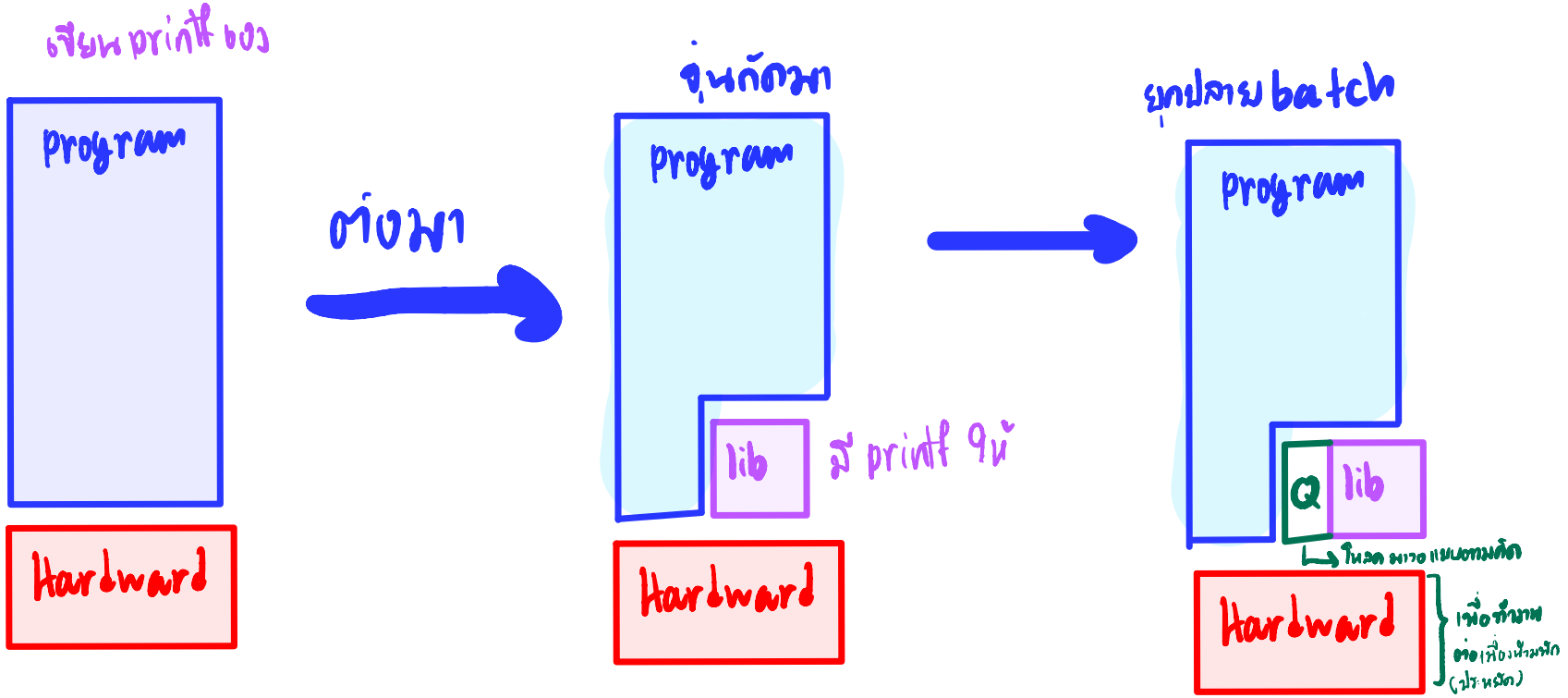
The Kernel Abstraction

Slides adopted from CSE 451 class at UW, CS162 class at Berkeley and CSE 421/521 class at UB

