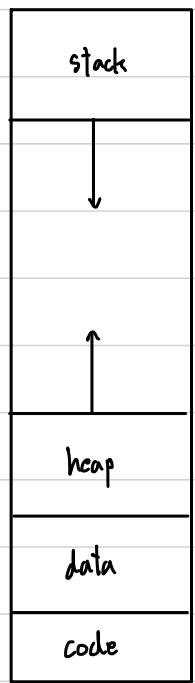


# Chapter 8 Memory Management

## Process Memory Layout

ໃນ 1 process ຈະມີການເກີນຫຸ້ນຮົດຕົວທີ່ (General)



- 1) **code** ໂດຍໄດ້ເປັນ executable program ໃລ້ວ run ໄດ້ sequential
- 2) **data** ຫຸ້ນລະ / ຕົວຢັງທີ່ຈະມາຄວາມເຂົ້າແຂ້ງ ໃນ constant, global var
- 3) **heap** ຮັບແປກທີ່ໄມ້ກູ່ນາຄວາມເກົ່ານ ມາກໍຕອນ run
- 4) **stack** ເກີນ return address

ໄຕລະ: OS ດັ່ງນີ້ແຍະເປັນສິ້ນທີ່ໄມ້ເຫັນດີກັນ

Segmentation fault: ນີ້ memory ທີ່ໄດ້ຈຳກັດ

Stack Overflow: stack ໄດ້ມາຮັບເປັນ heap

```
#include <math.h> ← code
int global; → data
int f(int farg)
{
    int flocal; ← stack → heap
}

main()
{
    int local; ← heap
    int *dynamic; ← dynamic
    dynamic = malloc(10, sizeof(int));
    f(1); ← stack ຫຼື return address
    local = sqrt(2.0);
    ↑ stack
}
```

## Logical vs Physical Address Space

Logical Address (Virtual Address) - ເລີນອອກຕາມແນ່ງຕົວເປົ້າ: CPU ໄກສອນໃຫ້ມາຮັບອຳນວຍຕົວເປົ້າໃນ program

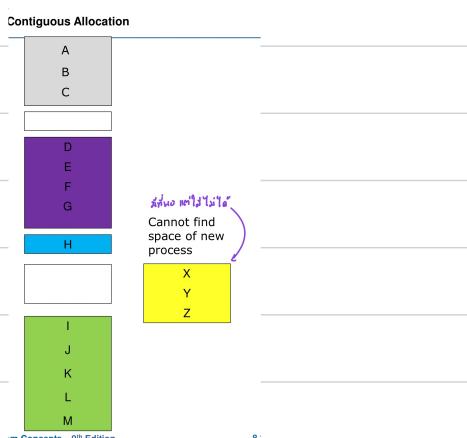
Physical Address - ກົດໝູ້ໃນ RAM ທີ່ຈີ,

Memory Management Unit (MMU) - ມີກຳທີ່ມາຮັບ Logical Address → Physical Address

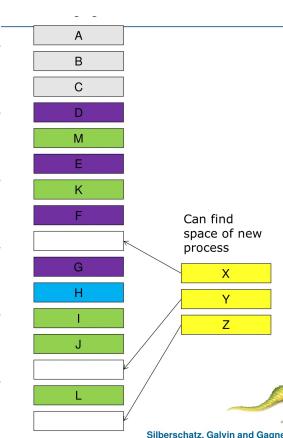
Address Space - ອຳນວຍຕົວເປົ້າໃນ process ມັງ

## Memory Allocation

### 1) Contiguous Allocation



### 2) Paging



ເຕັມໄດ້ບັນຫຼັກ

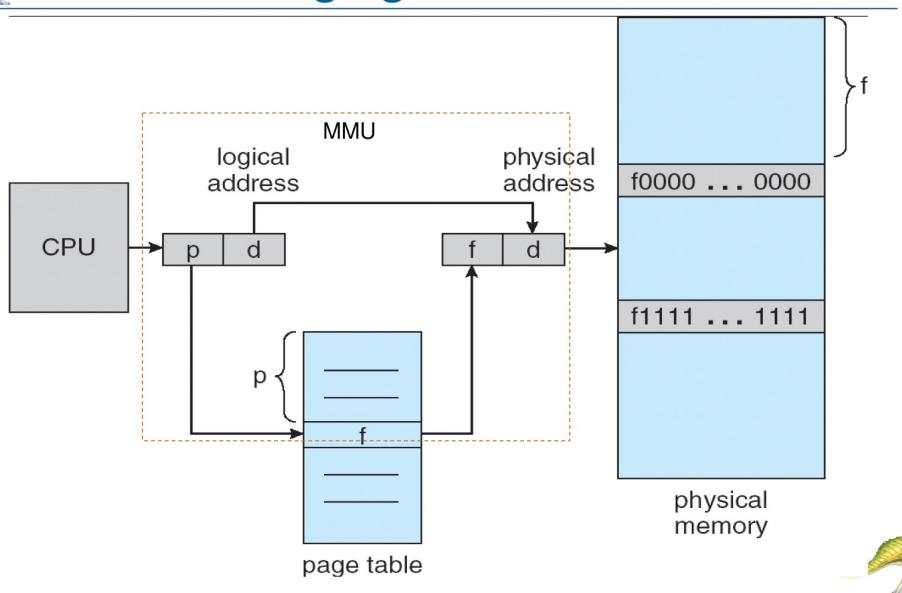
## Paging

1) Physical Memory - ແມ່ນຢືນກວ່າງທີ່ທາດຄວ່າ ເຊິ່ງກ່າວ **frame**

2) Logical Memory - ແມ່ນຢືນກວ່າງທີ່ທາດເທົກ່ານ ເຊິ່ງກ່າວ **page**

OS ດະຍຸດວ່າມີກ່າວໃນ ເຊິ່ງກ່າວ **free frame**

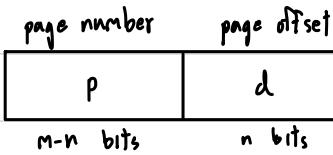
ໃຊ້ **Page table** ອຸກການປັບອານ **logical**  $\rightarrow$  **Physical address** (map **ຈານ page**  $\rightarrow$  **frame**)



