Lập trình Hệ thống

Unix Programming

Introduction Os

Nguyễn Quốc Tuấn

Network and Communication System Department
Faculty of Electronics and Communications
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Introduction

❑ MỤC TIÊU

- Giúp sinh viên làm quen với môi trường lập trình trên Linux (Ubuntu)
- Tìm hiểu kiến trúc của Linux, một số thư viện lập trình
- Làm việc với các bài toán hệ thống cơ bản trên Linux như
  - ✓ Vào ra với hệ thống file,
  - ✓ Quản lý tiến trình,
  - ✓ Lập trình đa nhiệm, đa người dùng

INTRODUCTION

Outcome (Yêu cầu đạt được)	Level 1	Level 2	Level 3	Level 4
1. Kiến thức				
- Các hệ điều hành	x			
- Lập trình cơ bản Shell, C++ - Hiểu được các vấn đề lập trình hệ thống		x		
- Kiến trúc của Linux,		x		
- Một số thư viện lập trình		x		
- Bài toán hệ thống cơ bản trên Linux		x		
		x		

INTRODUCTION

Outcome (Yêu cầu đạt được)	Level 1	Level 2	Level 3	Level 4
1. Kỹ năng				
- Áp dụng hiểu biết xây dựng chương trình		x		
- Phân định các bài toán hệ thống				
- Hệ thống file	x			
- Quản lí tiế trình	x			
- Thực hành 1 project nhỏ	x			

Level 1: Ability to know (concepts)
Level 2: Ability to understand (methods in detail)
Level 3: Ability to apply (e.g., to solve problems using given methods)
Level 4: Ability to analyze/synthesize (e.g., compare different methods)

INTRODUCTION

- **Thời lượng**
  - 03 tín chỉ : 30 giờ bài giảng
    15x2 giờ thực hành
  - Đánh giá bài tập
    - Thực hành – giữa kì (30%)
    - Cuối kì (70%)
  - Tài liệu