อดีต < 1940

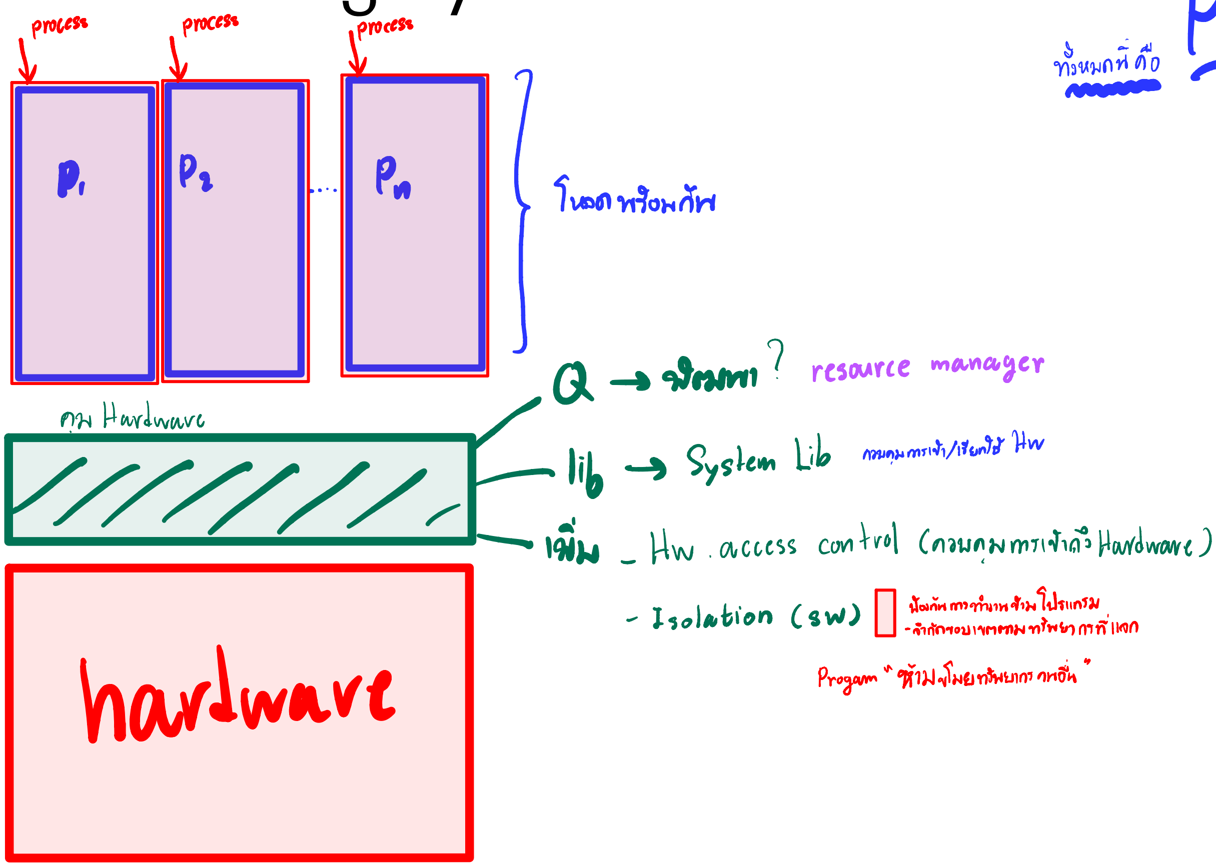
- Single task system



ปัจจุบัน > 1980

"Protection"

- Multitasking system



Activity #1