ຕົກລະ: process 0 ຂັ້ນ  
page table ພິມຂອງຕົກລະ

## Logical Address ໄປກຳນົດ

1) Page Number (p)



: ຕັ້ງທີ່ໃນ 1 page ກົບຂອງໄວ້  $2^n$  bytes (Page Size) ລວມຈຳກວດ

2) Page Offset (d)

page ລົມນົດ  $2^{m-n}$  ຫຼືໃຈ logical address space  $2^m$  ດີ 16

## Example

Address Width  $20 + 12 = 32$  bits

Page Size  $2^{12} = 4K$  bytes

Offset 12 bits

Page Number 20 bits

## Translation Look-Aside Buffer

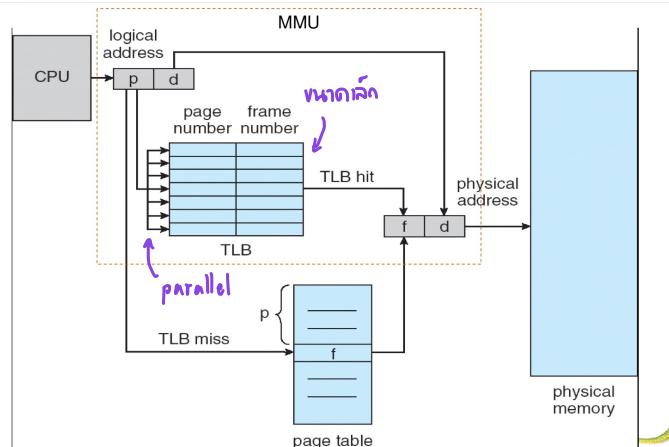
ໂຄສະນາປັ້ງນາດີນີ້ Two memory access problem

ດ້ວຍ page table ໃຊ້ອານໃນ memory

: ເວັບໄດ້ access memory  $2^m$  (ນົບວິທີ)

ຕ້ອງນິ້ນຫຼັກໃນ cache ໃຊ້ອານ TLB

1 process 1 TLB



## Effective Access Time

Associative Lookup =  $\varepsilon$  time unit

Hit ratio =  $\alpha$

Memory access time =  $M$  time unit

$$\therefore \text{EAT} = (\text{hit})\alpha + (\text{miss})(1-\alpha)$$

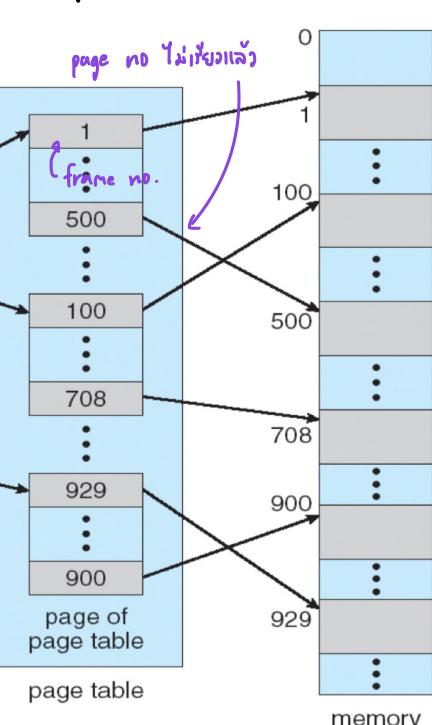
$$= (2-\alpha)M + \varepsilon$$

### Scenario

$\varepsilon$  lookup  $\rightarrow$   $\alpha$  hit  $\rightarrow$   $M$  physical memory

$\varepsilon$  lookup  $\rightarrow$   $1-\alpha$  miss  $\rightarrow$   $M$  page table  $\rightarrow$   $M$  physical memory

## Hierarchical Page Table



### Outer page Table

- 9<sup>th</sup> map all logical address space

- 11 bits entry  $\Rightarrow$  map 1120 pages via page table

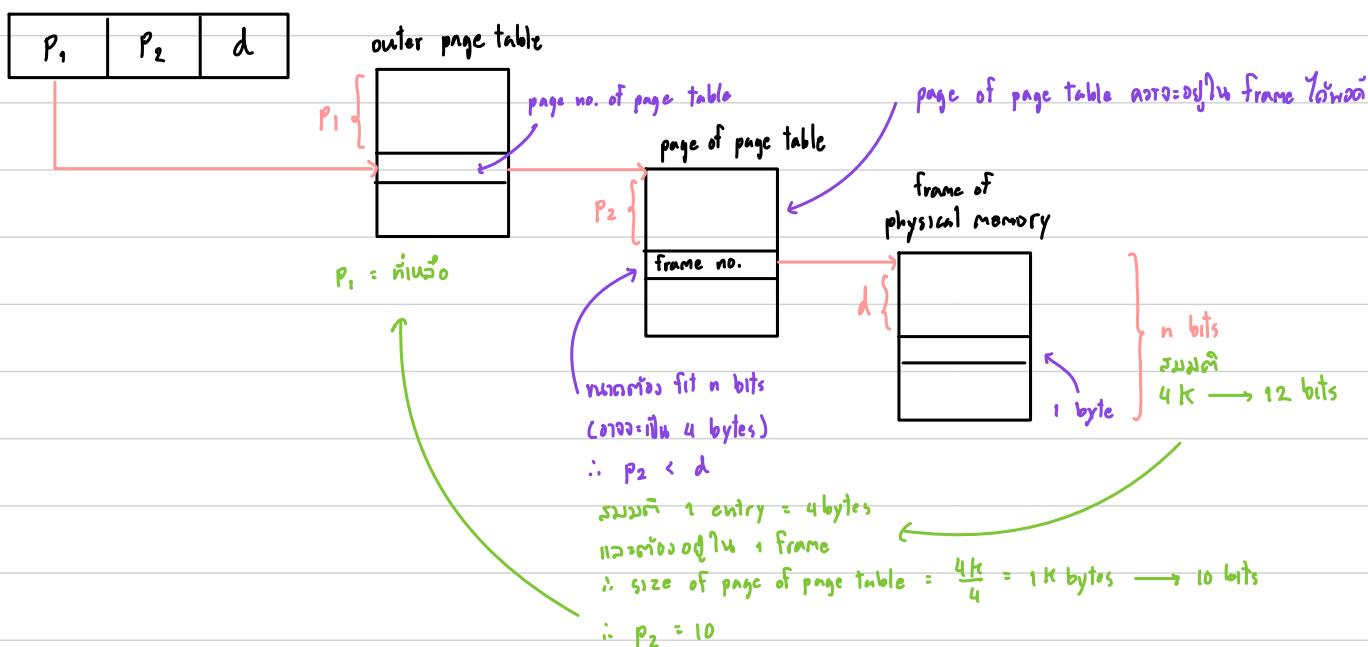
### Page table

- เก็บไว้: pages ที่กำลังอยู่ (dynamic allocated)