    - Unix in a Nutshell

      Arnold Robbins

      Published by O'Reilly & Associates 1999

    - Advanced Linux Programming,

      Mark Mitchell, Jeffrey Oldham, và Alex Samuel,

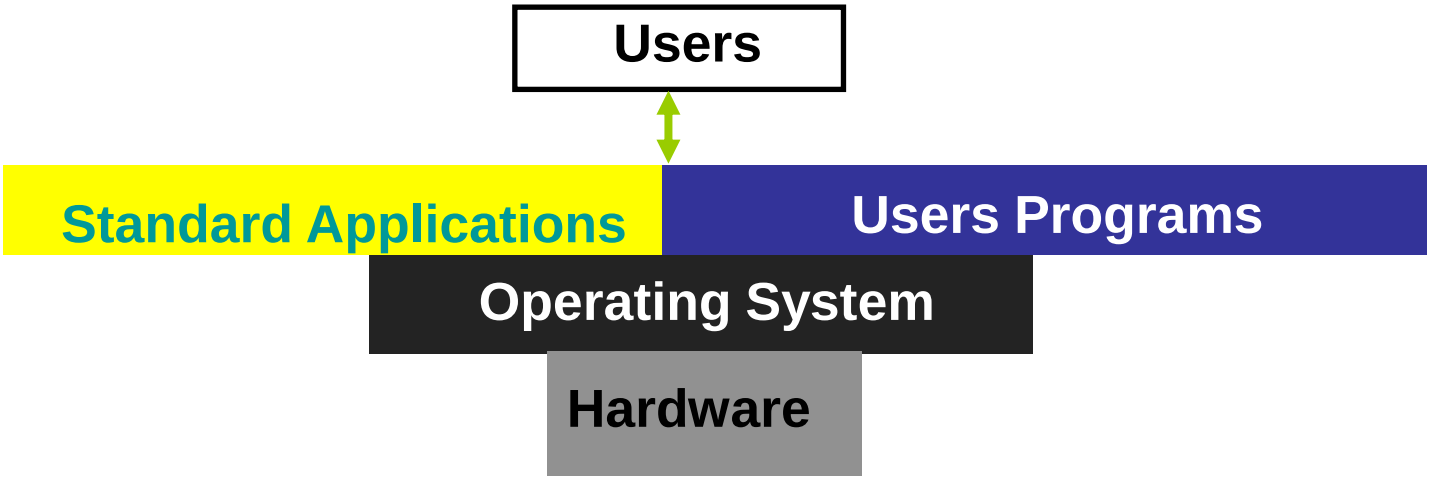
      New Riders Publishing, 2001

    - Giáo trình: Lập trình C/C++ trên Linux,

      Nguyễn Trí Thành, 2010

HỆ ĐIỀU HÀNH

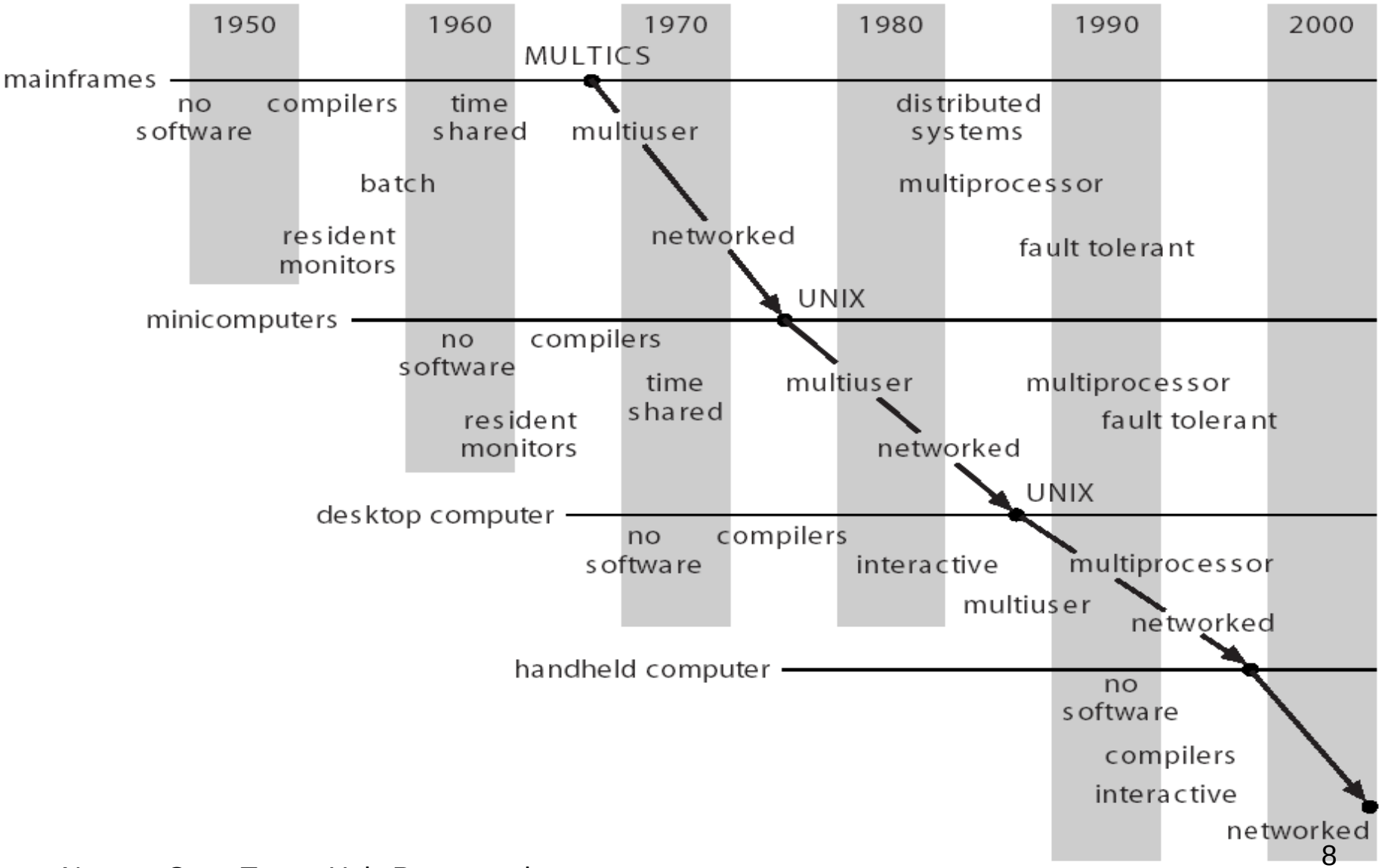
- Hệ điều hành :
  - Một chương trình quản lí máy tính
  - Một phần mềm biến phần cứng thành một thứ hữu ích
  - Hình ảnh phân lớp: Phần cứng, Hệ điều hành, Ứng dụng