- การเปลี่ยนแปลงจากระบบแบบ single task ไปเป็นระบบแบบ Multitask ... สิ่งที่จะต้องมีการปรับแต่งหรือเพิ่มเติมเข้ามาใน OS ได้แก่...

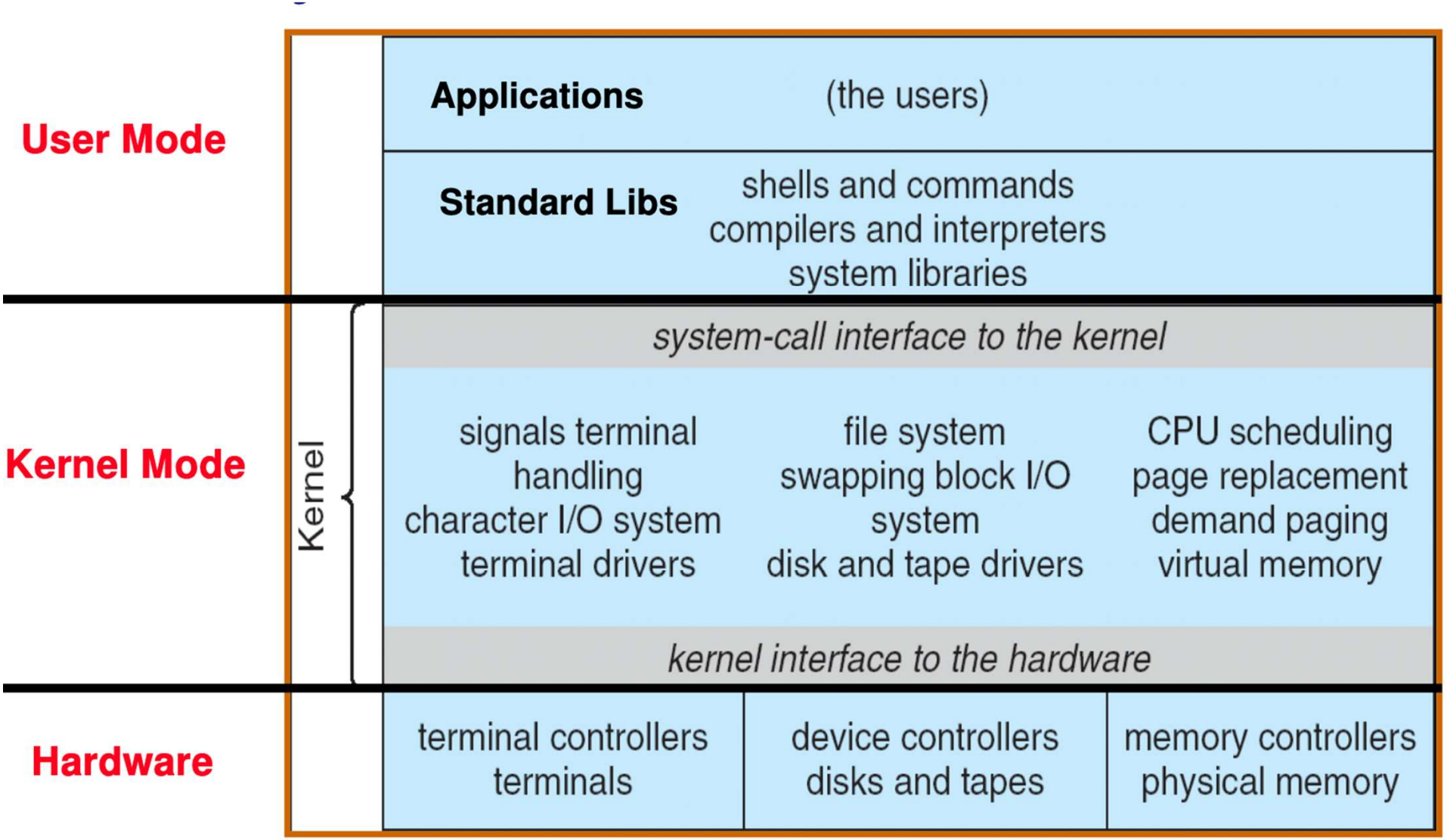
What does an OS do...