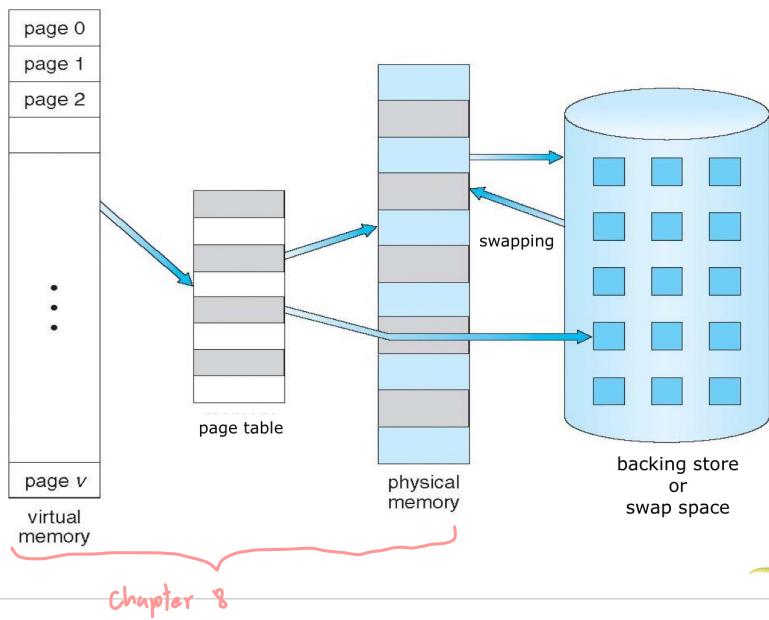
### 1 Process 1 Outer Space

ไม่เกิน 1 เส้นแล้ว OK!

## Logical address



# Chapter 9 Virtual Memory



## Demand Paging

- invalid reference → abort
- not-in-memory → bring to memory
- no-free-frame → initialize swap space

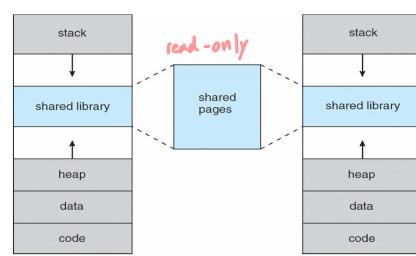
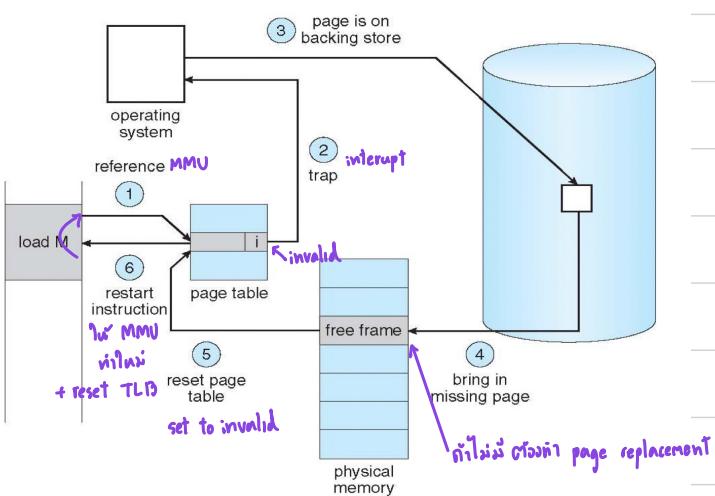
in page table នឹងតាមរឿង valid-invalid bit



invalid នៅក្នុង swap space នូវមួយនាយក

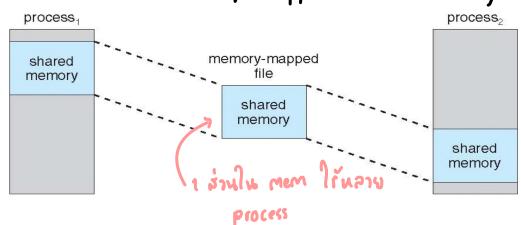
## Page Fault

- ស្ថើ invalid នៃ page table ទូទៅ page fault interrupt នៃ DS ដើម្បី page fault handler

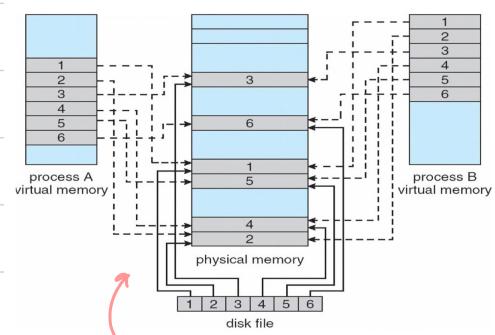


Shared library  
using Virtual  
Memory

## Memory-mapped shared memory



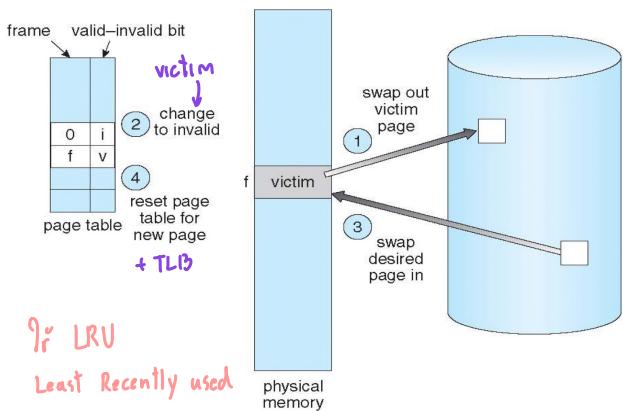
## memory-mapped file



តាមដំឡើងរក្សា file ត្រូវបានបញ្ជូន  
ឬ file បានបិន block នៃវគ្គភ័យ Memory

## Page Replacement

ក្នុងទំនាក់ទំនង page មួយនៃ mem នូវ



នូវ LRU  
Least Recently used

# Chapter 11 File-System Interface

File - โครงสร้าง ต้องที่เก็บข้อมูล ( มองเป็นสมุด ) เป็น series ของ binary มีห้องกัน  
มีองค์ประกอบเรียงกัน Attributes ต่อไป tag file → แบ่งได้成 block device หรือ character device  
↑  
ห้องสำหรับบล็อก  
↑  
ห้องสำหรับ series