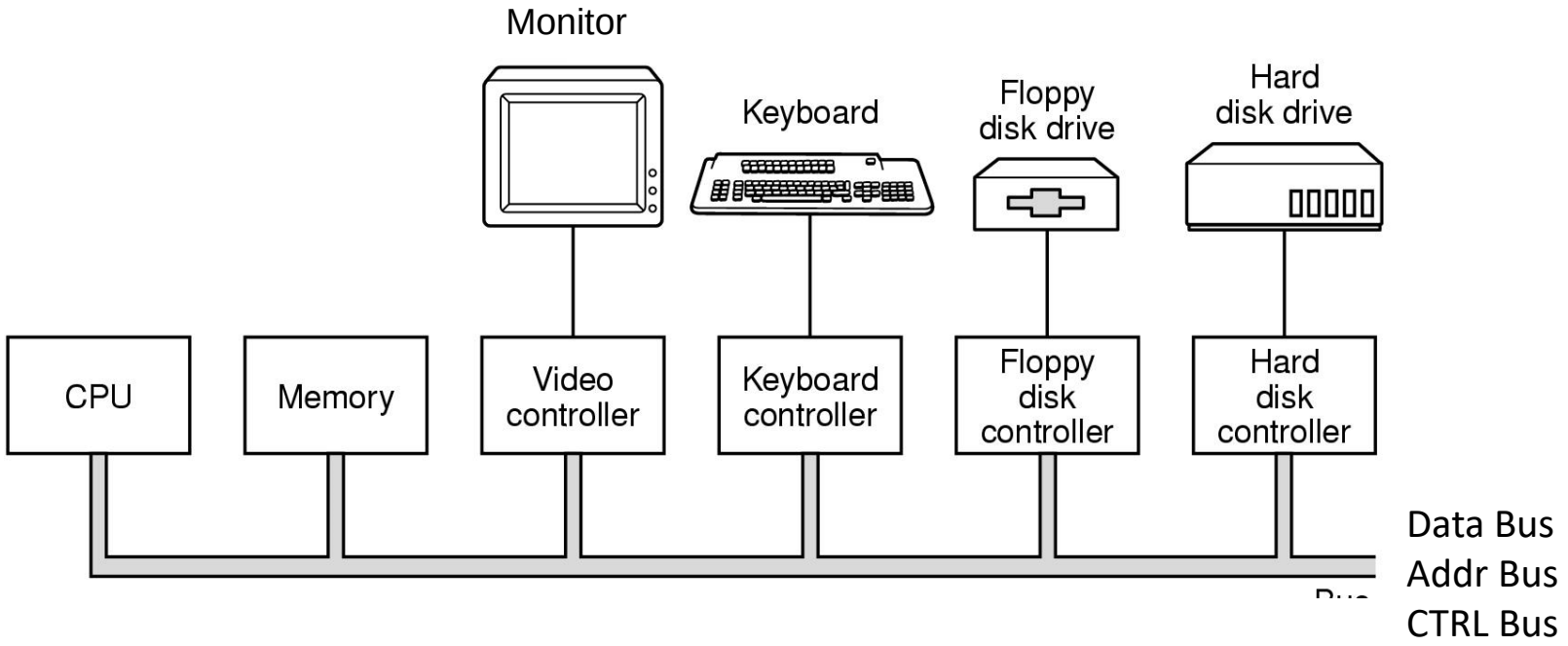
MỘT VÀI OS

- Linux
- Unix
- Windows
- BSDUnix
- Mac/Tiger
- Vista
- Ubuntu
- PalmOS
- TinyOS
- WinCE
- Symbian
- ….

