwd
/
user1@vm2:/$ ls
bin   cdrom  etc   lib         media  opt   root  sbin  srv  tmp  var
boot  dev    home  lost+found  mnt    proc  run   snap  sys  usr
user1@vm2:/$ cd bin
user1@vm2:/bin$ ls
 NF                                gtk-encode-symbolic-svg          sbkeysync
 X11                               gtk-launch                       sbsiglist
'['                                gtk-query-settings               sbsign
 aa-enabled                        gtk-update-icon-cache            sbvarsign
 aa-exec                           gunzip                           sbverify
 aa-features-abi                   gzexe                            scandeps
 aarch64-linux-gnu-addr2line       gzip                             scp
```




```
user1@vm2:/bin$ pwd
/bin
```




```
user1@vm2:/bin$ pwd
/bin
user1@vm2:/bin$ cd /etc/netplan
user1@vm2:/etc/netplan$ ls
00-installer-config.yaml
user1@vm2:/etc/netplan$
```




```
user1@vm2:/bin$ pwd
/bin
user1@vm2:/bin$ cd /etc/netplan
user1@vm2:/etc/netplan$ ls
00-installer-config.yaml
user1@vm2:/etc/netplan$
user1@vm2:/etc/netplan$ cd
user1@vm2:~$ pwd
/home/user1
```

cd แล้วเอ็นเตอร์ คือกลับบ้าน

การใช้คำสั่ง rm

- คำสั่ง rm คือคำสั่งสำหรับลบไฟล์และ directory ต้องใช้อย่างระวังเพราะในระบบ UNIX/Linux พื้นที่ Disk จะถูกใช้ร่วมกันโดยหลาย User ดังนั้นพอไฟล์ถูกลบ พื้นที่ disk อาจถูกนำไปใช้เก็บข้อมูลไฟล์อื่นของ User เดียวกัน หรือ user อื่นทันที

```
$ rm myfile1
  คือการลบไฟล์เดียว
$ rm –rf mydir3
  คือการลบ subtree mydir3 แบบ Recursive ทั้งไฟล์และ
  Dir
$ rm *
  คือการลบไฟล์ทั้งหมดใน Directory * คือ wildcard
$ rm –rf *
  คือการลบทั้งไฟล์และ subtree ทั้งหมดในไดเรกทอรี่ (ควรระวัง)
```

คำสั่ง Linux CLI พื้นฐาน 2

- คำสั่ง cat พิมพ์เนื้อหาขอ file ออกหน้าจอ ถ้าไม่ระบุ input file cat จะพิมพ์ข้อมูลจาก keyboard (stdin) ออกหน้าจอ (stdout)
- คำสั่ง echo แสดงข้อความออกหน้าจอ
- คำสั่ง more แสดงข้อความทีละหน้า
- คำสั่ง man แสดงวิธีการใช้งานคำสั่งและ system calls และ library interfaces ต่างๆ
- คำสั่ง nano ใช้เรียก editor เพื่อเขียนข้อความ text หรือเขียนโปรแกรม
- คำสั่ง tar รวบรวมข้อมูลใน directory ทำให้เป็นไฟล์เดียว
- คำสั่ง gzip บีบอัดข้อมูลในไฟล์

การใช้คำสั่ง cat

- คำสั่ง cat เป็นคำสั่งที่อ่านค่าจาก ไฟล์ และแสดงผลทาง stdout (หน้าจอ)
  - ถ้าไม่ใส่ชื่อไฟล์จะอ่านค่าจาก stdin (คีย์บอร์ด)
- Operator “>” คือการ redirect ค่า output ที่โปรแกรมส่งออก stdout ไปเก็บในไฟล์
- Operator ”<“ คือการ redirect อ่านค่าจากไฟล์แล้วป้อนให้เป็น stdin

```
$ cat myfile3 จะเหมือนกับ $ cat < myfile3
$ cat
Abcd
Abcd
^C
$ cat > mytextfile
Cat will redirect this to the file ”mytestfile”.
^C
$ cat < myfile3 > myfile4
```

การใช้คำสั่ง echo

- คำสั่ง echo เป็นคำสั่งที่ built-in ในโปรแกรมเชล เพื่อพิมพ์ข้อความออกสู่หน้าจอ

```
user1@vm1:~$ ls -l
total 0
user1@vm1:~$ echo hello world
hello world
user1@vm1:~$ echo "hello *"
hello *
user1@vm1:~$ echo hello *
hello *
```

ถ้าไม่มีรายการในไดเรกทอรี่จะพิมพ์ *

การใช้คำสั่ง echo

- คำสั่ง echo เป็นคำสั่งที่ built-in ในโปรแกรมเชล เพื่อพิมพ์ข้อความออกสู่หน้าจอ

```
user1@vm1:~$ ls -l
total 0
user1@vm1:~$ echo hello world
hello world
user1@vm1:~$ echo "hello *"
hello *
user1@vm1:~$ echo hello *
hello *
user1@vm1:~$ mkdir sub1 sub2
user1@vm1:~$ ls -l
total 8
drwxrwxr-x 2 user1 user1 4096 Aug 25 02:17 sub1
drwxrwxr-x 2 user1 user1 4096 Aug 25 02:17 sub2
user1@vm1:~$ touch file1
user1@vm1:~$ ls -l
total 8
-rw-rw-r-- 1 user1 user1    0 Aug 25 02:18 file1
drwxrwxr-x 2 user1 user1 4096 Aug 25 02:17 sub1
drwxrwxr-x 2 user1 user1 4096 Aug 25 02:17 sub2
user1@vm1:~$
user1@vm1:~$ echo hello *
hello file1 sub1 sub2
user1@vm1:~$
```

ถ้าไม่มีรายการในไดเรกทอรี่จะพิมพ์ *

แต่ถ้ามีจะพิมพ์รายชื่อเอ็นติตี้ทั้งหมดในนั้น

การใช้คำสั่ง echo

- คำสั่ง echo เป็นคำสั่งที่ built-in ในโปรแกรมเชล เพื่อพิมพ์ข้อความออกสู่หน้าจอ

```
user1@vm1:~$ echo 'date'
date
user1@vm1:~$ echo $(date)
Wed Aug 25 02:19:05 UTC 2021
user1@vm1:~$ echo Today is $(date)
Today is Wed Aug 25 02:19:23 UTC 2021
user1@vm1:~$ echo "*Today is $(date)*"
*Today is Wed Aug 25 02:19:46 UTC 2021*
user1@vm1:~$ echo $(echo nested hello)
nested hello
user1@vm1:~$ echo $(echo nested hello $(date))
nested hello Wed Aug 25 02:21:08 UTC 2021
user1@vm1:~$ $(date)
Command 'Wed' not found, did you mean:
  command 'sed' from deb sed (4.7-1ubuntu1)
  command 'zed' from deb zfs-zed (2.0.2-1ubuntu5.1)
  command 'jed' from deb jed (1:0.99.19-8)
  command 'red' from deb ed (1.17-1)
  command 'ed' from deb ed (1.17-1)
  command 'med' from deb ncl-ncarg (6.6.2-6build1)
Try: apt install <deb name>
user1@vm1:~$
```

$(.....) จะรันคำสั่งแล้วให้ข้อความผลลัพธ์เป็นสตริง (ถ้ามี)

ใช้ $(.....) เพื่อ nested echo และ date

bash นึกว่า Wed เป็นคำสั่ง

การใช้คำสั่ง echo

- คำสั่ง echo เป็นคำสั่งที่ built-in ในโปรแกรมเชล เพื่อพิมพ์ข้อความออกสู่หน้าจอ

```
user1@vm1:~$ echo 'date'
date
user1@vm1:~$ echo $(date)
Wed Aug 25 02:19:05 UTC 2021
user1@vm1:~$ echo Today is $(date)
Today is Wed Aug 25 02:19:23 UTC 2021
```

$(.....) จะรันคำสั่งแล้วให้ข้อความผลลัพธ์เป็นสตริง (ถ้ามี)

การใช้คำสั่ง echo

- คำสั่ง echo เป็นคำสั่งที่ built-in ในโปรแกรมเชล เพื่อพิมพ์ข้อความออกสู่หน้าจอ

```
user1@vm1:~$ echo 'date'
date
user1@vm1:~$ echo $(date)
Wed Aug 25 02:19:05 UTC 2021
user1@vm1:~$ echo Today is $(date)
Today is Wed Aug 25 02:19:23 UTC 2021
user1@vm1:~$ echo "*Today is $(date)*"
*Today is Wed Aug 25 02:19:46 UTC 2021*
user1@vm1:~$ echo $(echo nested hello)
nested hello
user1@vm1:~$ echo $(echo nested hello $(date))
nested hello Wed Aug 25 02:21:08 UTC 2021
```

$(.....) จะรันคำสั่งแล้วให้ข้อความผลลัพธ์เป็นสตริง (ถ้ามี)

ใช้ $(.....) เพื่อ nested echo และ date

การใช้คำสั่ง echo

- คำสั่ง echo เป็นคำสั่งที่ built-in ในโปรแกรมเชล เพื่อพิมพ์ข้อความออกสู่หน้าจอ

```
user1@vm1:~$ echo 'date'
date
user1@vm1:~$ echo $(date)
Wed Aug 25 02:19:05 UTC 2021
user1@vm1:~$ echo Today is $(date)
Today is Wed Aug 25 02:19:23 UTC 2021
user1@vm1:~$ echo "*Today is $(date)*"
*Today is Wed Aug 25 02:19:46 UTC 2021*
user1@vm1:~$ echo $(echo nested hello)
nested hello
user1@vm1:~$ echo $(echo nested hello $(date))
nested hello Wed Aug 25 02:21:08 UTC 2021
user1@vm1:~$ $(date)
Command 'Wed' not found, did you mean:
  command 'sed' from deb sed (4.7-1ubuntu1)
  command 'zed' from deb zfs-zed (2.0.2-1ubuntu5.1)
  command 'jed' from deb jed (1:0.99.19-8)
  command 'red' from deb ed (1.17-1)
  command 'ed' from deb ed (1.17-1)
  command 'med' from deb ncl-ncarg (6.6.2-6build1)
Try: apt install <deb name>
user1@vm1:~$
```

$(.....) จะรันคำสั่งแล้วให้ข้อความผลลัพธ์เป็นสตริง (ถ้ามี)

ใช้ $(.....) เพื่อ nested echo และ date

bash นึกว่า Wed เป็นคำสั่ง

การใช้คำสั่ง more เพื่อดูข้อความทีละหน้า

```
openstack@cs715host2:~/mydir2$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats
--More--(51%)
```

การใช้คำสั่ง man เพื่อดูคู่มือคำสั่ง

```
openstack@cs715host2:~/mydir2$ man ls
LS(1)                                User Commands                                LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the  FILEs  (the  current directory by default).  Sort entries
       alphabetically if none of -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..
```

การใช้คำสั่ง man เพื่อดูคู่มือ system call

```
openstack@cs715host2:~/mydir2$ man fork
FORK(3am)                        GNU Awk Extension Modules                        FORK(3am)

NAME
       fork, wait, waitpid - basic process management

SYNOPSIS
       @load "fork"

       pid = fork()

       ret = waitpid(pid)

       ret = wait();

DESCRIPTION
       The fork extension adds three functions, as follows.
```

การedit ไฟล์ด้วยคำสั่ง nano

- คำสั่ง nano เป็นคำสั่งสำหรับ edit text ไฟล์ (รองรับภาษาไทย?)
- กด ctrl X เพื่อ save และ exit
- ให้ทดลองใช้คำสั่ง ในบรรทัดล่างด้วยตนเอง (ตัดแปะได้จาก)
  - https://github.com/kasidit/system-programming/blob/master/data/fruits.txt
- **$ nano file2**

```
  GNU nano 5.4                                file2
fruit  unit_price  amount  staff
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

copy directory ลบ directory

- **$ copy –r <source directory> <destination directory>**
- $ ls -l <directory> #แสดงรายการใน directory
- $ rm –rf <directory> #ลบ directory และทุกอย่างในนั้นแบบไม่ถาม
- $ rm * #ลบไฟล์ทั้งหมดใน directory ต้องระวังเวลาใช้

```
$ cp -r /etc/netplan mynetwork
$ ls -l mynetwork
$ rm -rf mynetwork
$ rm *
```