- Hiding Complexity
  - Variety of HW
    - E.g. different CPU, amount of RAM, I/O devices

- Kernel is the part of the OS that running all the time on the computer
  - Core part of the OS
  - Manages system resources
  - Acts as a bridge between apps and HW

1980

UNIX Structure

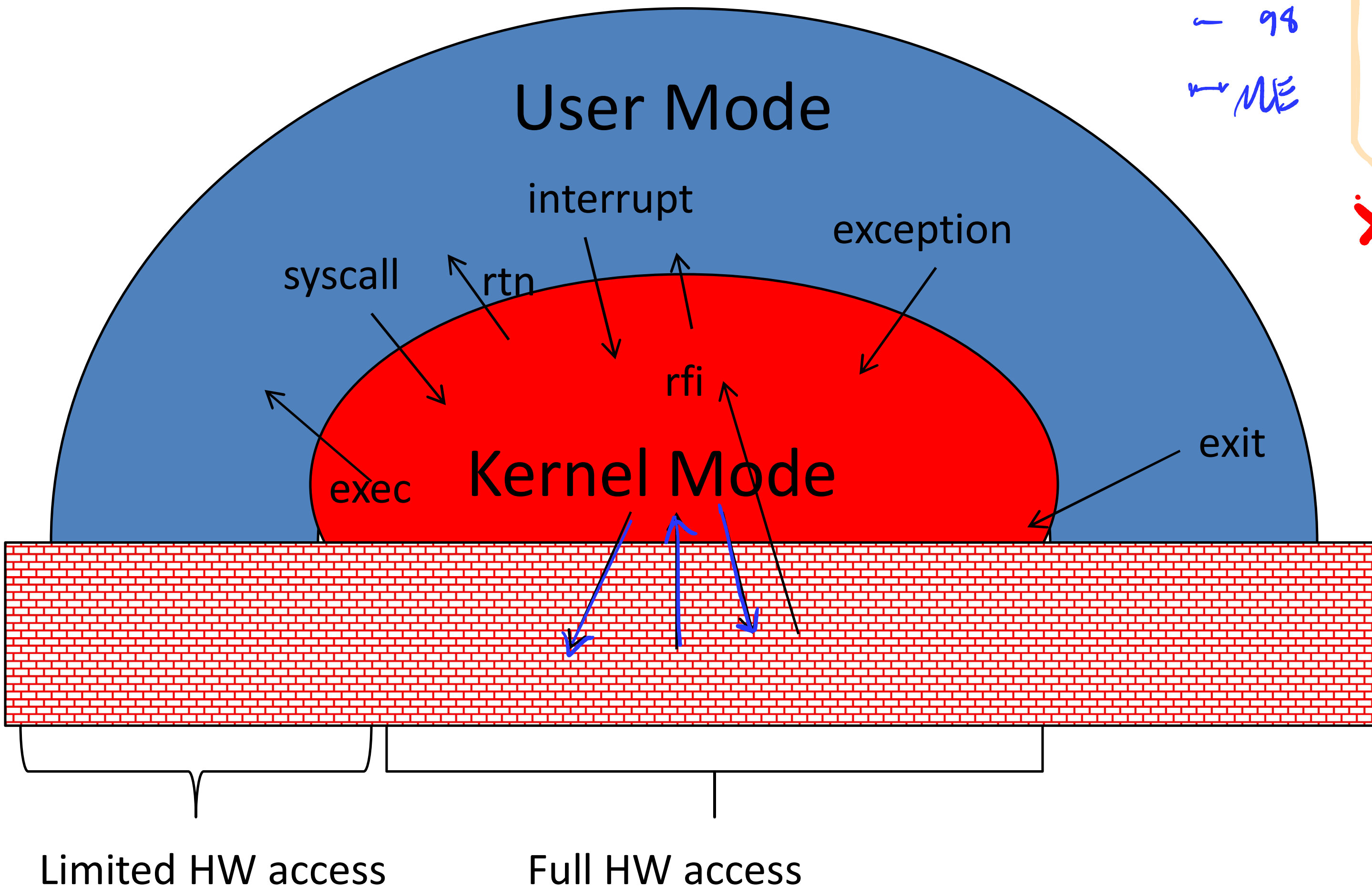


ระบบกลางเก่า กลางใหม่

User/Kernel (Privileged) Mode

เพื่อที่เครื่องไม่เปลี่ยน ใช้ OS ตัวใหม่ได้ ← win95 – 98 – ME

บางทีมี program เก่าที่เข้าถึง HW โดยตรง (เครื่องเก่า) เลยออกแบบมาแบบนี้



เข้าถึง Hardware โดยตรง

ข้อเสีย ระบบไม่เสถียร/รวน

HW บางชิ้นใช้ได้แค่ Pro. เดียว

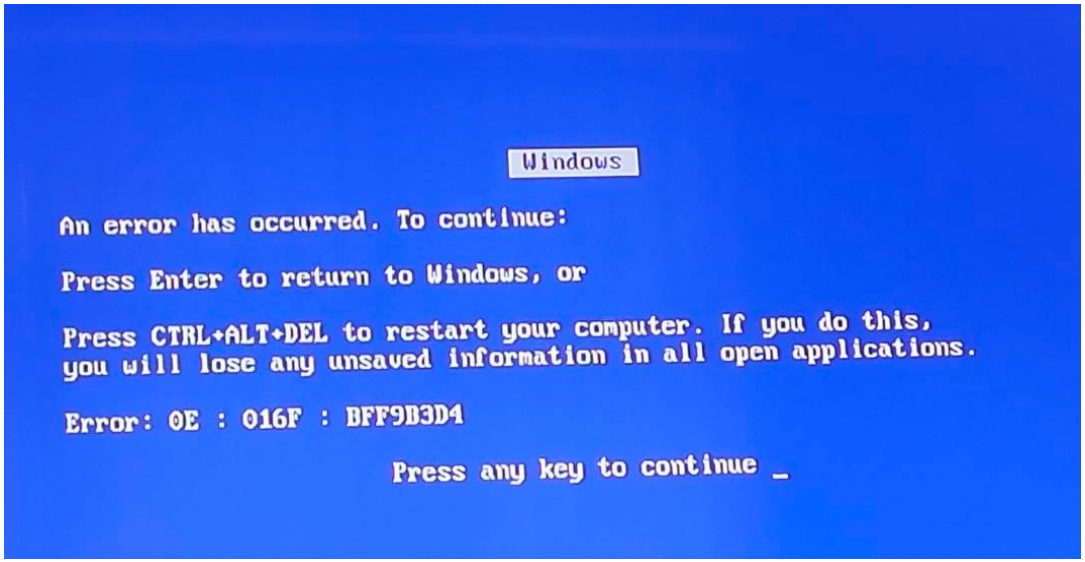
ใช้ Pro. พร้อมกันไม่ได้

(HW บางชิ้นใช้ได้หลาย Pro. แล้วมาแย่งกัน

HW Pro. เดียว = รวน

One of the major goals of OS is...

- Protecting Process and the Kernel
  - Running multiple programs
    - Keep them from interfering with the OS - kernel
    - Keep them from interfering with each other