File System - ต้องขยายพื้นที่ของจัดการ file อย่างมีประสิทธิภาพ จัดเก็บไฟล์อยู่ใน จดจำ / ปิดยังไง จะเก็บอยู่ไหน

File Operation	1) Create	5) Delete
ptr = pointer {	2) Write → ห้องสำหรับ write pointer	6) Truncate → กรณี / เล่ม / s/o
	3) Read → ห้องสำหรับ read pointer	7) Open → ให้ pointer รู้ไป file แล้วจึง pointer
	4) Seek → ขยับตำแหน่งของ pointer	8) Close → คืน pointer + ปั๊บลง disk ตัวเอง

Open File Locking → concept คล้าย semaphore

Shared lock - ระหว่าง process ที่ต้อง lock พื้นที่เดียวกันได้ ( คล้าย reader lock )

Exclusive lock - แค่ต่อ 1 process ( คล้าย writer lock )

File Allocation Table  
(FAT)

File Extension เกิดต้นจากบน DOS ( รุ่นเดียวทั่วโลก กด F1 หามสกุลของไฟล์ ) ทุนตั้งที่ UNIX ตั้งตัวเองไว้

File Structure คือโครงสร้างไฟล์ ของ OS เดียวๆ กัน

IBM ที่มี built-in DB filesystem ของว่ากัน database  
แบบ ( ชื่อ index ) ที่ method ISAM

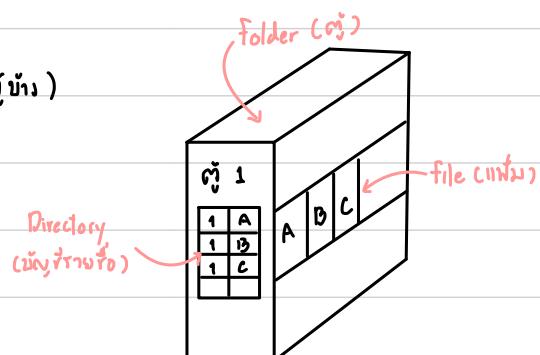
## Directory Structure

Directory คือสมุดรายร่อง / รายการ map รายการ → รายการ ( ดูต้องที่ในร่อง )

Folder คือตู้เก็บเอกสารของไฟล์ ( files ) รวบรวมกัน

File คือไฟล์ที่ตั้งร่อง

not Disk แต่ concept ของ directory , folder หรือ ?



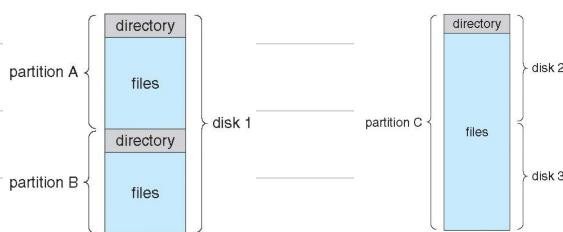
## Disk Structure

partition disk ทุก個จะมี

สอง เรียงกัน partition

ไม่มี partition ทางด้าน

File System ตกลงแบบ



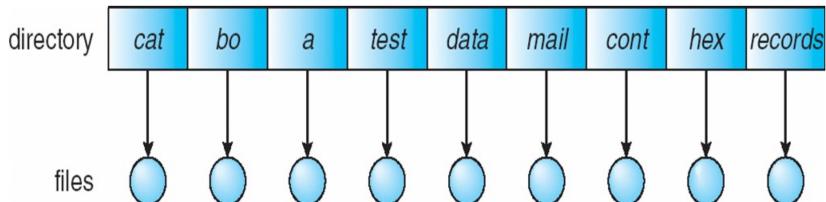
process

## Type of File System

ເພື່ອຕົວ procfs ຕົວ FS ຖະ /proc

ໄນ້ກັບສະນິກຳ general-purposed file system ຫຼຸດ ufs, zfs, NTFS, ext, MPFS, AppleFS

## Single-Level Directory



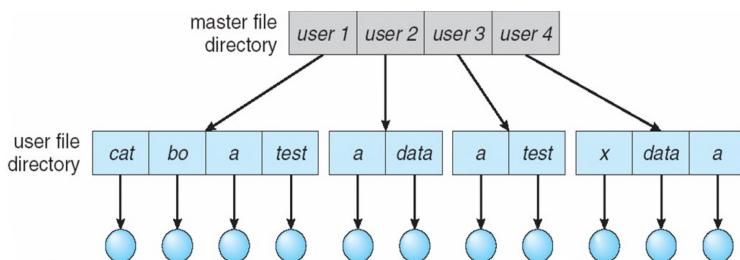
### Naming Problem

ໄດ້ directory ຂໍາໃນໂດຍ

### Grouping Problem

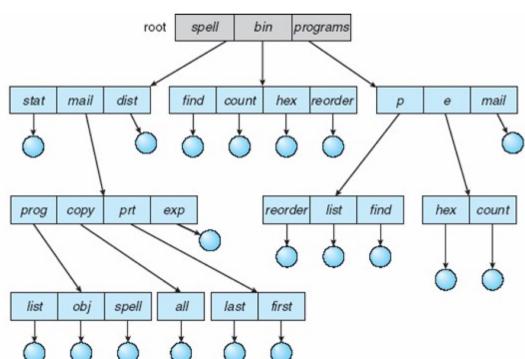
ຈັດກຸ່ມຝຶກສໍາລັບໄດ້

## Two-level Directory



## Tree-Structured Directories

ຈະເກີບເພົ່າ disk ຜົນ ?