การใช้ Pipe และคำสั่งเพื่อประมวลผล Text files

- Pipe คือเครื่องมือที่อนุญาตให้ shell นำ output ของโปรแกรมหนึ่งมาส่งเป็น input ของอีกโปรแกรมหนึ่ง
- ใช้สัญลักษณ์ |
- ในตัวอย่างถัดไปจะใช้โปรแกรมที่เกี่ยวข้องดังนี้
  - cut เป็นคำสั่งเลือกเอาบางส่วนเช่นคอลัมบางคอลัมจากไฟล์
  - sort เรียงลำดับ, uniq ลบคำเหมือน
  - wc นับตัวอักษร คำ บรรทัด
  - tee นำข้อมูลอินพุท ส่งออกเอ้าพุท และเขียนลงไฟล์
  - sed เป็น stream editor จัดการ edit ไฟล์ได้ตามคำสั่ง

```
user1@vm1:~/sub2/sub1$
user1@vm1:~/sub2/sub1$ ls
file1  file2
user1@vm1:~/sub2/sub1$ cat file2
fruit  unit_price  amount  staff
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d"
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | wc -l
6
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | grep apple
apple 200       30           somnuk
apple 130       100          bob
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1
orange
banana
apple
mango
apple
orange
user1@vm1:~/sub2/sub1$
```

```
user1@vm1:~/sub2/sub1$
user1@vm1:~/sub2/sub1$ ls
file1  file2
user1@vm1:~/sub2/sub1$ cat file2
fruit  unit_price  amount  staff
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
```

```
user1@vm1:~/sub2/sub1$
user1@vm1:~/sub2/sub1$ ls
file1  file2
user1@vm1:~/sub2/sub1$ cat file2
fruit  unit_price  amount  staff
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d"
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
```

ตัด 1 บรรทัดออก

```
user1@vm1:~/sub2/sub1$
user1@vm1:~/sub2/sub1$ ls
file1  file2
user1@vm1:~/sub2/sub1$ cat file2
fruit  unit_price  amount  staff
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d"
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | wc -l
6
```

นับจำนวน บรรทัด

ของ output

```
user1@vm1:~/sub2/sub1$
user1@vm1:~/sub2/sub1$ ls
file1  file2
user1@vm1:~/sub2/sub1$ cat file2
fruit  unit_price  amount  staff
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d"
orange  50           20    somchai
banana  100       7           natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | wc -l
6
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | grep apple
apple 200       30           somnuk
apple 130       100          bob
```

พิมพ์เฉพาะบรรทัดที่

มี string apple

```
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d"
orange  50          20    somchai
banana  100       7          natty
apple 200       30           somnuk
mango 60         20          alice
apple 130       100          bob
orange 10        3000        john
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | wc -l
6
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | grep apple
apple 200       30           somnuk
apple 130       100          bob
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1
orange
banana
apple
mango
apple
orange
user1@vm1:~/sub2/sub1$
```

ใช้ ‘ ‘ ช่องว่าง เป็นตัวแยก Field แลพพิมพ์เฉพาะ field ที่ 1 ของแต่ละบรรทัด

```
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort
apple
apple
banana
mango
orange
orange
```

เรียงลำดับจากน้อยไปมาก

```
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort
apple
apple
banana
mango
orange
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq
apple
banana
mango
orange
```

ตัดคำซ้ำในบรรทัด
ที่ติดกันออก

```
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort
apple
apple
banana
mango
orange
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq
apple
banana
mango
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq | wc -l
4
```

```
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort
apple
apple
banana
mango
orange
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq
apple
banana
mango
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq | wc -l
4
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq > abcd.txt
```

Redirect output
ไปไว้ในไฟล์ abcd.txt

```
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort
apple
apple
banana
mango
orange
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq
apple
banana
mango
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq | wc -l
4
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq > abcd.txt
user1@vm1:~/sub2/sub1$ sort -r < abcd.txt
orange
mango
banana
apple
```

Redirect output
ไปไว้ในไฟล์ abcd.txt

```
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort
apple
apple
banana
mango
orange
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq
apple
banana
mango
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq | wc -l
4
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq > abcd.txt
user1@vm1:~/sub2/sub1$ sort -r < abcd.txt
orange
mango
banana
apple
user1@vm1:~/sub2/sub1$ cat abcd.txt | sort -r | tee abcd2.txt | wc -l
4
```

เรียงจากมากไปน้อย

พิมพ์ไฟล์ออก stdout
และเก็บใน acbd2.txt

```
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort
apple
apple
banana
mango
orange
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq
apple
banana
mango
orange
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq | wc -l
4
user1@vm1:~/sub2/sub1$ cat file2 | sed "1d" | cut -d' ' -f1 | sort | uniq > abcd.txt
user1@vm1:~/sub2/sub1$ sort -r < abcd.txt
orange
mango
banana
apple
user1@vm1:~/sub2/sub1$ cat abcd.txt | sort -r | tee abcd2.txt | wc -l
4
user1@vm1:~/sub2/sub1$ cat abcd2.txt
orange
mango
banana
apple
user1@vm1:~/sub2/sub1$
```

ฝึกด้วยตนเอง

การใช้ pipe และ cut และ grep และ tee และ wc

- ใช้ “nano” เพื่อสร้างไฟล์ข้อมูล

```
$ nano fruits.txt
1 orange       10
2 papaya       5
3 banana       6
4 mango        20
5 tomato       6
6 potato       8
7 pineapple    30
8 apple        70
9 strawberry    1
10 coconut      6
```

การใช้ pipe ร่วมกับคำสั่ง cut และ grep และ sort และ tee และ wc

```
$ cat fruits.txt
$ cat fruits.txt | cut -d' ' -f2
$ cat fruits.txt | cut -d' ' -f2 | sort
$ cat fruits.txt | cut -d' ' -f2 | grep pp
$ cat fruits.txt | cut -d' ' -f2 | grep pp \
> | wc -l
$ cat fruits.txt | cut -d' ' -f2 | grep m \
> | tee fwithm.txt | sort
$ cat fruits.txt | cut -d' ' -f2 | grep m \
> | tee fwithm.txt | sort | wc -l
$ ls -l; cat fwithm.txt
```

การใช้ pipe ร่วมกับ who และ cut และ sort

openstack@source: ~

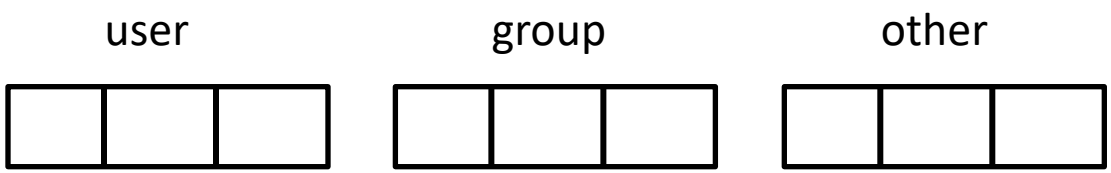
```
openstack@source:~$ who
openstack tty1         2019-01-29 13:39
openstack pts/0        2019-01-29 13:55 (192.168.56.1)
u01       pts/1        2019-01-29 14:27 (192.168.56.1)
openstack@source:~$ who | cut -d' ' -f1
openstack
openstack
u01
openstack@source:~$ who | cut -d' ' -f1 | sort
openstack
openstack
u01
openstack@source:~$ man sort
openstack@source:~$ who | cut -d' ' -f1 | sort -r
u01
openstack
openstack
openstack@source:~$ who | cut -d' ' -f1 | sort -r | uniq
u01
openstack
openstack@source:~$ who | cut -d' ' -f1 | sort -r | uniq | wc -l
2
openstack@source:~$
```

Redirection

- ทุก process ใน Linux มีไฟล์ 3 ไฟล์ที่ถูกเปิดตั้งแต่เริ่มต้นการทำงาน ได้แก่ stdin (keyboard) และ stdout (Display Terminal) และ stderr (Display Terminal)
- ค่า file descriptor ของ stdin stdout และ stderr คือ 0 1 และ 2
- Redirection คือการนำข้อมูลที่ส่งออก output ทาง stdout หรือทาง stderr เก็บลงไฟล์ และการนำข้อมูลจากไฟล์ส่งเข้าสู่ stdin
- ตัวอย่างเช่น ในคำสั่ง cat และ find ที่ได้กล่าวถึงไปก่อนหน้า

Permission Flags สำหรับไฟล์

- Permission flag คือข้อมูลเมตาดาต้าของไฟล์แต่ละไฟล์ที่ประกอบไปด้วยข้อมูลเลขฐานแปดสามตัวเลขที่ระบุขอบเขตการใช้งานของไฟล์นั้น



- ไฟล์ในระบบไฟล์ มีขอบเขตของการใช้งานตามที่ผู้ที่เป็นเจ้าของไฟล์หรือผู้สร้างไฟล์อนุญาต ซึ่งสำหรับไฟล์หนึ่งที่ถูกสร้างขึ้นจะมีผู้ใช้สามแบบได้แก่
  - เจ้าของ (user)
  - บัญชีผู้ใช้อื่นที่อยู่ในกลุ่มเดียวกัน (group)
  - บัญชีผู้ใช้อื่นทั้งหมด (other)
- กำหนดว่าแต่ละแบบมีสิทธิอ่านเขียนและรันไฟล์ได้หรือไม่

Permission Flags

- user group และ other มีสิทธิทำอะไรได้บ้าง
- ชนิดของสิทธิมีสามชนิด
  - Read : อนุญาตให้อ่านไฟล์ได้
  - write : อนุญาตให้เขียนไฟล์ได้ (แก้ไข และ ลบ)
  - execute : อนุญาตให้รันไฟล์ได้
- ตำแหน่งของแต่ละสิทธิคือ

user			group			other		
r	w	x	r	w	x	r	w	x

- ถ้ามีสัญลักษณ์ “-” แทนที่ ”r” “w” “x” แต่ละตำแหน่ง ก็หมายถึงไม่มีสิทธิ

Permission Flags

no	Owner	Group	Other
1	r w x	- - -	- - -
	1 1 1	0 0 0	0 0 0
	7	0	0
2	- - -	- - -	- - -
	0 0 0	0 0 0	0 0 0
	0	0	0
3	r w -	r - -	r - -
	1 1 0	1 0 0	1 0 0
	6	4	4
4	r w x	r w x	r w x
	1 1 1	1 1 1	1 1 1
	7	7	7

24/Aug/2021

การกำหนดค่า Permission ด้วย absolute number

- ผู้ใช้สามารถใช้คำสั่ง chmod เพื่อกำหนดค่า permission flag ให้กับไฟล์

  chmod <permission flag> <file name>

- กำหนดให้ ผู้ใช้เจ้าของ ไฟล์ myscript สามารถอ่านได้ เขียนและลบได้ และรันไฟล์นี้ได้ แต่ไม่ให้ผู้อื่นไม่ว่าจะอยู่ในกลุ่มเดียวกันหรือต่างกลุ่มอ่าน เขียนหรือรันได้เลย

  chmod 700 myscript.sh

- กำหนดให้ เจ้าของ และผู้ใช้อื่นในกลุ่มเดียวกันดูไฟล์ข้อมูลนี้ได้เท่านั้น

  chmod 660 mydata.txt

Absolute Permission Flag

- rwxrwxrwx
- 111111111 = 777_8
- rw-rw-rw-
- 110110110 = 666_8
- r-xr-xr—
- 101101100 = 554_8
- r--r--r--
- 100100100 = 444_8
- ---------
- 000000000 = 000_8

```
kasidit@enterprise:~/Desktop/CS222$ cp /etc/hosts abcd.txt
kasidit@enterprise:~/Desktop/CS222$
kasidit@enterprise:~/Desktop/CS222$ ls -l
total 4
-rw-r--r-- 1 kasidit kasidit 225 ม.ค.  31 14:31 abcd.txt
kasidit@enterprise:~/Desktop/CS222$
kasidit@enterprise:~/Desktop/CS222$ chmod 444 abcd.txt
kasidit@enterprise:~/Desktop/CS222$ ls -l
total 4
-r--r--r-- 1 kasidit kasidit 225 ม.ค.  31 14:31 abcd.txt
kasidit@enterprise:~/Desktop/CS222$
kasidit@enterprise:~/Desktop/CS222$ rm abcd.txt
rm: remove write-protected regular file 'abcd.txt'? n
kasidit@enterprise:~/Desktop/CS222$ chmod 000 abcd.txt
kasidit@enterprise:~/Desktop/CS222$ ls -l
total 4
---------- 1 kasidit kasidit 225 ม.ค.  31 14:31 abcd.txt
kasidit@enterprise:~/Desktop/CS222$
kasidit@enterprise:~/Desktop/CS222$ rm abcd.txt
rm: remove write-protected regular file 'abcd.txt'? y
kasidit@enterprise:~/Desktop/CS222$ ls -l
total 0
kasidit@enterprise:~/Desktop/CS222$
```

จำนวนไฟล์ซิสเต็ม block ที่ใช้

permission หลังจาก copy ไฟล์

- ค่า permission ของ destination file จะเหมือนกับของ source file

```
kasidit@enterprise:~/Desktop/CS222$ ls -l /etc/hosts
-rw-r--r-- 1 root root 225 เม.ย.  7  2023 /etc/hosts
kasidit@enterprise:~/Desktop/CS222$
kasidit@enterprise:~/Desktop/CS222$ cp /etc/hosts abcd.txt
kasidit@enterprise:~/Desktop/CS222$ ls -l
total 4
-rw-r--r-- 1 kasidit kasidit 225 ม.ค.  31 13:02 abcd.txt
kasidit@enterprise:~/Desktop/CS222$
```