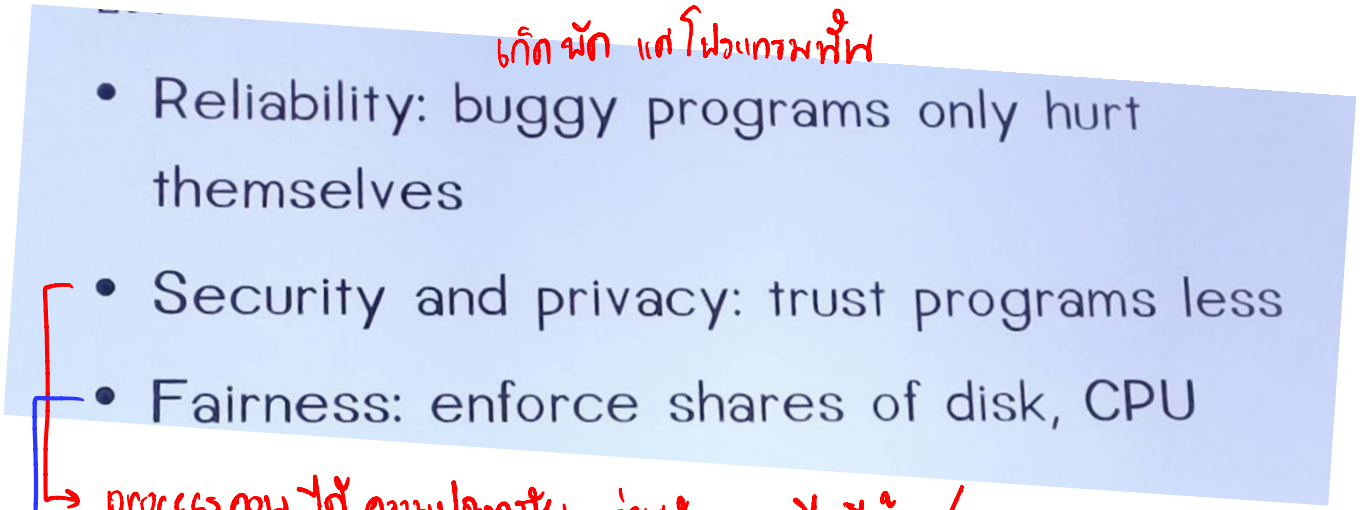


จอฟ้า = ทำผิดกฎ
↳ P_1 ไปยุ่งหน้าที่ P_2
1. นิดระบบ
2. (ถ้า 1 ไม่ได้) ปิดตัวเอง → จอฟ้า

Activity #2: Protection: WHY?

เวลา 10 นาที

การ protect Process และ Kernel ทำให้เกิด impact อะไรกับระบบบ้าง และยังต้อง protect อะไรอีกบ้าง เพื่ออะไร

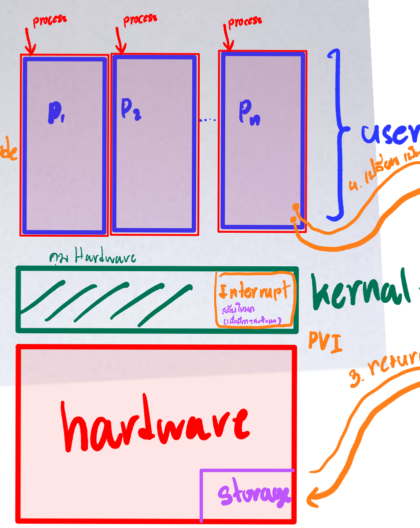


Hard soft

Protection: How? (HW/SW)

- 2 Main HW mechanism
  - Memory address translation — map addr process กับ addr จริง (Virtual addr)
  - Dual mode operation — Microprocesscer แยกคำสั่ง 2 กลุ่ม
    - ชุดคส.1 • Privileged/non-privileged mode — หรือ kernal mode / user mode
    - ชุดคส.2 • System calls — ชุดคำสั่งประมวลผล if + - x ÷ / ไม่เกี่ยว I/O HW อ่านดิส / loop
- SW
  - Process — จะใช้ส่วน user / ใช้ส่วน virtual addr
  - System calls

ต้องการ save ไว้ใน Hardisk



1 เปลี่ยนโหมดเป็น kernal (System Call "การเรียกใช้ Sys Lib)

Lib System Lib

2

3. return

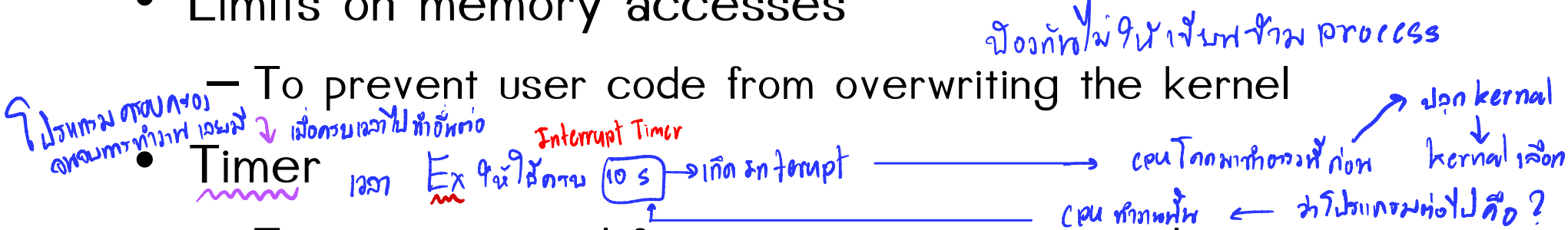
4. เปลี่ยนเป็น user mode

Hardware Support: Dual-Mode Operation

- Kernel mode
  - Execution with the full privileges of the hardware
  - Read/write to any memory, access any I/O device, read/write any disk sector, send/read any packet
- User mode
  - Limited privileges
  - Only those granted by the operating system kernel
- On the x86, mode stored in EFLAGS register
- On the MIPS, mode in the status register

Hardware Support: Dual-Mode Operation

- Privileged instructions
  - Available to kernel
  - Not available to user code
- Limits on memory accesses
  - To prevent user code from overwriting the kernel
- Timer
  - To regain control from a user program in a loop
- Safe way to switch from user mode to kernel mode, and vice versa



Privileged instructions

- Examples?