rm

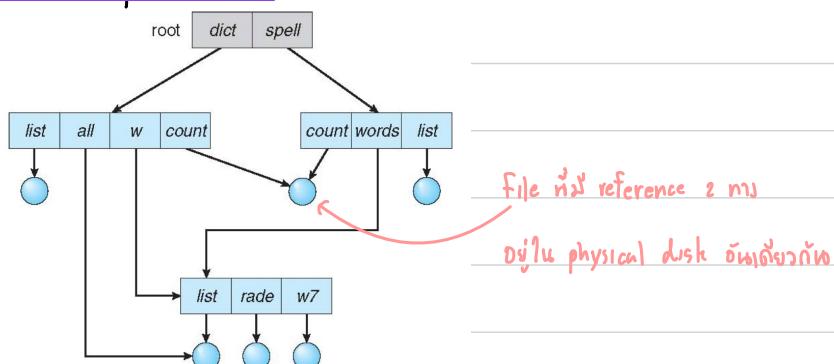
ລົບໄກ້

mkdir

ສ້າງ directory

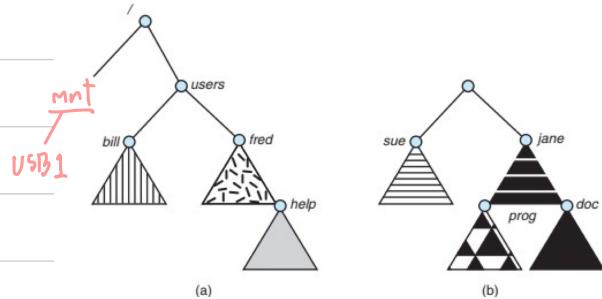
{ ຢັບໃຫຍ່ໂຄງຮຽງຕົວໆໄຟ

## Acyclic-Group Directories



## File System Mounting

mirror volume ស្រីបិនុយ directory



រំលែកថ្លែង disk ទៅកូល mount

ជំនាញគម្រោង Single root - UNIX

ជំនាញគម្រោង Multiple root - Windows

drive C, D, E

: រាយការណ៍ map drive now mount → ស្រាវជ្រាវ ឱ្យចូល

\* នូវឯកសារ drive A, B ឲ្យ floppy disk

## File Sharing

ដំឡើងការកំណត់ឯកសារជាផ្លូវការ

### Protection

owner - ផ្ទើរឃាត

Read - R

group - រួម្បារឃាត

Write - W (append, delete)

Execute - X (view folder = browse (ls) ឱ្យរួច)

Append

Delete

List

} window

	mode	RWX
owner access (U)	7	111
group access (G)	6	110
public access (O)	1	001

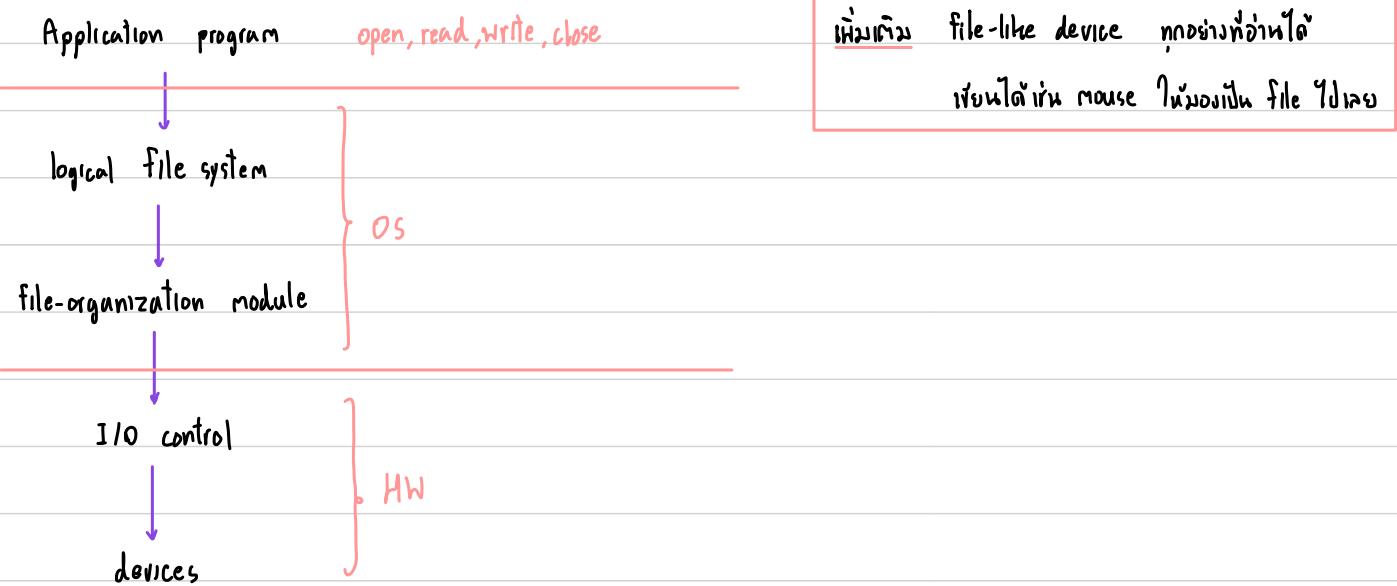
? chmod <mmm> <path> ឱ្យស្នើសុំ mode

chgrp <group-name> <path> ឱ្យស្នើសុំ group

chown <owner-name> <path> ឱ្យស្នើសុំ owner

# Chapter 12 FS Implementation

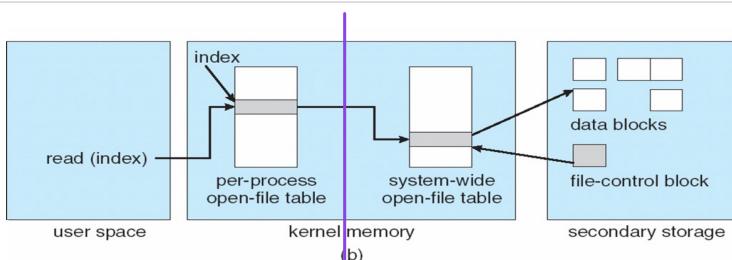
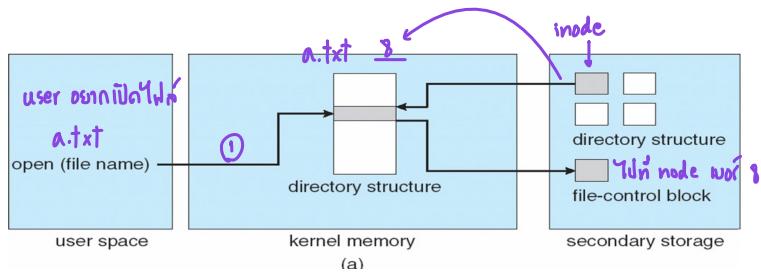
ໃຫຍ່ UNIX ພົມໄບນ໌