QUÁ TRÌNH PHÁT TRIỂN OS

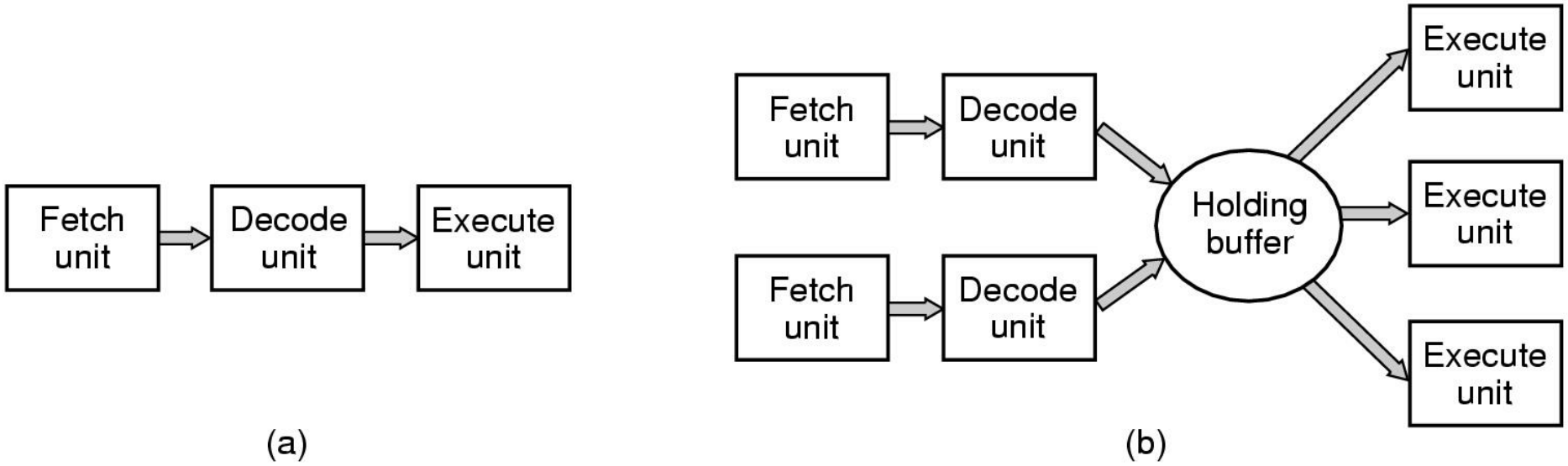


QUÁ TRÌNH PHÁT TRIỂN OS



- Các thành phần của một máy tính cá nhân đơn giản
  - CPU
  - MEM
  - I/O

CHU TRÌNH LỆNH MÁY TÍNH

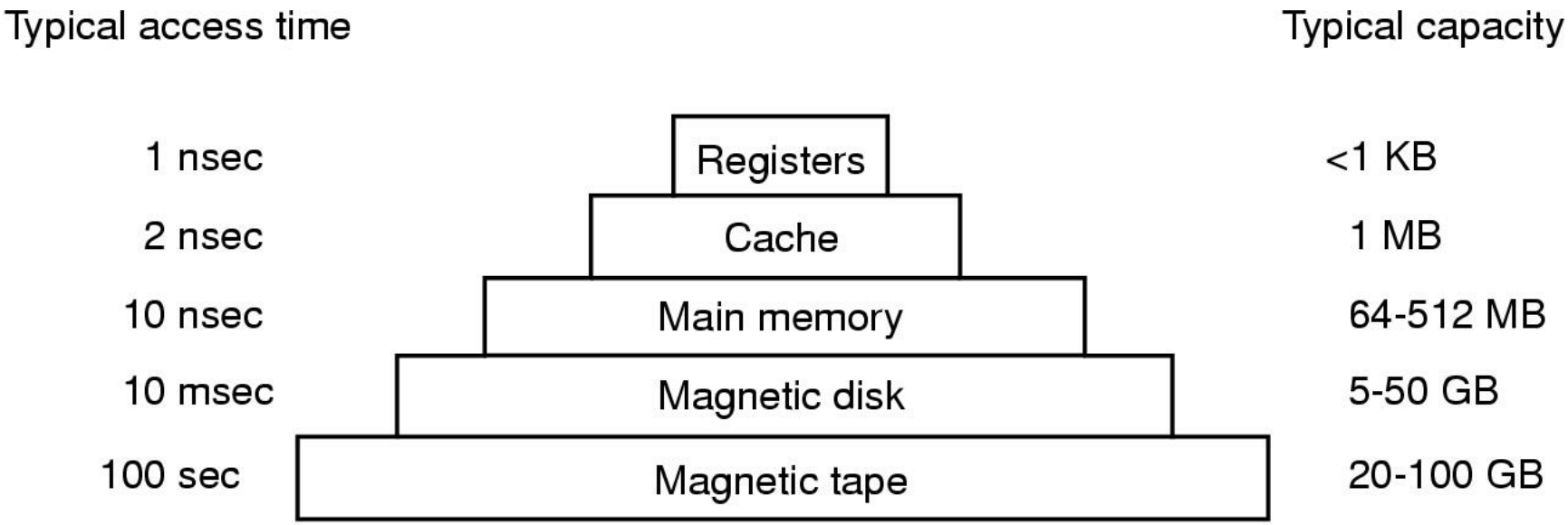


(a) Đường ống ba giai đoạn

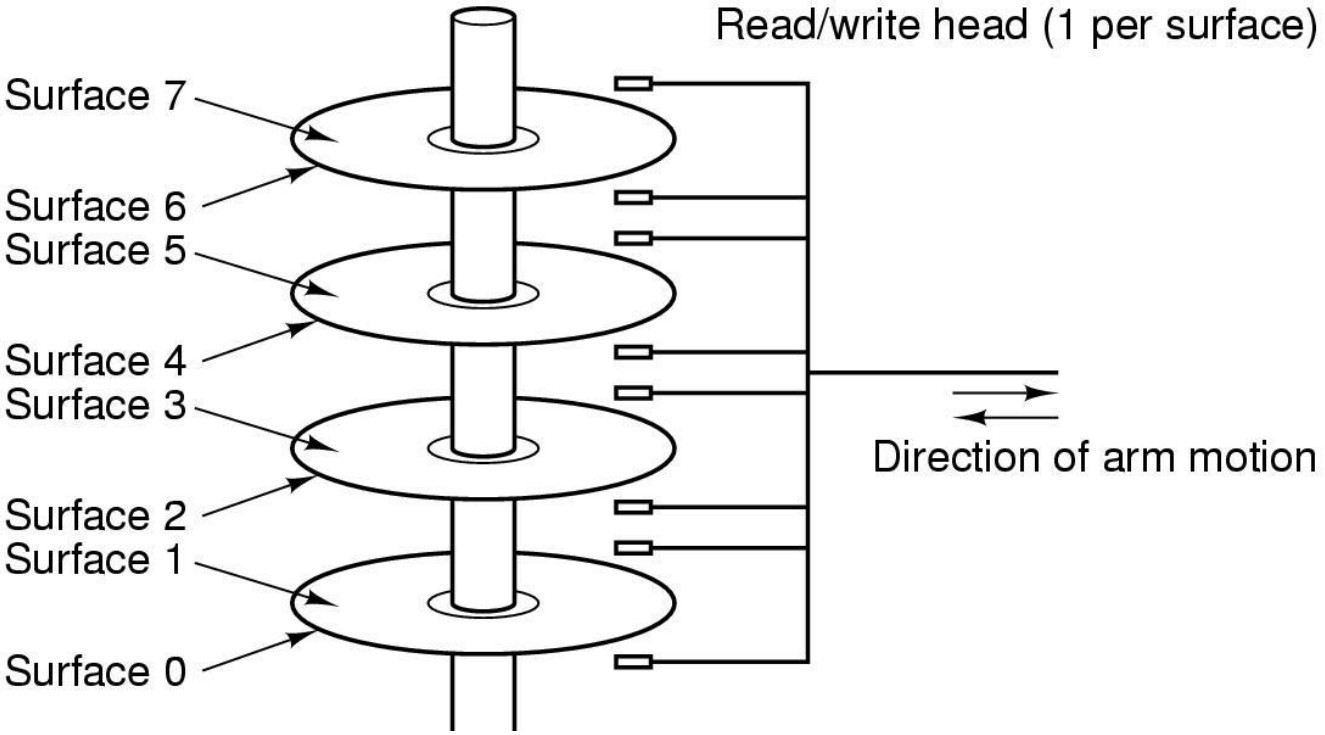
(b) Một CPU siêu thanh

PHẦN CỨNG MÁY TÍNH

- Hệ thống phân cấp bộ nhớ điển hình