ชื่อไฟล์

วันเวลาที่สร้าง

ขนาดของไฟล์

ชื่อ account เจ้าของ และกลุ่มของเจ้าของ

Reference count ของไฟล์

ไฟล์ permission

- คือไฟล์

d คือไดเรกทอรี่

- ถ้าเป็นไฟล์ปกติเช่น text file หรือ data file ค่า Default permission เมื่อ linux create ไฟล์คือ 666
- แต่ถ้าเป็น **executable file** หรือ**ไดเรกทอรี่** ค่า Default permission เมื่อ linux create ไฟล์คือ 777
- หลังจากสร้างไฟล์แล้ว ระบบจะเอา permission มาผ่านการกำหนดค่าของ user ซึ่ง user สามารถกำหนดได้ว่าอยากปิดกั้นหรือเปิด permission ใด โดยการกำหนดค่า umask flag
- Default umask flag ของ linux user คือ 002
- ดังนั้น permission ที่เราจะได้คือ จากสมการ

  Original Permission **<u>bitwise and</u>** (NOT umask)

  – 666 and (NOT 002) = 666 and 775 = 664

  – 777 and (NOT 002) = 777 and 775 = 775

permission หลังจากสร้างไฟล์ใหม่

- ค่า Default permission ของ text ไฟล์หลังจาก umask คือ 664

```
kasidit@enterprise:~/Desktop/CS222$ touch xyz.txt
kasidit@enterprise:~/Desktop/CS222$ ls -l
total 4
-rw-r--r-- 1 kasidit kasidit 225 ม.ค.  31 13:02 abcd.txt
-rw-rw-r-- 1 kasidit kasidit   0 ม.ค.  31 13:03 xyz.txt
kasidit@enterprise:~/Desktop/CS222$
kasidit@enterprise:~/Desktop/CS222$ nano xyz2.txt
kasidit@enterprise:~/Desktop/CS222$ ls -l
total 8
-rw-r--r-- 1 kasidit kasidit 225 ม.ค.  31 13:02 abcd.txt
-rw-rw-r-- 1 kasidit kasidit  21 ม.ค.  31 13:15 xyz2.txt
-rw-rw-r-- 1 kasidit kasidit   0 ม.ค.  31 13:03 xyz.txt
kasidit@enterprise:~/Desktop/CS222$ cat > xyz3.txt
hello hello
kasidit@enterprise:~/Desktop/CS222$ ls -l
total 12
-rw-r--r-- 1 kasidit kasidit 225 ม.ค.  31 13:02 abcd.txt
-rw-rw-r-- 1 kasidit kasidit  21 ม.ค.  31 13:15 xyz2.txt
-rw-rw-r-- 1 kasidit kasidit  12 ม.ค.  31 13:15 xyz3.txt
-rw-rw-r-- 1 kasidit kasidit   0 ม.ค.  31 13:03 xyz.txt
kasidit@enterprise:~/Desktop/CS222$
```

permission หลังจากสร้างไฟล์ใหม่

- ค่า Default permission ของ exe ไฟล์หลังจาก umask คือ 775

```
kasidit@enterprise:~/Desktop/software/contagent/client$ gcc pgm1-echo-client.c
kasidit@enterprise:~/Desktop/software/contagent/client$
kasidit@enterprise:~/Desktop/software/contagent/client$ ls -l
total 24
-rwxrwxr-x 1 kasidit kasidit 16688 ม.ค.  31 14:26 a.out
-rw-rw-r-- 1 kasidit kasidit  1279 พ.ย.  26 00:51 pgm1-echo-client.c

kasidit@enterprise:~/Desktop/software/contagent/client$ file a.out
a.out: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linke
d, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=f33e74a221f3525c1022dc1
c594907e262796023, for GNU/Linux 3.2.0, not stripped
```

คำสั่ง find

- ตัวอย่าง $ find <directory> -name <regex>
- เป็นคำสั่งสำหรับหาไดเรกทอรี่หรือไฟล์ที่มีชื่อตามที่กำหนด โดยเริ่มต้นหาจากไดเรกทอรี่ที่กำหนดและค้นหาลงใน ไดเรกทอรี่ย่อยในโครงสร้างต้นไม้จากไดเรกทอรี่ที่กำหนดไปจนกระทั่งพบหรือหาไม่พบ
- ผู้ใช้สามารถระบุชื่อที่จะค้นหาด้วย find เป็นแบบ regex
- ในกรณีที่ค้นหานอก $HOME ไดเรกทอรี่ อาจมีข้อความแจ้งว่าไม่มี permission แสดงทาง stderr
- ผู้ใช้สามารถใช้ redirection “2>” เพื่อส่งค่าจาก stderr ไปเก็บในไฟล์ หรือส่งไปทิ้งที่ /dev/null (null device)

```
user1@vm1:~$
user1@vm1:~$ ls */*
sub2/myhosts

sub2/sub1:
file1  file2
user1@vm1:~$
user1@vm1:~$ find . -name file2
./sub2/sub1/file2
user1@vm1:~$ find . -name sub1
./sub2/sub1
user1@vm1:~$ find . -name file3
user1@vm1:~$
user1@vm1:~$ find /etc -name *plan
find: '/etc/ssl/private': Permission denied
find: '/etc/polkit-1/localauthority': Permission denied
find: '/etc/multipath': Permission denied
/etc/netplan
user1@vm1:~$
user1@vm1:~$ find /etc -name *plan 2> err.txt
/etc/netplan
user1@vm1:~$ find /etc -name *plan 2> /dev/null
/etc/netplan
user1@vm1:~$
user1@vm1:~$
```

ไดเรกทอรี่เริ่มต้นการค้นหา

รับ regex

การทำ command substitution ด้วย $()

- command substitution คือการที่เชลรันคำสั่งย่อยก่อนแล้วนำผลลัพธ์ของคำสั่งนั้นใช้เป็นสตริง
- รูปแบบ: $(command) โดยที่ command คือคำสั่งเชลคอมมานด์

```
user1@vm1:~/mydir$ echo Today is $(date)
Today is Wed Aug 25 10:20:59 UTC 2021
user1@vm1:~/mydir$ echo my dir has $(ls)
my dir has link3 link4 myfile1 myfile3
user1@vm1:~/mydir$ echo my hostname is $(hostname)
my hostname is vm1
```

Redirection

- ทุกโปรเซสมี ไฟล์เปิดสามไฟล์โดยอัตโนมัติ
- FD 0: stdin, FD 1: stdout, FD 2: stderr
- โอเปอร์เรเตอร์ “> <filename>” เป็นนำข้อมูลจาก stdout ไปเก็บไฟล์
- โอเปอร์เรเตอร์ “< <filename>“ เป็นนำข้อมูลจากไฟล์ส่งเข้า stdin
- โอเปอร์เรเตอร์ “2> <filename>” เป็นนำข้อมูลจาก stderr ไปเก็บไฟล์
- โอเปอร์เรเตอร์ “1> <filename>” เป็นนำข้อมูลจาก stdout ไปเก็บไฟล์
- โอเปอร์เรเตอร์ “2>&1” เป็นนำข้อมูลจาก stderr ส่งไป stdout
- ข้างต้น ถ้า <filename> มีอยู่แล้วจะเก็บข้อมูลทับ
- โอเปอร์เรเตอร์ “>> <filename>” เป็นนำข้อมูลจาก stdout ไปเก็บไฟล์ ที่ถ้า <filename> มีอยู่แล้วจะเก็บข้อมูลต่อจากบรรทัดสุดท้าย

```
user1@vm1:~$ ls
mydir  mydir2  sub2
user1@vm1:~$
user1@vm1:~$ ls mydir
link3  link4  myfile1  myfile3
user1@vm1:~$ cat mydir/myfile3 > content1.txt
user1@vm1:~$ ls -l content1.txt
-rw-rw-r-- 1 user1 user1 218 Aug 25 14:47 content1.txt
```

Redirect stdout

ไปเก็บใน content1.txt

```
user1@vm1:~$ ls
mydir  mydir2  sub2
user1@vm1:~$
user1@vm1:~$ ls mydir
link3  link4  myfile1  myfile3
user1@vm1:~$ cat mydir/myfile3 > content1.txt
user1@vm1:~$ ls -l content1.txt
-rw-rw-r-- 1 user1 user1 218 Aug 25 14:47 content1.txt
user1@vm1:~$
user1@vm1:~$ cat mydir > content2.txt
cat: mydir: Is a directory
```

ผมพยายามที่จะ cat ไดเรกทอรี่
ซึ่งจะเกิด error และจะนำ
สิ่งที่ปรากฏบนหน้าจอไปไวในไฟล์ content2.txt

```
user1@vm1:~$ ls
mydir  mydir2  sub2
user1@vm1:~$
user1@vm1:~$ ls mydir
link3  link4  myfile1  myfile3
user1@vm1:~$ cat mydir/myfile3 > content1.txt
user1@vm1:~$ ls -l content1.txt
-rw-rw-r-- 1 user1 user1 218 Aug 25 14:47 content1.txt
user1@vm1:~$
user1@vm1:~$ cat mydir > content2.txt
cat: mydir: Is a directory
user1@vm1:~$ ls -l content2.txt
-rw-rw-r-- 1 user1 user1 0 Aug 25 14:48 content2.txt
user1@vm1:~$
```

content2.txt มีเนื้อหาเป็น 0 คือไม่มีเนื้อหา เพราะไม่มีผลทาง stdout มีแต่ error message ทาง stderr

```
user1@vm1:~$ ls
mydir  mydir2  sub2
user1@vm1:~$
user1@vm1:~$ ls mydir
link3  link4  myfile1  myfile3
user1@vm1:~$ cat mydir/myfile3 > content1.txt
user1@vm1:~$ ls -l content1.txt
-rw-rw-r-- 1 user1 user1 218 Aug 25 14:47 content1.txt
user1@vm1:~$
user1@vm1:~$ cat mydir > content2.txt
cat: mydir: Is a directory
user1@vm1:~$ ls -l content2.txt
-rw-rw-r-- 1 user1 user1 0 Aug 25 14:48 content2.txt
user1@vm1:~$
user1@vm1:~$ cat mydir 2> err.txt
user1@vm1:~$ ls -l err.txt
-rw-rw-r-- 1 user1 user1 27 Aug 25 14:49 err.txt
user1@vm1:~$ cat err.txt
cat: mydir: Is a directory
user1@vm1:~$
```

ต้องเปลี่ยนไปใช้ 2> content2.txt แทน

```
user1@vm1:~$ cat mydir > content2.txt 2>&1
user1@vm1:~$ cat content2.txt
cat: mydir: Is a directory
user1@vm1:~$
user1@vm1:~$ find /etc -name netplan > content2.txt 2>&1
user1@vm1:~$ cat content2.txt
find: '/etc/ssl/private': Permission denied
find: '/etc/polkit-1/localauthority': Permission denied
find: '/etc/multipath': Permission denied
/etc/netplan
user1@vm1:~$
```

การใช้ 2>&1 ทำให้ error message ถูก redirect จาก stderr ไปยัง stdout

3 บรรทัดแรกเป็น error message บรรทัดสุดท้ายเป็น output

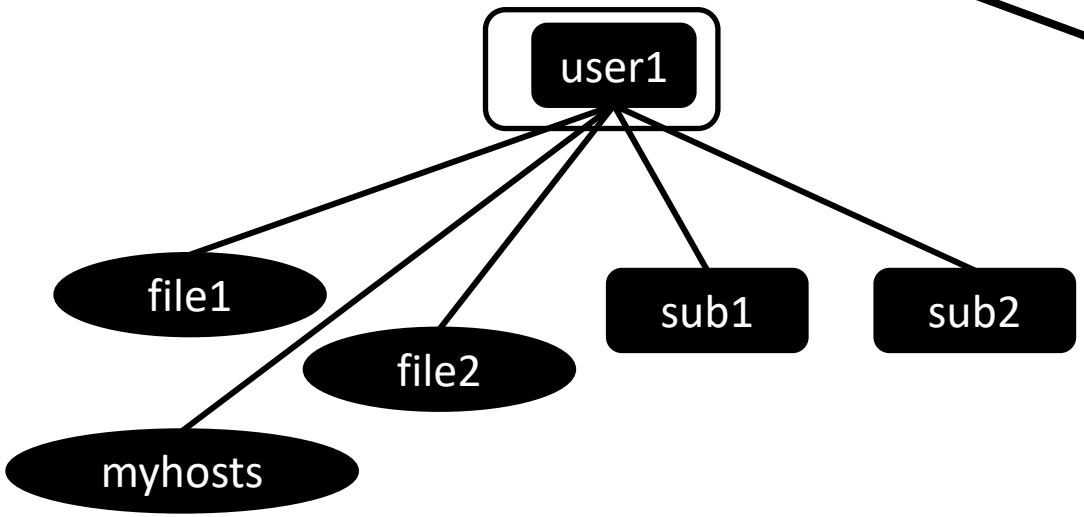
เป็นการ Redirect จาก stdout ไปยัง content2.txt