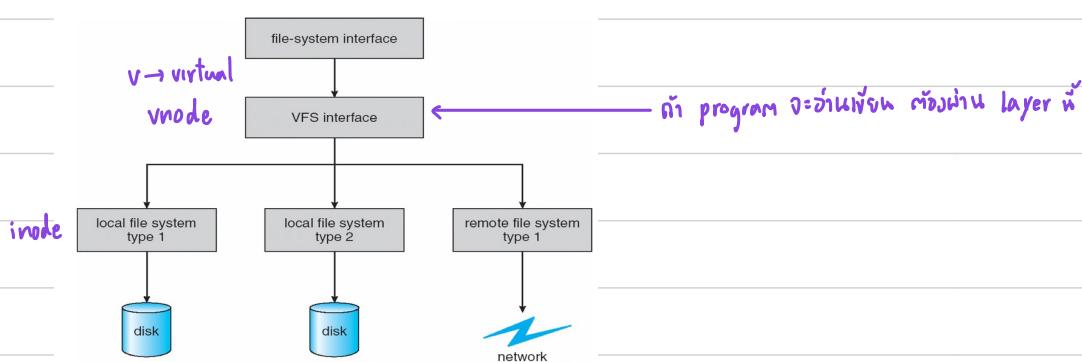
ເພີ້ມເຕີມ file-like device ຖກອຍງ໌ກໍອ່ານໄວ້

ເສັນໄດ້ໃໝ່ mouse ໃຫ້ນອັນປິນ file ອຳຈາຍ

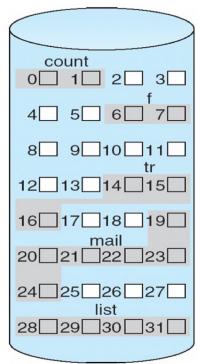
**Inode** (Index node) ເທົ່າວັນໂລກທີ່ກ່ຽວຂ້ອງ, ແລ້ວ file ອັງຢູ່ໃນ disk (ກະລະເໜີຄອງກົງໝັ້ນ)



## Virtual File System



## Contiguous Allocation



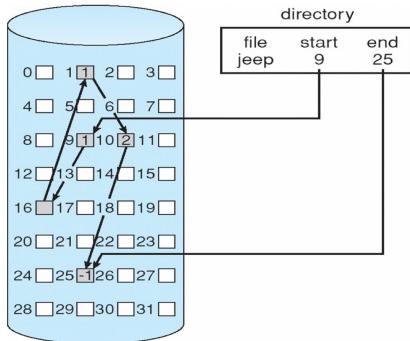
directory		
file	start	length
count	0	2
4	14	3
5	19	6
6	28	4
7	6	2
tr		
mail		
list		
f		

ข้อดี : ง่าย รวดเร็ว

ข้อเสีย : เกิด fragmentation (การตัดส่วน)

└─> แก้ไขโดยการ defragmentation (จัดเรียงไฟล์)