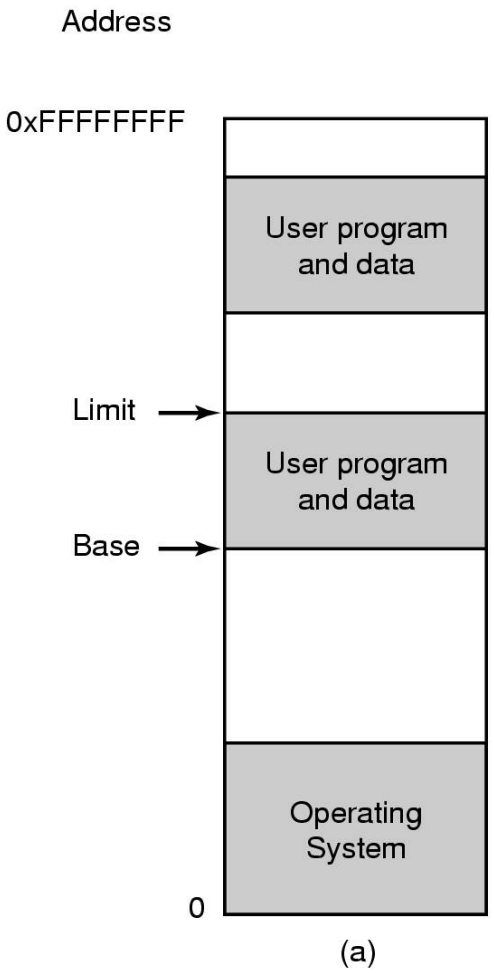


PHẦN CỨNG MÁY TÍNH

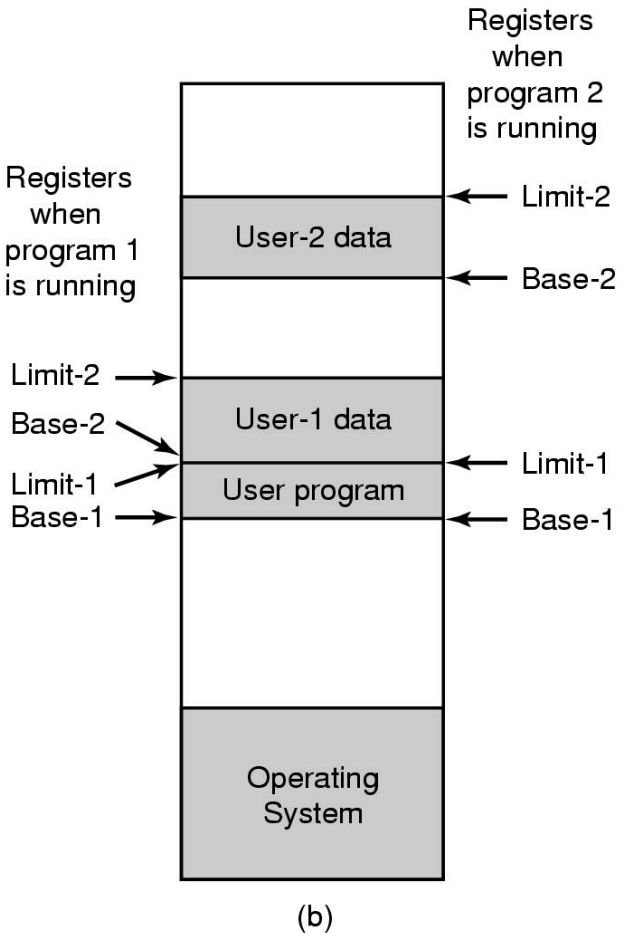


Cấu trúc ổ đĩa

PHẦN CỨNG MÁY TÍNH

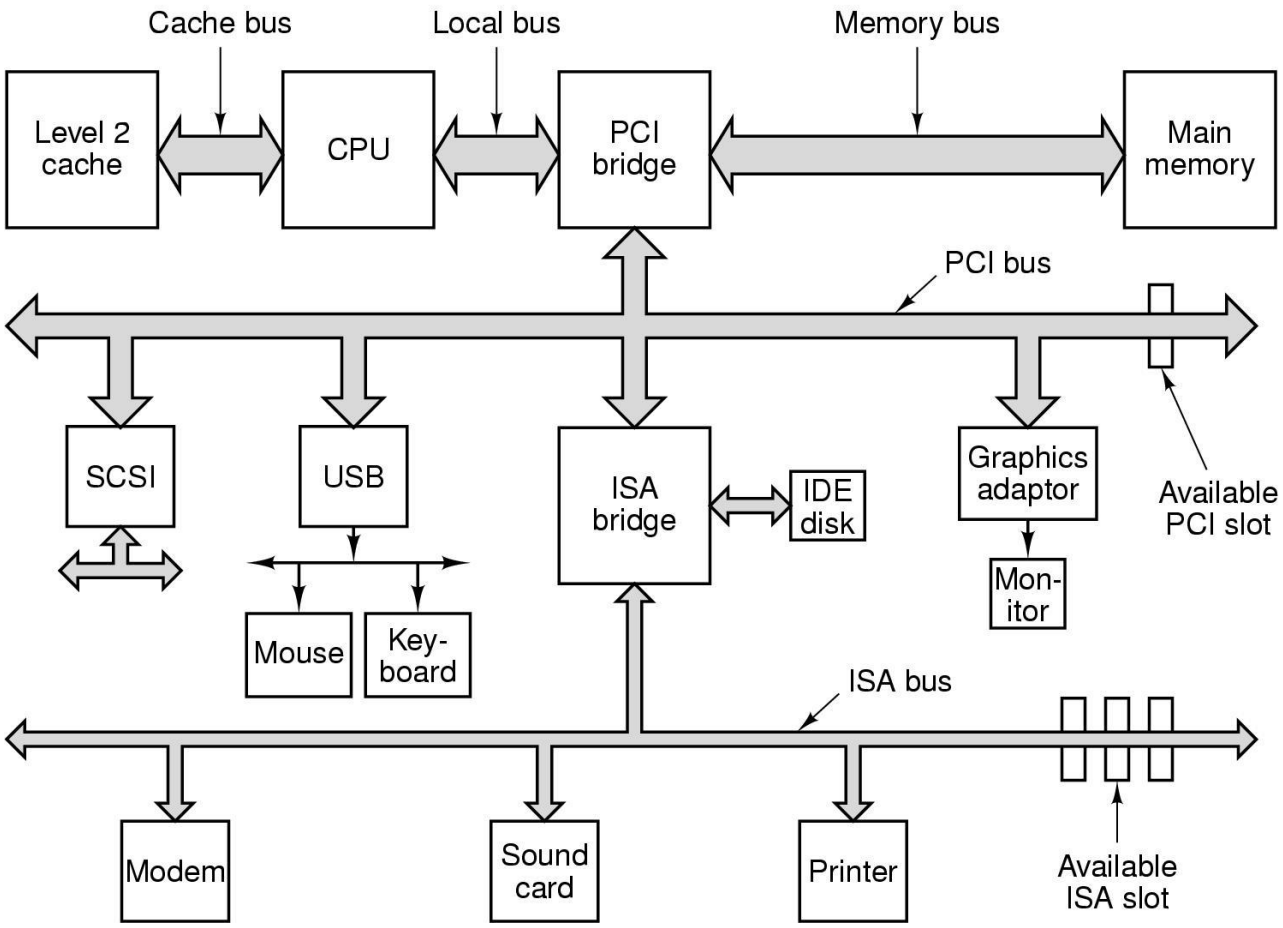


a) Một cặp giới hạn cơ sở



b) Hai cặp giới hạn cơ sở

PHẦN CỨNG MÁY TÍNH



Cấu trúc của một hệ thống Pentium lớn

TÓM LƯỢC Os

Nguồn tài nguyên	Dịch vụ sử dụng
▪ Phân bổ	▪ Trừu tượng
▪ Bảo vệ	▪ Đơn giản hóa