- What should happen if a user program attempts to execute a privileged instruction?

-โปรแกรม error ถูกปิด

-ดองฟ้า

User Mode

- Application program
  - Running in process

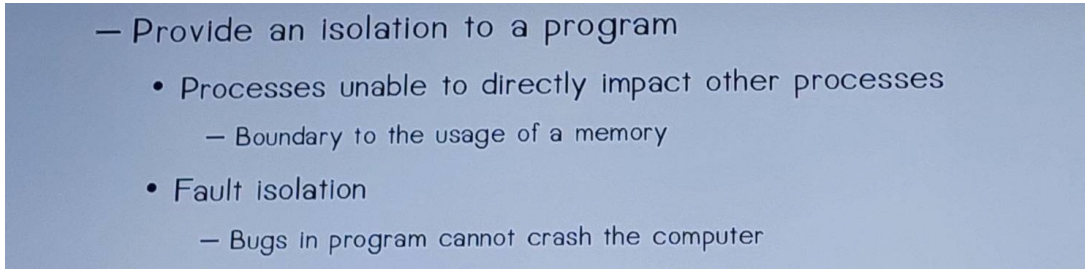
Virtual Machine:VM

คอมพิวเตอร์เสมือน

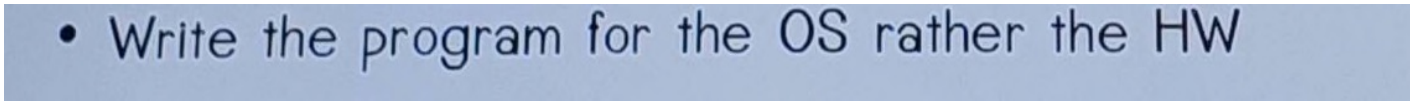
- Software emulation of an abstract machine
  - Give programs illusion they own the machine
  - Make it look like HW has feature you want
- 2 types of VM
  - Process VM จำลองแยกให้ OS เดียวสั่งทำงานได้
    - Supports the execution of a single program (one of the basic function of the OS)
  - System VM จำลองระดับ HW
    - Supports the execution of an entire OS and its applications

Process VMs

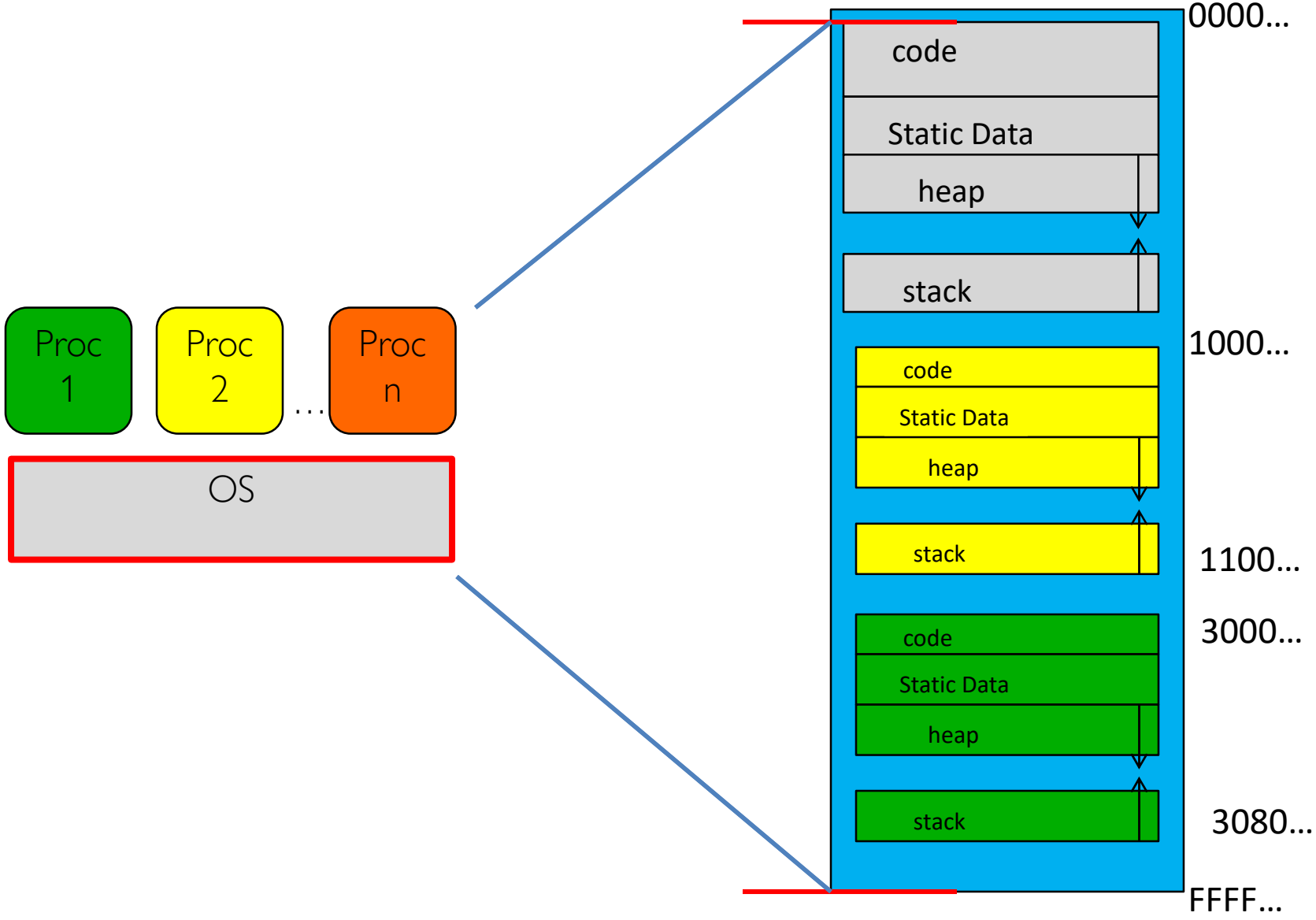
- GOAL: เป้าหมายการสร้าง
  - Provide an isolation to a program
    ไม่รบกวน ซึ่งกันและกัน/ระบบ