การสร้าง archive ไฟล์ด้วยคำสั่ง tar และใช้ gzip

- คำสั่ง tar เป็นคำสั่งสำหรับสร้าง archive ไฟล์ (หรือทาร์ tar ไฟล์ หรือ tarball) หรือไฟล์ที่รวบรวมเนื้อหาของโครงสร้าง subtree ภายไดเรกทอรี่ที่ระบุทั้งหมด ไว้ในไฟล์เดียว
- ผู้ใช้สามารถ compress และส่งทาร์ไฟล์ไปที่อื่น และ
- สามารถแตกไฟล์นั้นออกเป็นโครงสร้าง subtree แบบเดิมได้
- $ tar <option> <tar file name> <ชื่อ directory หรือ ลิสต์ของ files>
- $ gzip <option> <file>

การสร้าง archive ไฟล์ด้วยคำสั่ง tar

```
user1@vm1:~$ ls
file1  file2  sub1  sub2
user1@vm1:~$
user1@vm1:~$ cp /etc/hosts myhosts
```



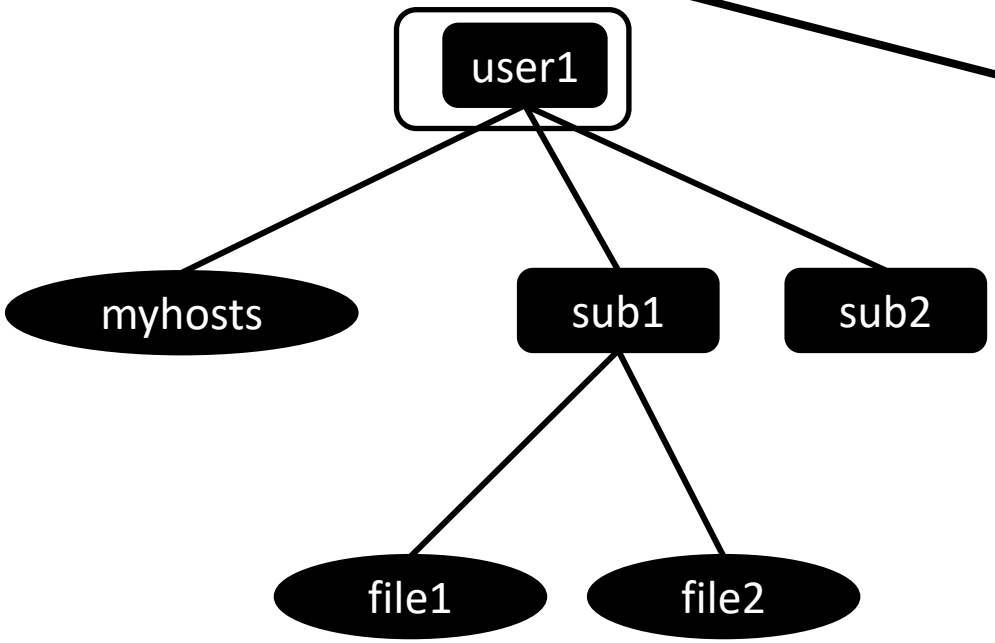
ถ้าใช้คำสั่ง ls จะมีผลลัพธ์เป็นอย่างไร

การสร้าง archive ไฟล์ด้วยคำสั่ง tar

```
user1@vm1:~$ ls
 file1  file2  sub1  sub2
user1@vm1:~$
user1@vm1:~$ cp /etc/hosts myhosts
user1@vm1:~$ mv f* sub1
```

ย้ายไฟล์ทุกไฟล์ที่มีชื่อนำหน้าด้วย f ไปไว้ใน Directory sub1

ถ้าใช้คำสั่ง ls จะมีผลลัพธ์เป็นอย่างไร

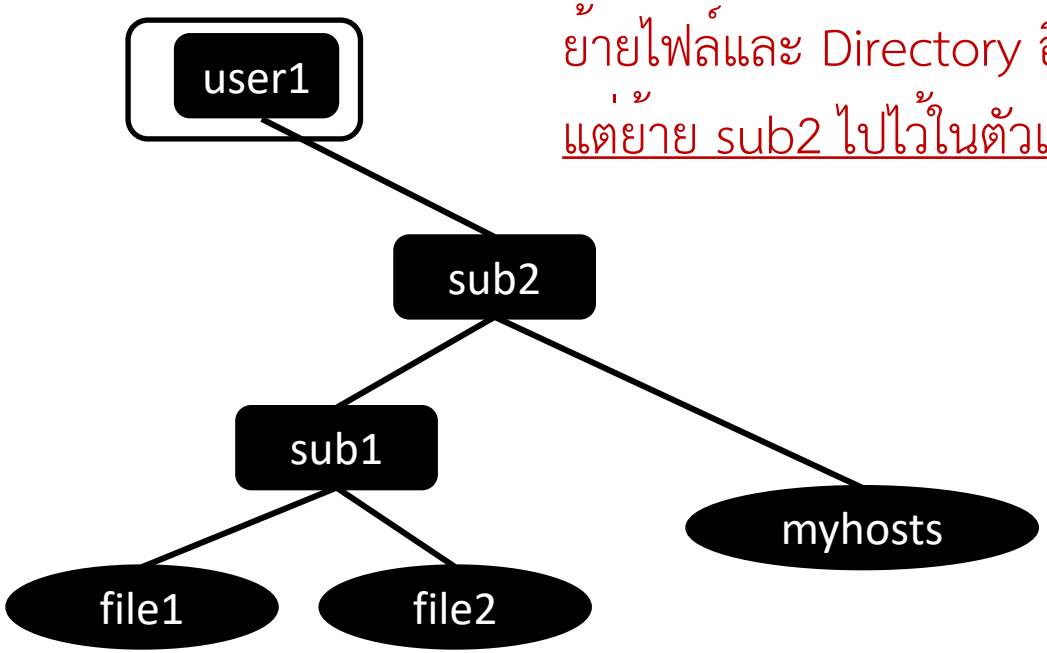


การสร้าง archive ไฟล์ด้วยคำสั่ง tar

```
user1@vm1:~$ ls
file1  file2  sub1  sub2
user1@vm1:~$
user1@vm1:~$ cp /etc/hosts myhosts
user1@vm1:~$ mv f* sub1
user1@vm1:~$ mv * sub2
mv: cannot move 'sub2' to a subdirectory of itself, 'sub2/sub2'
user1@vm1:~$
```

พยายามย้ายทุกไฟล์ทุกไฟล์
และ directory ไปไว้ใน
Directory sub2

ย้ายไฟล์และ Directory อื่นไปไว้ใน sub2 ได้หมด
<u>แต่ย้าย sub2 ไปไว้ในตัวเองไม่ได้</u>



การสร้าง archive ไฟล์ด้วยคำสั่ง tar

```
user1@vm1:~$ ls
file1  file2  sub1  sub2
user1@vm1:~$
user1@vm1:~$ cp /etc/hosts myhosts
user1@vm1:~$ mv f* sub1
user1@vm1:~$ mv * sub2
mv: cannot move 'sub2' to a subdirectory of itself, 'sub2/sub2'
user1@vm1:~$
user1@vm1:~$ ls */*
sub2/myhosts

sub2/sub1:
file1  file2
```

1. List ชื่อไฟล์ใน directory ปัจจุบันและชื่อไฟล์และชื่อไดเรกทอรีใน subdirectory ของ directory ปัจจุบัน
2. list ชื่อไฟล์และ subdirectory ของทุก subdirectory

<u>ถ้าออกคำสั่ง $ ls * ผลลัพธีจะต่างจาก $ ls */* อย่างไร</u>

การสร้าง archive ไฟล์ด้วยคำสั่ง tar

```
user1@vm1:~$ ls
file1  file2  sub1  sub2
user1@vm1:~$
user1@vm1:~$ cp /etc/hosts myhosts
user1@vm1:~$ mv f* sub1
user1@vm1:~$ mv * sub2
mv: cannot move 'sub2' to a subdirectory of itself, 'sub2/sub2'
user1@vm1:~$
user1@vm1:~$ ls */*
sub2/myhosts

sub2/sub1:
file1  file2
user1@vm1:~$ tar cvf mytar1.tar sub2
sub2/
sub2/myhosts
sub2/sub1/
sub2/sub1/file1
sub2/sub1/file2
user1@vm1:~$ ls -l
total 16
-rw-rw-r-- 1 user1 user1 10240 Aug 25 03:16 mytar1.tar
drwxrwxr-x 3 user1 user1  4096 Aug 25 03:14 sub2
user1@vm1:~$
```

คำสั่งเหล่านี้ทำให้โครงสร้างไดเรกทอรี่เปลี่ยนอย่างไร (กราฟ)

คำสั่ง tar cvf

c = create

v = verbose

f = ใช้ชื่อถัดไปเป็นชื่อ archive file

การ compress ไฟล์ด้วย gzip

คำสั่งบีบอัดไฟล์

```
user1@vm1:~$ ls -l
total 16
-rw-rw-r-- 1 user1 user1 10240 Aug 25 03:16 mytar1.tar
drwxrwxr-x 3 user1 user1  4096 Aug 25 03:14 sub2
user1@vm1:~$
user1@vm1:~$ gzip mytar1.tar
user1@vm1:~$
user1@vm1:~$ ls -l
total 8
-rw-rw-r-- 1 user1 user1  479 Aug 25 03:16 mytar1.tar.gz
drwxrwxr-x 3 user1 user1 4096 Aug 25 03:14 sub2
user1@vm1:~$
user1@vm1:~$ mv mytar1.tar.gz /tmp
user1@vm1:~$ cd /tmp
```

ย้ายทาร์ไฟล์

การ uncompress/untar ไฟล์

```
user1@vm1:/tmp$ ls
mytar1.tar.gz
snap.lxd
systemd-private-c34050aa8824428aae2ce7cfdb2ee666-systemd-logind.service-6p9Km5
systemd-private-c34050aa8824428aae2ce7cfdb2ee666-systemd-resolved.service-j2NotJ
systemd-private-c34050aa8824428aae2ce7cfdb2ee666-systemd-timesyncd.service-GsW9ym
user1@vm1:/tmp$
user1@vm1:/tmp$ gzip -d mytar1.tar.gz
user1@vm1:/tmp$ ls
mytar1.tar
snap.lxd
systemd-private-c34050aa8824428aae2ce7cfdb2ee666-systemd-logind.service-6p9Km5
systemd-private-c34050aa8824428aae2ce7cfdb2ee666-systemd-resolved.service-j2NotJ
systemd-private-c34050aa8824428aae2ce7cfdb2ee666-systemd-timesyncd.service-GsW9ym
user1@vm1:/tmp$ tar xvf mytar1.tar
sub2/
sub2/myhosts
sub2/sub1/
sub2/sub1/file1
sub2/sub1/file2
user1@vm1:/tmp$ ls
mytar1.tar  systemd-private-c34050aa8824428aae2ce7cfdb2ee666-systemd-logind.service-6p9Km5
snap.lxd    systemd-private-c34050aa8824428aae2ce7cfdb2ee666-systemd-resolved.service-j2NotJ
sub2        systemd-private-c34050aa8824428aae2ce7cfdb2ee666-systemd-timesyncd.service-GsW9ym
user1@vm1:/tmp$
```