## Linked Allocation

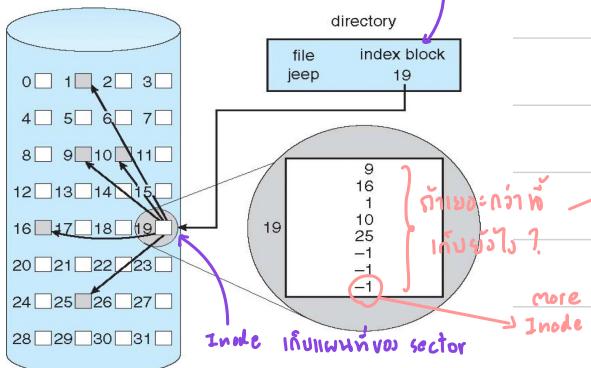


ข้อดี : ก้อน数据มี pointer ที่ดู sector ต่อไป

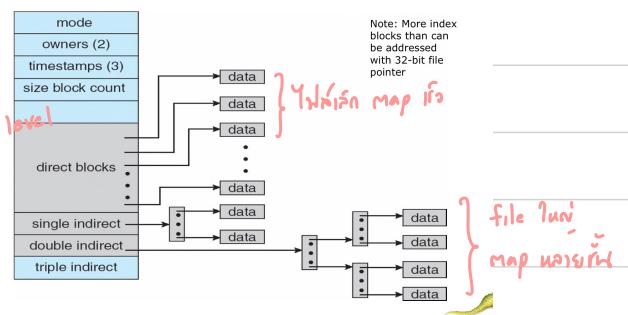
ข้อเสีย : ใช้พื้นที่มาก

จุดเด่น : ต่ำ

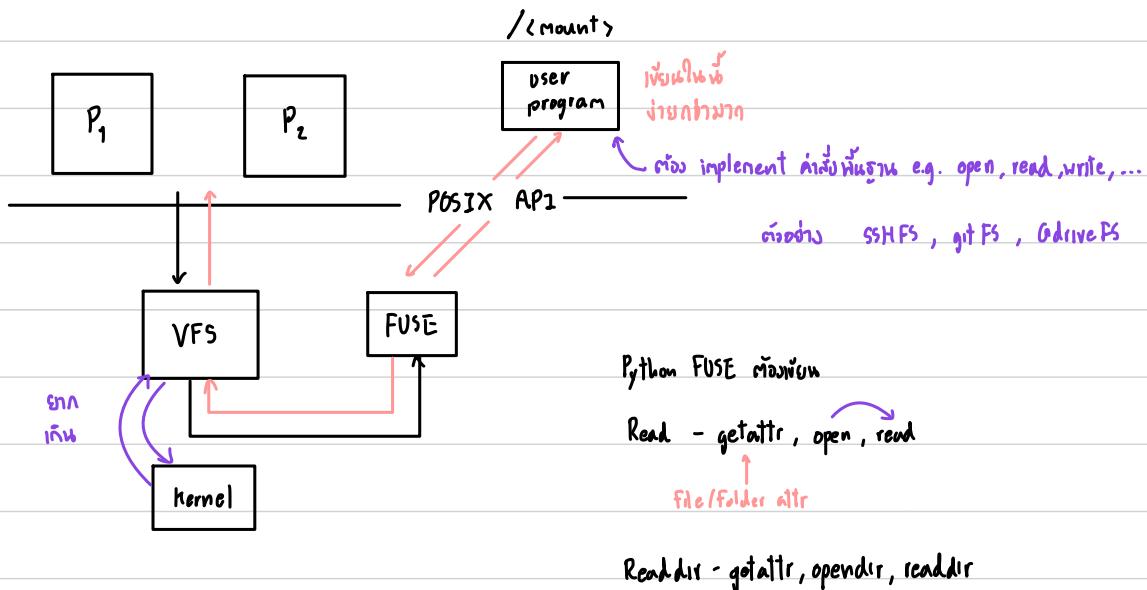
## Indexed Allocation



ข้อดี : ไม่มี fragmentation และ no fragment



# Chapter 13 FUSE



# Chapter 14 Kernel

**Kernel** ນັ້ນ, ນີ້ດີຈຳກັງນອງ OS ໃປ່ນສ່ວນໄຮກຖຸນ load ໂລະ ໃປ່ນສ່ວນຮູດທີ່ບໍ່ unload ຮຽມປິດຕາວົງ (ນັ້ນ driver ຫຼື Windows)

"Resource" Manager

Storage, Memory, Thread, CPU → Process

**Kernel Module** ສູ່ໂປຣມ ອີ່ kernel ດ້ວຍຕົກລົງ/ຄາຍໄດ້



command `insmod` - ມີນ kernel module

`rmmod` - ຜົນ kernel module

`dmesg` - ອີ່ log ຫຼື kernel

**Device Driver** ພິບຮັກທີ່ໃຫ້ OS ຈຶນ device → ອີ່ linux ຂີ່ມີລູນ file-like device

mouse  
1) Character Device - ຢ່ວງຂ່ານຕາມລ່ວດປັບ ທັນໄມ້ໄດ້ (seek ຍົກໄວ້)

2) Block Device - ວ່ານໄຟແກ້ວຂັ້ນທຸນຢູ່ (seek ຍົກໄວ້)

disk, USB camera



**Major No.** von driver

command `mknod`  $/<\text{path}>$   $<\text{type}>$   $<\text{major}>$   $<\text{minor}>$

b / c

**Minor No.** von sub device

# Chapter 15 OS Security

เนื้อหา “ระบบจะตรวจสอบว่ามีการเข้ากันทั้งทางบานและทางไฟฟ้าเมื่อต้องการ”

Kali Linux → ชุด tool ที่สำคัญ  
security tool:

Intruders (crackers) ผู้บุกรุก → Kevin Mitnick (cracker ดังมาก)

Threat ภัยคุกคาม (บุ่มปะ โอมต์)

Attack จิตวิเคราะห์ อ้างอิงไปยังตัวแอล

Security Measurement Levels

ความซับซ้อน level ต่อ human

- Physical - ลายนิ้วมือ, กล้อง CCTV, ยาง
- Application - แอปพลิเคชันที่มีความผิดร้าย
- Operating System - TEE protection, debugging, logging
- Network - ผู้ส่ง traffic

phising - หลอกลวง

social-engineering - หลอกลวง password / บันทึก

$$N = \frac{1}{\text{ความปนกวน}}$$

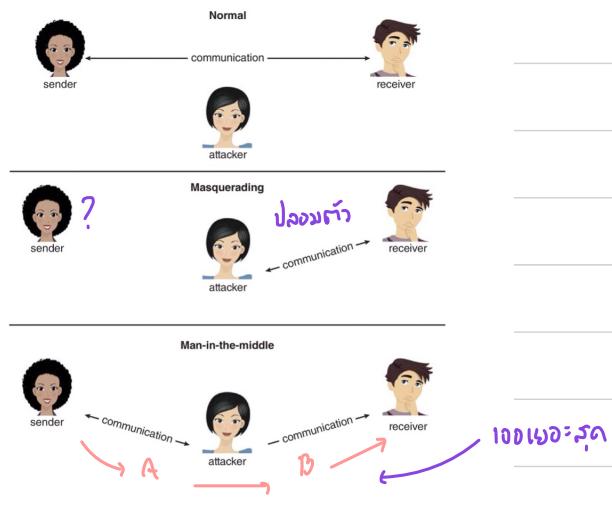
Program Threat

- Trojan horse คือ program ที่อยู่ใน program อื่นๆ
- Trap door (ประตูหลบ) programmer ที่ตั้งไว้เพื่อเข้าถึงหนังบ้านแบบง่ายๆ
- Salami Attack การโอนตัวเล็กหันบันยะสุม จนกลายเป็นการโอนมากๆ เช่น ห้องน้ำ ห้องเดียวบันยะห้องน้ำ
- Malware program ที่ทำให้เราเสีย ไฟล์ / ข้อมูล / ความเสียหาย คอมพิวเตอร์
- Viruses code fragment ใน program ที่ส่วนมากพากง่ายๆ เฉพาะเจาะจง CPU arch / OS
- Logic bomb มาจากคนใน เรียกว่า cookie monster (spam vs cookie + j lock file) ถ้าคนที่เข้ามาต้องตั้งค่า login แล้วจะปิดตัวเอง
- Keystroke logger program ที่ติดตั้งไว้ แล้วบันทึกรหัส password → เป็นไปได้ทั้ง SW / HW ให้หมด

System and Network Threat

- Worm standalone system ex. internet worm → exploit UNIX networking feature
- Port scanning ลองเช็คการของ port เปิดอยู่ใน กล่องของโคนวังน้ำทึบตัก port ไหน (nmap)
  - port disable port ที่ไม่ใช้งาน

## Standard Security Attack



## User Authentication

- password - ສິ້ນທີ່ຢູ່ (ວິເຄີອງ)
- ມົນຕະຫຼາກ - ສັງກຳນັບ (ໄປເນັດຈຸດ)
- Biometrics - ສິ້ນທີ່ຢູ່ (ຫົ້ວ, ຂ່າວຄາ)

or log invalid attempts

shoulder surfing - ໂດຍນົວ !?