  - Portability (Program) SW เขียนบน window ก็บน window ได้
    HW ต่างกันได้



Kernel mode & User mode

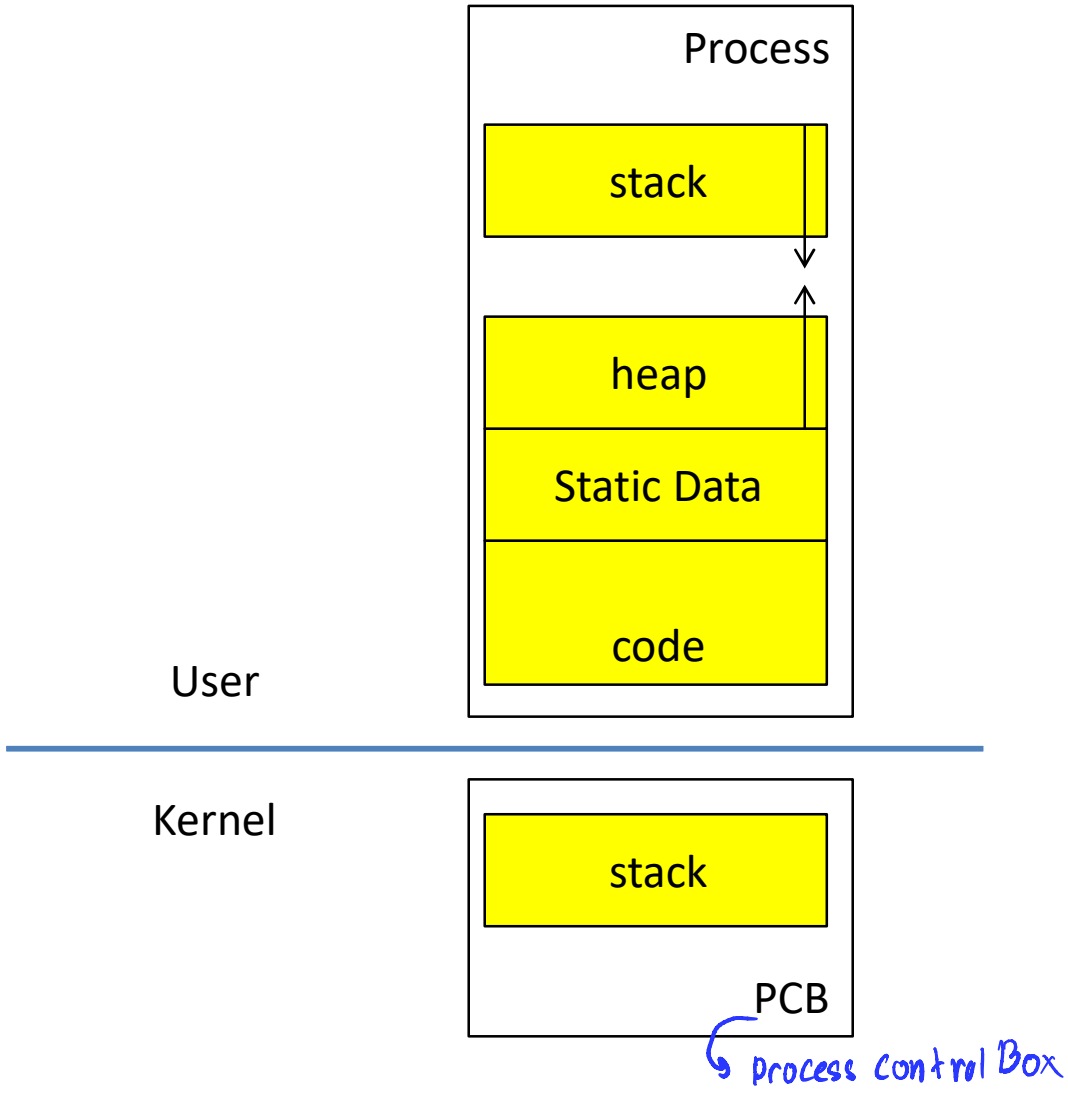


Process Abstraction

- Process: an *instance* of a program, running with limited rights
  - Thread: a sequence of instructions within a process
    - Potentially many threads per process (for now 1:1)
  - Address space: set of rights of a process
    - Memory that the process can access
    - Other permissions the process has (e.g., which system calls it can make, what files it can access)