คำสั่งขยายไฟล์

แตกทาร์ไฟล์

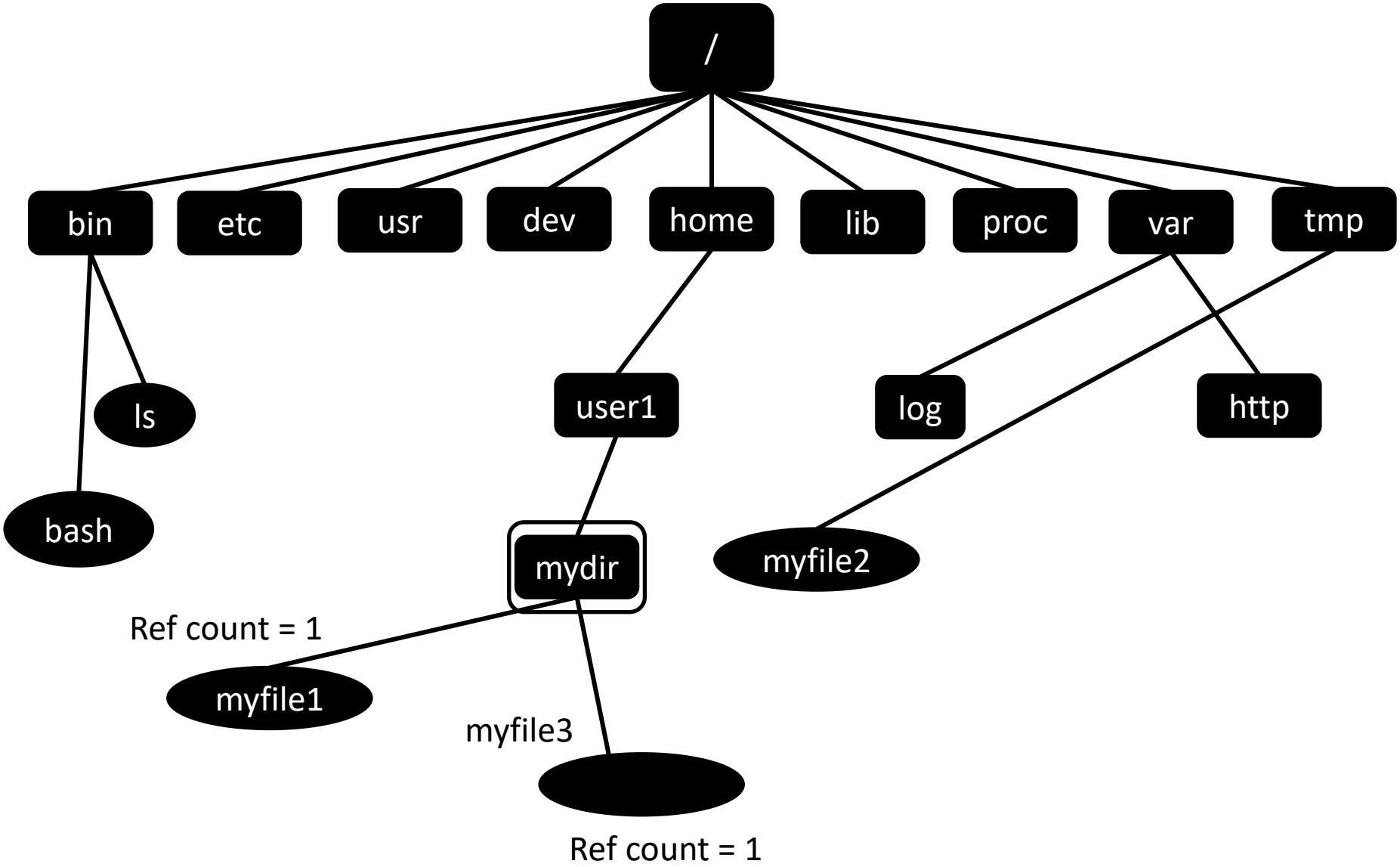
Link

- ไฟล์ Link มีสองแบบได้แก่ hard link และ symbolic link
- Hard link: $ ln <source file> <link name>
- เมื่อสร้างไฟล์ด้วย link จะเกิดการเพิ่ม reference count ให้กับ original file
- link คือการสร้างชื่ออ้างอิงเพิ่ม
- **ไม่ใช่**การคัดลอกเนื้อหาไปยังพื้นที่เก็บข้อมูลใหม่เหมือนคำสั่ง cp
- **กฎของการลบไฟล์คือเนื้อหาของไฟล์จะถูกลบจริงเมื่อ reference count มีค่าเท่ากับ 0**

Hard Link

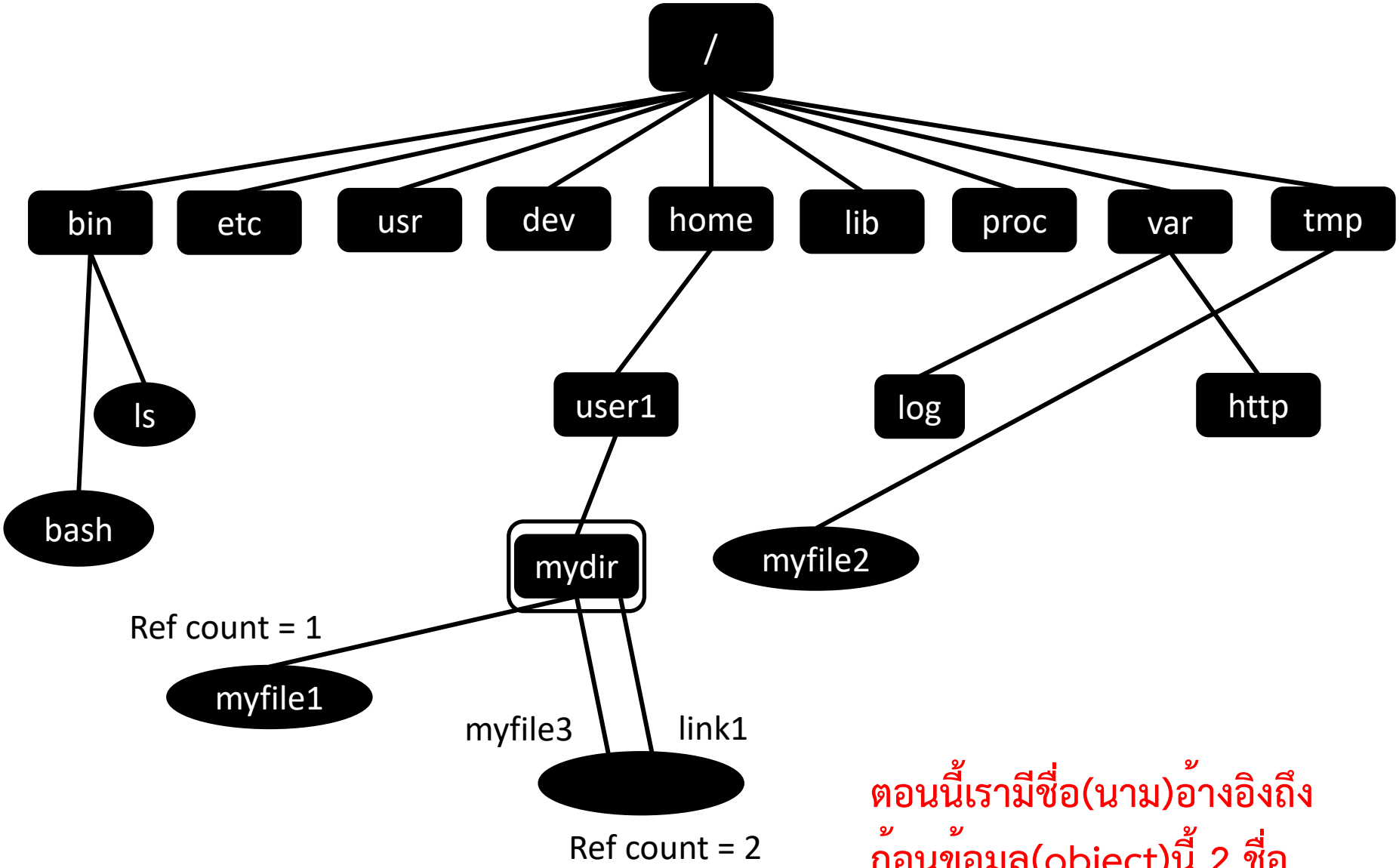
```
user1@vm1:~/mydir$ ls
myfile1  myfile3
user1@vm1:~/mydir$ ls -l
total 4
-rw-rw-r-- 1 user1 user1   0 Aug 25 08:10 myfile1
-rw-r--r-- 1 user1 user1 218 Aug 25 08:09 myfile3
user1@vm1:~/mydir$
```

Reference Count


/
bin
etc
usr
dev
home
lib
proc
var
tmp
ls
bash
user1
log
http
mydir
myfile2
Ref count = 1
myfile1
myfile3
Ref count = 1

```
user1@vm1:~/mydir$ ls
myfile1  myfile3
user1@vm1:~/mydir$ ls -l
total 4
-rw-rw-r-- 1 user1 user1   0 Aug 25 08:10 myfile1
-rw-r--r-- 1 user1 user1 218 Aug 25 08:09 myfile3
user1@vm1:~/mydir$
user1@vm1:~/mydir$ ln myfile3 link1
user1@vm1:~/mydir$ ls -l
total 8
-rw-r--r-- 2 user1 user1 218 Aug 25 08:09 link1
-rw-rw-r-- 1 user1 user1   0 Aug 25 08:10 myfile1
-rw-r--r-- 2 user1 user1 218 Aug 25 08:09 myfile3
user1@vm1:~/mydir$
```

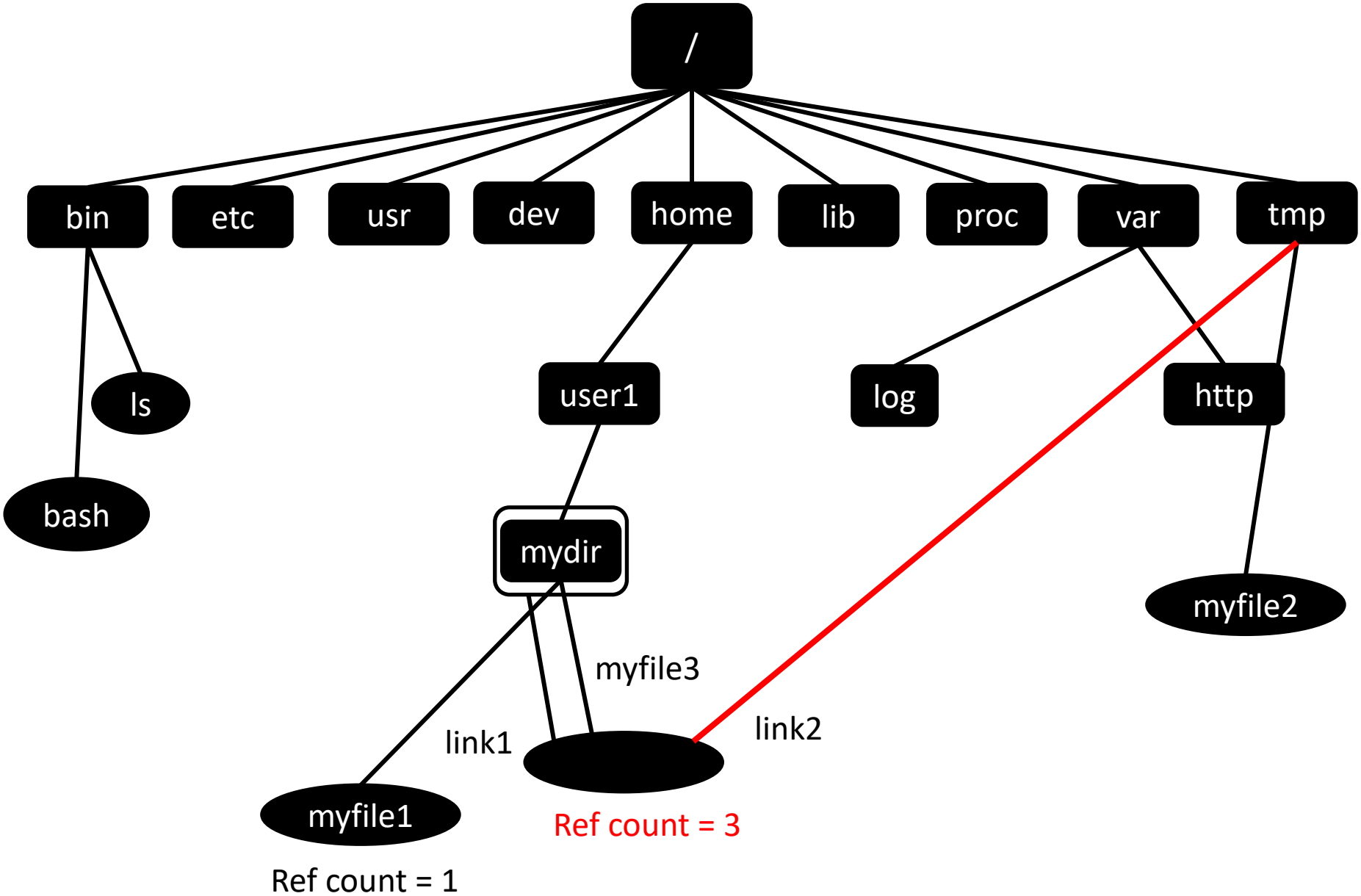
Reference Count



ตอนนี้เรามีชื่อ(นาม)อ้างอิงถึง
ก้อนข้อมูล(object)นี้ 2 ชื่อ
Ref count ของข้อมูลคือ 2

```
user1@vm1:~/mydir$
user1@vm1:~/mydir$ ln myfile3 /tmp/link2
user1@vm1:~/mydir$ ls -l
total 8
-rw-r--r-- 3 user1 user1 218 Aug 25 08:09 link1
-rw-rw-r-- 1 user1 user1   0 Aug 25 08:10 myfile1
-rw-r--r-- 3 user1 user1 218 Aug 25 08:09 myfile3
user1@vm1:~/mydir$ ls -l /tmp/link2
-rw-r--r-- 3 user1 user1 218 Aug 25 08:09 /tmp/link2
user1@vm1:~/mydir$
```