Unix: encrypted password in /etc/shadow

## Security Defense

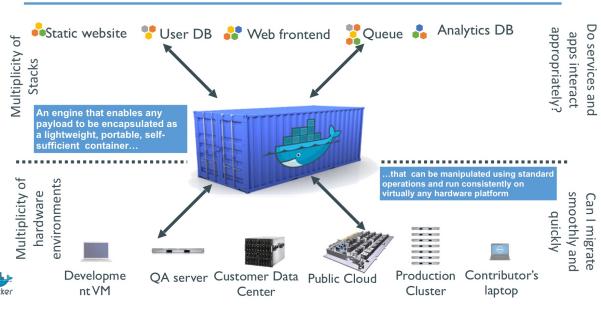
- 1) ໃຫ້ຄວາມຢູ່ກັບພິກສຕັ້ງ ex. ໄມ່ນໍານັກ, ເພື່ອນຫຼວງງານ
- 2) ໃຫ້ຄວາມຢູ່ເຮືອງ phising
- 3) lock ມີວັນ hardware
- 4) Configure OS ex. disable port, firewall
- 5) Update Software ex. log4j
- 6) logging
- 7) Anti-virus program
- 8) in Penetration Testing - ໂດຍ white hacker ມາວັດ hash ຢູ່ນັດ

## Code of Ethics ການ UseNIX

- 1) ເພື່ອກຳນົດວຸລະສ່ວນເກົາ ເມື່ອຈໍາເປັນທີ່ເກົ່າ
- 2) ເກັບຄວາມສັບໃນມືດ
- 3) ຕ້ອງແຫ່ງຈ່າຍເປົ້າ integrity, reliability and availability

# Chapter 16 Container and Docker

Docker ဆိုရင်ဝက်ပျော်ရွှေ့ dependencies ပါ။ versions မှာ self-contained → တစ်ခု pack software မြန်မာရ် environment



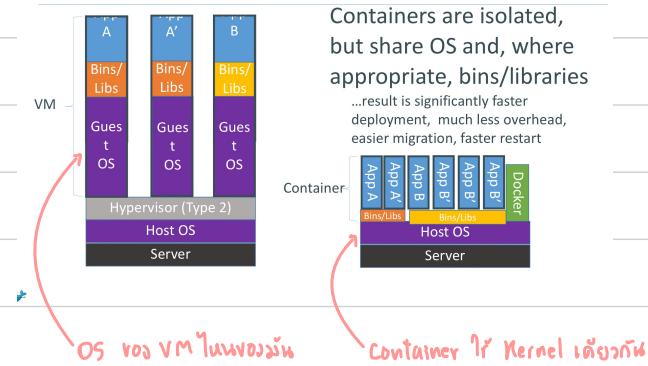
## Developers

"Build once run anywhere" (ဝေဆိပ်၍မြတ်စွာပေးသွန်နိုင်သူများ)

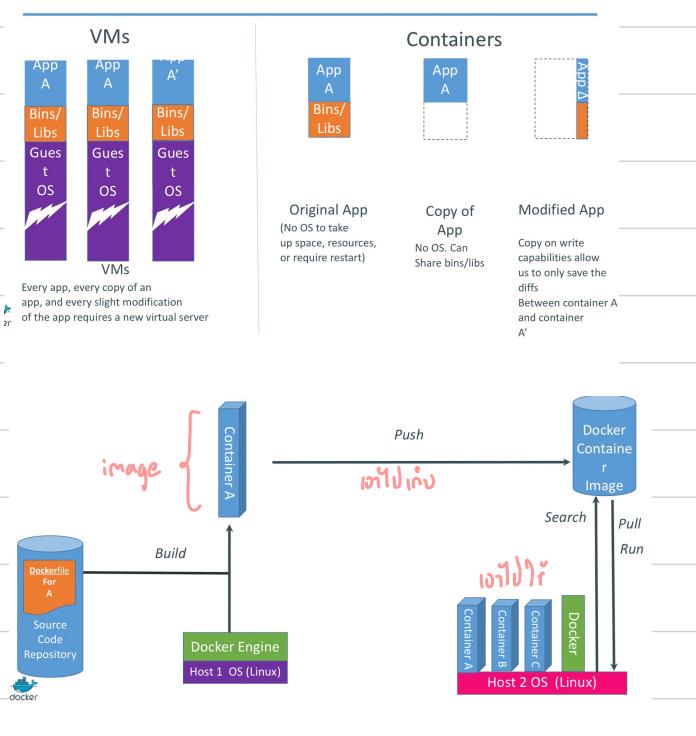
## Dev Ops

“Configure once run anything”

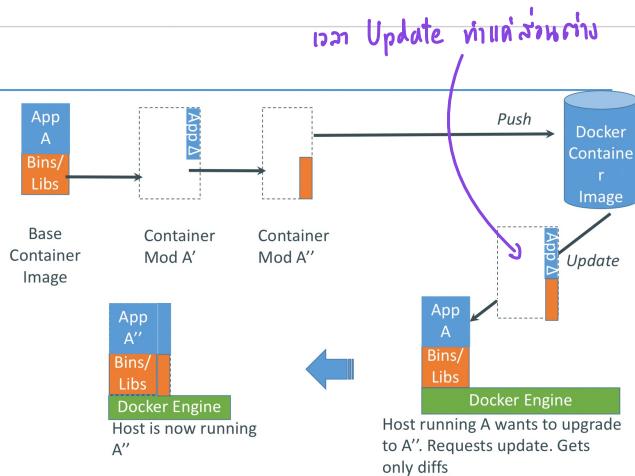
## VM vs Container



\* Container ពីនូ stateless (គណបត្រកែវឡើងទាំងអស់)



- ใน Dockerfile เวลา build จะแบ่งเป็น layer
  - สำหรับการสร้าง layer ใหม่หลังก็จะลบมันไปทิ้ง —> ประหยัด



# Docker Compose

ມີເພື່ອໄກ້ນຳໃຫຍງທາງ run docker ມາຍ, container ພຽບແຕ່ກີ່ —> Service Composition

## Swarm Mode

## ຕົກລົງພົມອາຮົາ / Cluster Management / Orchestration