▪ Yêu cầu sử dụng	▪ Tiện lợi
▪ Ảo hóa	▪ Tiêu chuẩn hóa

Làm cho máy tính đơn giản hơn (quản lý, sử dụng và hiểu)

TÓM LƯỢC Os

Tài nguyên

- Phân bổ
- Sự bảo vệ
- Yêu cầu
- Ảo hóa

Tài nguyên hữu hạn
Nhu cầu cạnh tranh
Ví dụ:
CPU
Bộ nhớ
Đĩa
Mạng

Government

Limited budget,
Land,
Oil,
Gas,

Printer problem: buffer! (Time Multiplexing)
Space Multiplexing: give CPU to one application, disk to another application

TÓM LƯỢC Os

Tài nguyên

- Phân bổ
- Sự bảo vệ
- Yêu cầu
- Ảo hóa

Bạn không thể làm tổn thương tôi
Tôi không thể làm tổn thương bạn

Hàm ý một số mức độ an toàn và bảo mật

Government

Law and order

TÓM LƯỢC Os

Tài nguyên

- Phân bổ
- Sự bảo vệ
- Yêu cầu
- Ảo hóa

Hệ điều hành cho

Hệ điều hành lấy đi

Tự nguyện tại thời gian chạy

Không tự nguyện

Hợp tác

Government

Income Tax