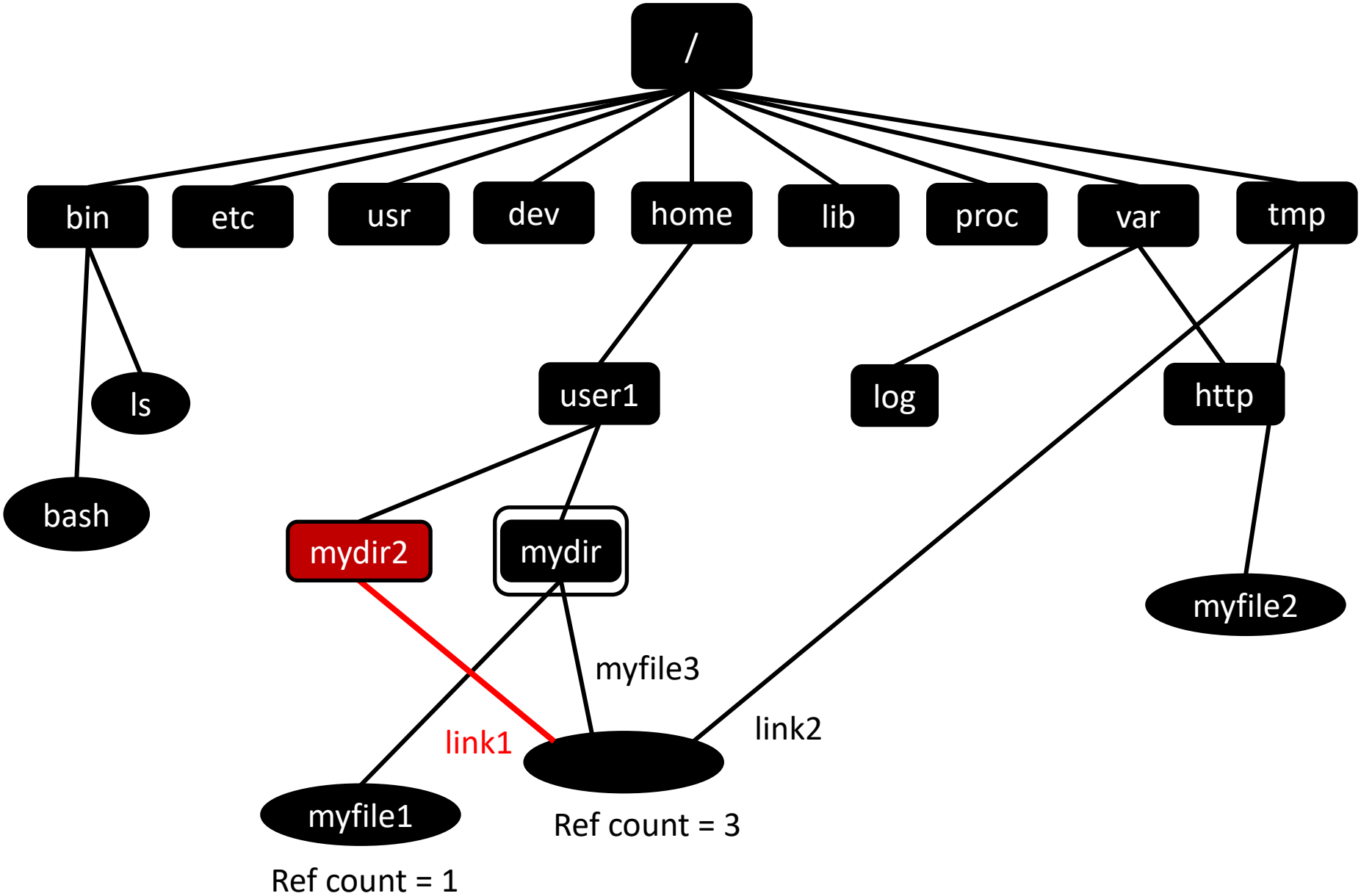
สามารถสร้าง link
บนไดเรกทอรี่อื่นได้


/
bin
etc
usr
dev
home
lib
proc
var
tmp
ls
bash
user1
log
http
mydir
myfile2
myfile3
link1
link2
myfile1
Ref count = 3
Ref count = 1

```
user1@vm1:~/mydir$
user1@vm1:~/mydir$ ln myfile3 /tmp/link2
user1@vm1:~/mydir$ ls -l
total 8
-rw-r--r-- 3 user1 user1 218 Aug 25 08:09 link1
-rw-rw-r-- 1 user1 user1   0 Aug 25 08:10 myfile1
-rw-r--r-- 3 user1 user1 218 Aug 25 08:09 myfile3
user1@vm1:~/mydir$ ls -l /tmp/link2
-rw-r--r-- 3 user1 user1 218 Aug 25 08:09 /tmp/link2
user1@vm1:~/mydir$
user1@vm1:~/mydir$ mkdir ../mydir2
user1@vm1:~/mydir$ mv link1 ../mydir2
user1@vm1:~/mydir$
user1@vm1:~/mydir$ ls -l
total 4
-rw-rw-r-- 1 user1 user1   0 Aug 25 08:10 myfile1
-rw-r--r-- 3 user1 user1 218 Aug 25 08:09 myfile3
user1@vm1:~/mydir$ ls -l ../mydir1
ls: cannot access '../mydir1': No such file or directory
user1@vm1:~/mydir$ ls -l ../mydir2
total 4
-rw-r--r-- 3 user1 user1 218 Aug 25 08:09 link1
user1@vm1:~/mydir$
```

สามารถสร้าง link บนไดเรกทอรี่อื่นได้

ย้าย link ไปยังไดเรกทอรี่อื่นได้

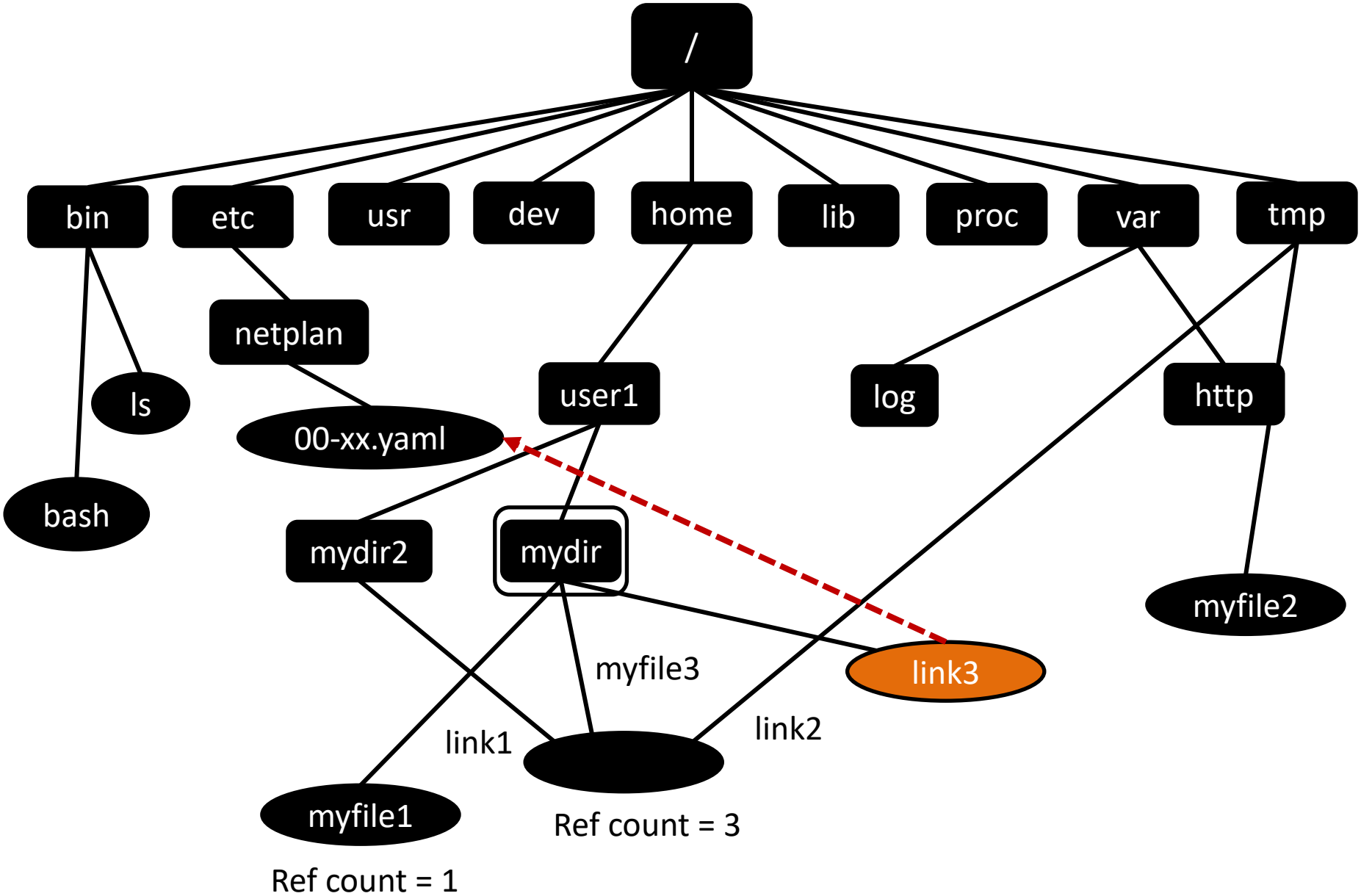

/
bin
etc
usr
dev
home
lib
proc
var
tmp
ls
bash
user1
log
http
mydir2
mydir
myfile2
myfile3
link1
link2
myfile1
Ref count = 3
Ref count = 1

Symbolic Link

- Symbolic link: $ ln -s <source file> <link name>
- เมื่อสร้างไฟล์ด้วย sym link จะเป็นการสร้างไฟล์ใหม่ที่เก็บ absolute pathname ของ source file (เหมือนไฟล์ short cut ใน windows)
- การลบ link คือการลบ short cut **ไม่มีผลต่อ ref count และไม่ได้ลบเนื้อหาจริง**
- ใช้แก้ปัญหาเมื่อผู้ใช้ต้องการสร้าง link กับไฟล์ที่ไม่ใช่เจ้าของ แต่ไม่สามารถทำได้ เช่น link ไปยังไฟล์ของรูท
- เป็นเหมือนพ๊อยเตอร์หรือ shortcut ชี้ไปยังไฟล์ปลายทาง

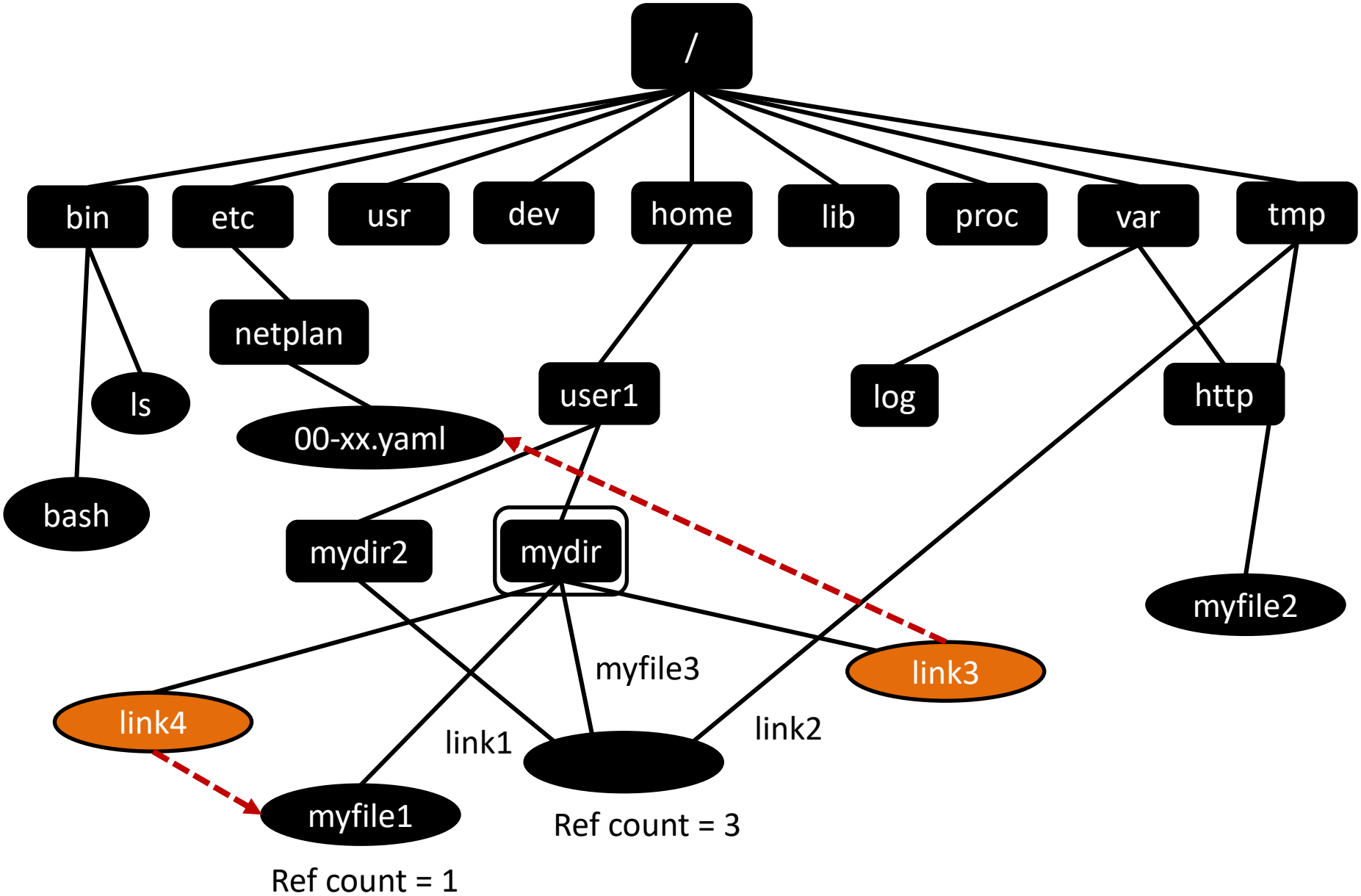
Symbolic Link

```
user1@vm1:~/mydir$
user1@vm1:~/mydir$ ln -s /etc/netplan/00-installer-config.yaml link3
user1@vm1:~/mydir$
user1@vm1:~/mydir$ cat link3
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s9:
      dhcp4: true
  version: 2
user1@vm1:~/mydir$ ls
link3  myfile1  myfile3
```