Process

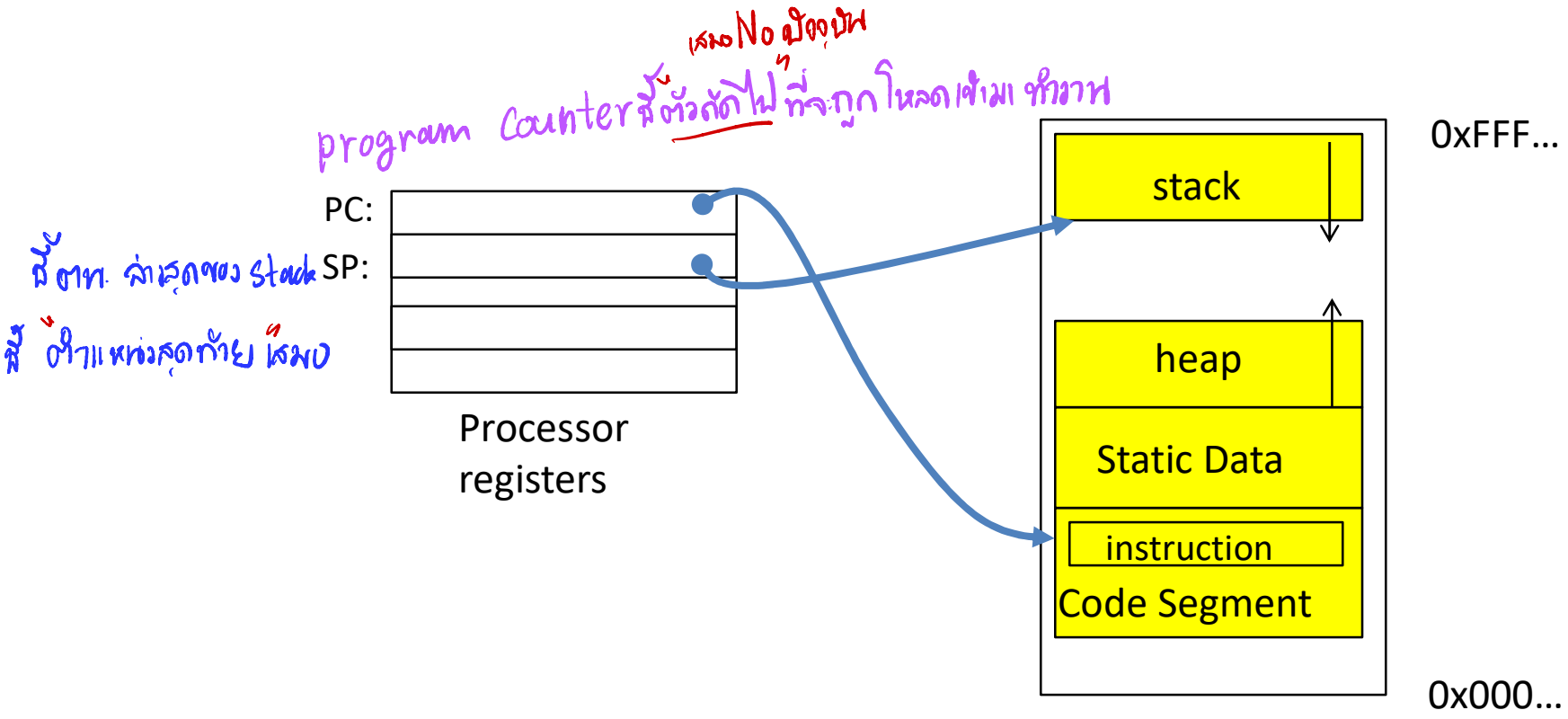
- 2 parts
  - PCB in kernel
  - Others in user



Process Control Block: PCB

- Kernel represents each process as a process control block (PCB)
  - Status (running, ready, blocked, ...)
  - Registers, SP, ... (when not running)
  - Process ID (PID), User, Executable, Priority, ...
  - Execution time, ...
  - Memory space, translation tables, ...
- Kernel Scheduler maintains a data structure containing the PCBs
- Scheduling algorithm selects the next one to run

Address Space: In a Picture



Break