/
bin
etc
usr
dev
home
lib
proc
var
tmp
netplan
ls
user1
log
http
00-xx.yaml
bash
mydir2
mydir
myfile2
myfile3
link3
link1
link2
myfile1
Ref count = 3
Ref count = 1

Symbolic Link

```
user1@vm1:~/mydir$ ln -s /etc/netplan/00-installer-config.yaml link3
user1@vm1:~/mydir$
user1@vm1:~/mydir$ cat link3
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s9:
      dhcp4: true
  version: 2
user1@vm1:~/mydir$ ls
link3  myfile1  myfile3
user1@vm1:~/mydir$ ln -s myfile1 link4
user1@vm1:~/mydir$ ls -l
total 4
lrwxrwxrwx 1 user1 user1  37 Aug 25 08:27 link3 -> /etc/netplan/00-install
fig.yaml
lrwxrwxrwx 1 user1 user1   7 Aug 25 08:29 link4 -> myfile1
-rw-rw-r-- 1 user1 user1   0 Aug 25 08:10 myfile1
-rw-r--r-- 3 user1 user1 218 Aug 25 08:09 myfile3
user1@vm1:~/mydir$
```


/
bin
etc
usr
dev
home
lib
proc
var
tmp
netplan
ls
00-xx.yaml
user1
log
http
bash
mydir2
mydir
myfile2
link3
myfile3
link4
link1
link2
myfile1
Ref count = 3
Ref count = 1

ตัวแปรสภาพแวดล้อม

- ในการทำงานของโปรแกรมเชล ตัวแปรสภาพแวดล้อมคือ attributes สำหรับเก็บค่าลักษณะของสภาพแวดล้อมที่มีผลต่อการประมวลผลของเชลหรือโปรแกรมที่รันจากเชล (เช่น อุณหภูมิ ฤดู สถานที่ เป็นตัวแปรสภาพแวดล้อมของแต่ละคน)
- เมื่อมีการรันโปรแกรมจากเชล โปรเซสลูกของเชลก็จะได้รับชุดของสภาพแวดล้อมเป็นมรดก
- ใน bash เรียกดูได้ด้วยคำสั่ง set ซึ่งเป็นคำสั่ง built-in command หรือ
- ใช้คำสั่ง env ซึ่งเป็น executable ใน /usr/bin

```
user1@vm1:~$ set | more
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_
basciiranges:histappend:interactive_comments:login_shell
BASH_ALIASES=()
BASH_ARGC=([0]="0")
BASH_ARGV=()
BASH_CMDS=()
BASH_COMPLETION_VERSINFO=([0]="2" [1]="11")
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="5" [1]="1" [2]="4" [3]="1" [4]="rele
BASH_VERSION='5.1.4(1)-release'
COLUMNS=98
DIRSTACK=()
EUID=1001
GROUPS=()
HISTCONTROL=ignoreboth
HISTFILE=/home/user1/.bash_history
HISTFILESIZE=2000
HISTSIZE=1000
HOME=/home/user1
HOSTNAME=vm1
HOSTTYPE=aarch64
IFS=$' \t\n'
LANG=C.UTF-8
LC_CTYPE=C.UTF-8
LESSCLOSE='/usr/bin/lesspipe %s %s'
LESSOPEN='| /usr/bin/lesspipe %s'
LINES=30
--More--
```

ตัวแปรสภาพแวดล้อม

- การกำหนดค่าตัวแปรใช้ V=D โดยที่ V เป็นชื่อตัวแปร และ D คือค่า ส่วนการดึงค่าที่เก็บในตัวแปร V ออกมาใช้ เราใช้วิธีการ dereferencing และโอเปอร์เรเตอร์ของการ dereferencing คือ “$” เช่น $V หมายถึงการดึงค่า D ที่เก็บใน V ออกมาใช้งาน
- ตัวแปรสภาพแวดล้อมที่น่าสนใจ เช่น USER, HOME, SHELL, PATH, PWD, HOSTNAME, HOSTTYPE
- <u>ตัวแปร PATH เป็นตัวแปรที่เก็บลิสต์ของชื่อไดเรกทอรี่ขั้นด้วย “:” ที่โปรแกรมเชลจะเข้าไปค้นหาชื่อที่ผู้ใช้ป้อนให้กับเชลเพื่อเรียกไฟล์ชื่อนั้นขึ้นมาประมวลผล ยกตัวอย่างเช่น ถ้า PATH=/usr/local/bin:/usr/bin:/usr/sbin และผู้ใช้เรียก ไฟล์ abc ขึ้นมาประมวลผล โปรแกรมเชลจะมองหาไฟล์ชื่อ abc ใน /usr/local/bin ก่อนถ้าม่พบจึงจะหาใน /usr/bin และ /usr/sbin ตามลำดับ</u>
- คำสั่ง which ใช้เช็คได้ว่าไฟล์ที่เรียกมาจากไดเรกทอรี่ไหนใน PATH

```
user1@vm1:~$
user1@vm1:~$ echo $USER
user1
user1@vm1:~$ echo $HOME
/home/user1
user1@vm1:~$ echo $SHELL
/bin/bash
user1@vm1:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
/bin:/usr/games:/usr/local/games:/snap/bin
user1@vm1:~$ echo $PWD
/home/user1
user1@vm1:~$ echo $HOSTNAME
vm1
user1@vm1:~$ echo $HOSTTYPE
aarch64
user1@vm1:~$
user1@vm1:~$ which ls
/usr/bin/ls
user1@vm1:~$ which env
/usr/bin/env
user1@vm1:~$ which date
/usr/bin/date
user1@vm1:~$ which ifconfig
user1@vm1:~$ which netplan
/usr/sbin/netplan
user1@vm1:~$
```

ลิสต์ของที่เก็บ executable

แสดงตำแหน่งที่เก็บ
ถ้าไม่พบก็จะไม่แสดงอะไร

ตัวแปร และคำสั่ง export

- การกำหนดค่าตัวแปรใช้ V=D โดยที่ V เป็นชื่อตัวแปร และ D คือค่า
- ปกติแล้วผู้ใช้สามารถกำหนดค่าตัวแปร และ dereference ค่าเพื่อใช้งานได้ตลอด เช่นใน การเขียนโปรแกรมด้วย shell scripts แต่ตัวแปรนั้นจะเป็นตัวแปร local เท่านั้น และไม่เป็นตัวแปรสภาพแวดล้อม
- ผู้ใช้สามารถทำให้ตัวแปร local กลายเป็นตัวแปรสภาพแวดล้อมได้โดยใช้คำสั่ง export ซึ่งเป็นคำสั่ง built-in ของโปรแกรมเชล
- เมื่อตัวแปรเป็นตัวแปรสภาพแวดล้อมแล้ว เมื่อมีการเรียกใช้โปรแกรมอื่นจากเชลค่าสภาพแวดล้อมจะถูกส่งผ่านไปยังโปรแกรมนั้นด้วย
  - เป็นวิธีการผ่านค่าพารามีเตอร์เข้าสู่โปรแกรมอย่างหนึ่ง
  - C โปรแกรมสามารถใช้ function getenv() อ่านค่าได้
- คำสั่ง unset เป็นคำสั่ง built-in ในเชลที่ยกเลิกตัวแปรสภาพแวดล้อม

```
user1@vm1:~$
user1@vm1:~$ VAR1="test v1"
user1@vm1:~$ echo $VAR1
test v1
user1@vm1:~$ bash
user1@vm1:~$
user1@vm1:~$ echo $VAR1

user1@vm1:~$ exit
exit
user1@vm1:~$ echo $VAR1
test v1
user1@vm1:~$ export VAR1
user1@vm1:~$ bash
user1@vm1:~$
user1@vm1:~$ echo $VAR1
test v1
user1@vm1:~$
user1@vm1:~$ exit
exit
user1@vm1:~$ unset VAR1
user1@vm1:~$ echo $VAR1

user1@vm1:~$
```

เมื่อ VAR1 เป็นตัวแปรโลคอล
เมื่อเรียกโปรแกรมเชลใหม่ (ซ้อน)
เชลใหม่จะไม่รู้จัก VAR1

ทำให้ VAR1 เป็น
Environment variable

ยกเลิก VAR1 และเคลียร์ค่า

26/8/64

การเพิ่ม directory ใน $PATH

```
$ cd $HOME
$ mkdir bin
$ echo "echo test scripts" > $HOME/bin/a.sh
$ chmod 755 $HOME/bin/a.sh; mkdir mydir/scripts
$ cp $HOME/bin/a.sh mydir/scripts
$ chmod 755 mydir/scripts/*
$ echo "echo test scripts 2" >> mydir/scripts/a.sh
$ mkdir $HOME/bin
$ PATH=$PATH:$HOME/bin
$ echo $PATH
$ which a.sh
$ a.sh
$ PATH=$HOME/mydir/scripts:$PATH
$ which a.sh
```

ถ้าต้องการลบ $HOME/bin และ $HOME/mydir/bin ออกจาก PATH ต้องทำอย่างไร

การสร้าง Shell script **Part 1**

- ใช้ “nano” เพื่อสร้าง a script file
- บรรทัดแรกต้องใส่ #!/bin/bash
- นศ สามารถใส่คำสั่ง command lines ใน script

```
$ nano hello.sh
#!/bin/bash
echo “Hello world!”
pwd
ls
```

การกำหนดค่า Permission Flag ให้ Shell Script

```
$ cd ; cd mydir
$ ./hello.sh
$ ls -l hello.sh
$ chmod 755 hello.sh
$ ls -l hello.sh
$ ./hello.sh
$ chmod 000 hello.sh
$ ls -l hello*
$ chmod 100 hello.sh
$ ls -l hello*
$ chmod 777 hello.sh
```

What do you see..

What do you see..

What do you see..

What happen..

What does it mean..

การสร้าง Shell script

- ผ่านค่า Parameters จาก CLI
- Working directory ในขณะที่สคริปต์ประมวลผลและหลังประมวลผลเสร็จ
- คำสั่ง sleep

```
$ nano hello2.sh
#!/bin/bash

echo “Hello world” ${1}
echo ${0}
cp /etc/hosts wk${1}.txt
sleep 20
pwd
echo done
```

การรันคำสั่งเป็น background process

- ในกรณีที่โปรแกรมหรือสคริปต์ประมวลผลนาน ผู้ใช้อาจไม่ต้องการรอให้โปรแกรมนั้นประมวลผลเสร็จก่อนที่จะรันคำสั่งอื่น ผู้ใช้สามารถสั่งให้ bash เชลรันโปรแกรมเป็น background process ได้ โดยส่เครื่องหมาย & ข้างหลังคำสั่ง
- ซึ่งจะส่งผลให้โปรแกรมเชลกลับมารอรับคำสั่งใหม่ทันที
- คำสั่ง ps จะแสดงรายการโปรเซสที่รันจากเชลโปรแกรม
- คำสั่ง ps –elf จะแสดงรายการโปรเซสทั้งหมดในระบบ
- คำสั่ง top และ htop จะแสดงรายการโปรเซสทั้งหมดในระบบ

การประมวลผล background process และแสดงสถานะ

```
$ cd $HOME/mydir
$ ./hello2.sh 1
$ ./hello2.sh 2
```

What happen? อธิบายว่าสคริปต์ทำอะไร

```
$ ./hello2.sh 3 &
$ ./hello2.sh 4 &
$ ps
```

What do you see..

```
$ ps -elf | grep hello
```

การสร้าง Shell script 3

```
$ nano hello3.sh
#!/bin/bash

cd $HOME
echo “Hello world” ${1}
echo ${0}
cp /etc/hosts wk${1}.txt
ls -l wk*
pwd
sleep ${2}
```

What happen? จงอธิบายว่าสคริปต์ทำอะไรและผลลัพธ์ต่างจาก hello.sh อย่างไรบ้าง

การดูสถานะและ kill process

- ในกรณีที่โปรแกรมรันนานเกินไป หรือเกิด bug ทำให้ผู้ใช้อยากจะหยุดการประมวลผลของโปรเซสนั้น ผู้ใช้สามารถใช้คำสั่ง kill เพื่อส่ง signal ซึ่งเป็นสัญญานขัดจังหวะการประมวลผลหรืออินเตอร์รับไปให้โปรเซสตามหมายเลขโปรเซสไอดี (pid)
- ผู้ใชสามารถ kill โปรเซสโดยระบุค่าสันญานอินเตอร์รับด้วยคำสั่ง

```
$  kill  -<signal num>  <process id> …
```

- หมายความว่าให้ส่งสันญาณอินเตอร์รับ <signal num> ไปยังโปรเซส <process id> ซึ่งอาจมีมากกว่าหนึ่งโปรเซสก็ได้
- สันญาณอินเตอร์รับ 9 หมายถึง SIGTERM คือให้จบโปรเซส
- ผู้ใช้ kill โปรเซสของตนได้ และผู้ใช้ root และผู้ใช้ที่ใช้ sudo แล้วมีสถานะเป็นเหมือนรูทสามารถ kill ทุกโปรเซส

การดูสถานะและ kill process

```
$ ./hello3.sh 3 5
$ ./hello3.sh 6 500 &
$ ./hello3.sh 7 1000 &
$ ps -elf
$ ps –elf | grep hello
$ kill -9 <process id of hello 6 500>
$ kill -9 <process id of hello 7 1000>
$ sudo su
# ./hello3.sh 8 1000 &
# exit
$ sudo kill -9 <process id of hello 8 1000>
```

การส่งสัญญาณขัดจังหวะด้วยคำสั่ง kill

- ผู้ใช้สามารถแสดงรายการโปรเซสของตนได้ด้วยคำสั่ง

  $ ps -u <user id>

  ยกตัวอย่างเช่น $ ps –u kasidit
- ค่า signal number ที่ใช้กันมากได้แก่
- $ kill -1 คือคำสั่งให้โปรเซส Hangup (โปรเซสสามารถบล็อกได้)
- $ kill -2 คือการขัดจังหวะด้วย ^C (โปรเซสสามารถบล็อกได้)
- $ kill -9 คือคำสั่งให้โปรเซส Terminate (ไม่สามารถบล็อกได้)

สัญญาณ signal ในลินุกซ์

```
$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
$
```

การตรวจจับสัญญาณ signal ของ process

- สร้างสคริปต์ mysignal.sh และใช้คำสั่ง trap เพื่อดักจับสัญญาน signal หมายเลขที่กำหนด ซึ่งเมื่อได้รับจะทำ action คือรันคำสั่งใน ‘echo …’ พารามิเตอร์ที่สองของคำสั่งนี้คือหมายเลย signal ที่ต้องการจะจับ
- Signal หมายเลข 9 ไม่สามารถอินเตอร์เซบได้

```
$ nano mysignal.sh
#!/bin/bash
trap 'echo I got Sig 1' 1
trap 'echo I got Sig 2' 2
while true
do
   echo "do something"
   sleep 1
done
```

การตรวจจับสัญญาณ signal ของ process

- สร้างสคริปต์ mysignal.sh และรันบน terminal ที่ 1 เป็น forground
- และเปิด terminal ที่ 2 เพื่อออกคำสั่ง kill -<num ber>

```
$ ./mysignal.sh
do something
^C
I got Sig 2
Do some ...
I got Sig 1
do something
$
```

```
$ ps –u user1 | grep my
1234
$ kill -1  1234




$ kill -9 1234
```

การสร้าง Shell script 3

```
$ nano hello3.sh
#!/bin/bash

cd $HOME
echo “Hello world” ${1}
echo ${0}
cp /etc/hosts wk${1}.txt
ls -l wk*
pwd
sleep ${2}
```

What happen? จงอธิบายว่าสคริปต์ทำอะไรและผลลัพธ์ต่างจาก hello.sh อย่างไรบ้าง

สร้าง shell script แบบให้ป้อนค่า input variables

- ตัวแปร first และ last เป็นตัวแปร shell variables
- คำสั่ง read เป็นคำสั่งอ่านค่าสตริงที่ผู้ใช้ป้อนเข้าสู่ตัวแปร -p เป็นการแสดงข้อความก่อนที่จะอ่านค่าตัวแปรจาก CLI
- $first และ $last เป็นการดึงค่าที่เก็บในตัวแปรมาใช้

```
$ nano printname.sh
#! /bin/sh
read -p "enter your name: " first last
echo "First name: $first"
echo "Last name: $last"
```

here scripts

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
$ cat bin/b.sh
#!/bin/bash
# example of here script

echo "<html>
<head>abcd</head>
<body>
hello
</body>
</html>"
$
```

```
$ cat bin/c.sh
#!/bin/bash
# example of here script
cat << _EOF_
<html>
<head>abcd</head>
<body>
hello
</body>
</html>
_EOF_
$
```

ตัวอย่างคำสั่ง alias และ function

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
$ alias l=‘ls –la’
$ l
...
$ today() {
    echo -n "Today's date is: "
    date +"%A, %B %-d, %Y"
}
$ today
...
$
```

Control flow: if

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
if commands; then
    commands
[elif commands; then
    commands...]
[else
    commands]
fi
```

```
kasidit@flyvm:~/mydir$ true
kasidit@flyvm:~/mydir$ echo $?
0
kasidit@flyvm:~/mydir$ false
kasidit@flyvm:~/mydir$ echo $?
1
kasidit@flyvm:~/mydir$
```

```
kasidit@flyvm:~/mydir$ echo $?
0
kasidit@flyvm:~/mydir$ ls x.sh
ls: cannot access 'x.sh': No such file or directory
kasidit@flyvm:~/mydir$ echo $?
2
kasidit@flyvm:~/mydir$
```

exit status

Control flow: for

```
#!/bin/bash

for i in 1 2 3 4 ; do
      echo $i
done
```

```
kasidit@koji:~$ ./myprog.sh
1
2
3
4
kasidit@koji:~$
```

```
#!/bin/bash

for i in $(seq 8) ; do
      echo $i
done
```

```
kasidit@koji:~$ ./myprog2.sh
1
2
3
4
5
6
7
8
kasidit@koji:~$
```

Control flow: for

```
#!/bin/bash

for i in $(seq 8 2 18) ; do
      echo $i
done
```

```
kasidit@koji:~$ ./myprog3.sh
8
10
12
14
16
18
kasidit@koji:~$
```

Control flow: while, until

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
#!/bin/bash

number=0
while [ "$number" -lt 10 ]; do
    echo "Number = $number"
    number=$((number + 1))
done
```

```
#!/bin/bash

number=0
until [ "$number" -ge 10 ]; do
    echo "Number = $number"
    number=$((number + 1))
done
```

Arithematic expansion

Control flow: test

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
# First form

test expression

# Second form

[ expression ]
```

```
if [ -f .bash_profile ]; then
    echo "You have a .bash_profile. Things are fine."
else
    echo "Yikes! You have no .bash_profile!"
fi
```

Control flow: if

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

Expression	**Description**
-d *file*	True if *file* is a directory.
-e *file*	True if *file* exists.
-f *file*	True if *file* exists and is a regular file.
-L *file*	True if *file* is a symbolic link.
-r *file*	True if *file* is a file readable by you.
-w *file*	True if *file* is a file writable by you.
-x *file*	True if *file* is a file executable by you.
file1 -nt *file2*	True if *file1* is newer than (according to modification time) *file2*.
file1 -ot *file2*	True if *file1* is older than *file2*.
-z *string*	True if *string* is empty.
-n *string*	True if *string* is not empty.
string1 = *string2*	True if *string1* equals *string2.*
string1 != *string2*	True if *string1* does not equal *string2.*

Control flow: test superuser status

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
if [ "$(id -u)" = "0" ]; then
    echo "superuser"
fi
```

```
if [ "$(id -u)" != "0" ]; then
    echo "You must be the superuser to run this script" >&2
    exit 1
fi
```

Control flow: test superuser status

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
function home_space {
    # Only the superuser can get this information

    if [ "$(id -u)" = "0" ]; then
        echo "<h2>Home directory space by user</h2>"
        echo "<pre>"
        echo "Bytes Directory"
            du -s /home/* | sort -nr
        echo "</pre>"
    fi

}   # end of home_space
```

ตัวอย่างการใช้งาน function

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
#!/bin/bash

# sysinfo_page - A script to produce a system information HTML file

##### Constants

TITLE="System Information for $HOSTNAME"
RIGHT_NOW="$(date +"%x %r %Z")"
TIME_STAMP="Updated on $RIGHT_NOW by $USER"

##### Functions

system_info()
{
    echo "<h2>System release info</h2>"
    echo "<p>Function not yet implemented</p>"

}   # end of system_info


show_uptime()
{
    echo "<h2>System uptime</h2>"
    echo "<pre>"
    uptime
    echo "</pre>"

}   # end of show_uptime
```

ตัวอย่างการใช้งาน function

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
drive_space()
{
    echo "<h2>Filesystem space</h2>"
    echo "<pre>"
    df
    echo "</pre>"

}   # end of drive_space


home_space()
{
    # Only the superuser can get this information

    if [ "$(id -u)" = "0" ]; then
        echo "<h2>Home directory space by user</h2>"
        echo "<pre>"
        echo "Bytes Directory"
        du -s /home/* | sort -nr
        echo "</pre>"
    fi

}   # end of home_space
```

ตัวอย่างการใช้งาน function

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
##### Main

cat <<- _EOF_
  <html>
  <head>
      <title>$TITLE</title>
  </head>
  <body>
      <h1>$TITLE</h1>
      <p>$TIME_STAMP</p>
      $(system_info)
      $(show_uptime)
      $(drive_space)
      $(home_space)
  </body>
  </html>
_EOF_
```

Control flow: case

อ้างอิงจาก https://linuxcommand.org/lc3_writing_shell_scripts.php

```
case word in
    patterns ) commands ;;
esac
```

```
#!/bin/bash

read -p "Enter a number between 1 and 3 inclusive > " character
case $character in
    1 ) echo "You entered one."
        ;;
    2 ) echo "You entered two."
        ;;
    3 ) echo "You entered three."
        ;;
    * ) echo "You did not enter a number between 1 and 3."
esac
```

การพัฒนาโปรแกรมภาษา C

- ในการพัฒนาโปรแกรมขนาดใหญ่ โปรแกรมเมอร์มักจะสร้างไฟล์หลายไฟล์ และคอมไพล์ไฟล์เหล่านั้นแยกกัน เนื่องจากไฟล์แต่ละไฟล์มีโค๊ดที่เกี่ยวข้องกับเรื่องใดเรื่องหนึ่งโดยเฉพาะ
- นอกจากนั้น ในกรณีที่มีผู้พัฒนาโปรแกรมหลายคนแต่ละคนอาจพัฒนาไฟล์แยกกัน
- แต่เมื่อจะสร้าง executable file จะต้องนำไฟล์เหล่านั้นมา link ร่วมกัน
- วิธีการพื้นฐานในการพัฒนาโปรแกรมภาษาซีอย่างหนึ่งคือการใช้ make
- ในการคอมไพล์โปรแกรมเช่น qemu-kvm บนลินุกซ์ใช้วิธีนี้ร่วมกับ tool อื่น

ติดตั้ง C compiler และ compile C program

```
$ sudo apt install gcc make
(or apt install build-essentials)
$ nano hello.c
#include <stdio.h>
main(){
  printf(“hello world :)\n”);
}
$ gcc hello.c
$ ./a.out
$ gcc –o hello hello.c
$ ./hello
```

การคอมไพล์ object code และ link โปรแกรม (1)

```
$ mkdir myprograms
$ cd myprograms
$ nano hellosub1.c
#include <stdio.h>
#include “hello1.h”
void fun1(void){
  printf("hello1 %d\n”, CONSTANT1);
}
$ gcc –c hellosub1.c
```

การคอมไพล์ object code และ link โปรแกรม (1)

```
$ nano hello1.h
$ cat hello1.h
#define CONSTANT1 100
$
```

การคอมไพล์ object code และ link โปรแกรม (2)

```
$ nano hellosub2.c
#include <stdio.h>

void fun2(void){
  printf("hello2\n");
}
$ gcc -c hellosub2.c
$ ls -l
```

การคอมไพล์ object code และ link โปรแกรม (3)

```
$ nano hellomain.c
#include <stdio.h>
extern void fun1(void);
extern void fun2(void);

int main(){
  fun1();
  fun2();
}
$ gcc -c hellomain.c
$ ls -l
```

function call prototypes

การคอมไพล์ object code และ link โปรแกรม (4)

```
$ gcc -o hello hellomain.o \
> hellosub1.o hellosub2.o
$ ls -l
$ file *
$ ./hello
```

สรุป

- แนะนำระบบลินุกซ์ และการใช้งานคอมมานด์ไลน์เชลเพื่อออกคำสั่ง
- อธิบายโครงสร้างระบบไฟล์
- อธิบาย Permission Flags สำหรับไฟล์
- อธิบายการใช้งานคำสั่งของระบบลินุกซ์ที่จำเป็น
- อธิบายการพัฒนาโปรแกรมภาษาซีบนระบบลินุกซ์