TÓM LƯỢC Os

Tài nguyên

- Phân bổ
- Sự bảo vệ
- Yêu cầu
- Ảo hóa

Ảo tưởng về tài nguyên
vô hạn, riêng tư

Bộ nhớ so với đĩa
CPU được chia sẻ
thời gian

Nhiều trường hợp khắc
nghiệt hơn có thể
(& tồn tại)

Government

Social security

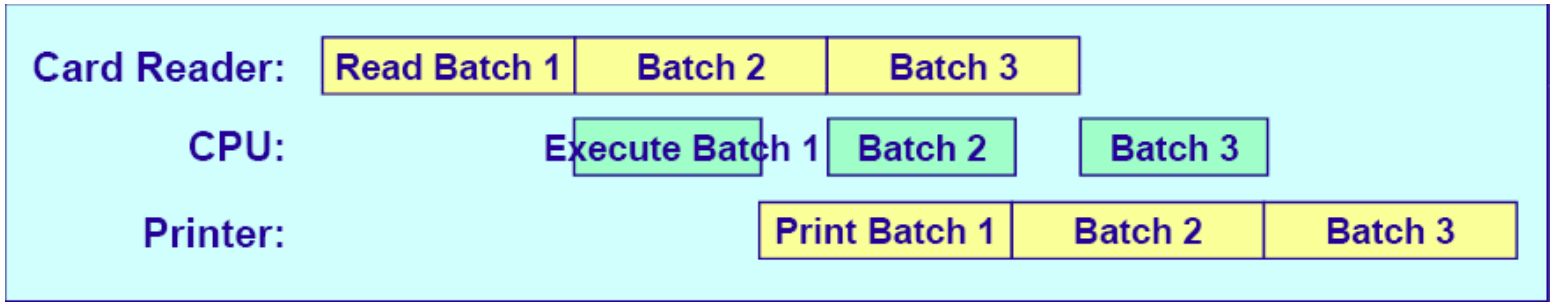
CÁC ĐẶC TÍNH Os

- Một cơ chế lập lịch công việc hoặc tiến trình
- Phương pháp để thực thi đồng thời CPU và xử lý IO
- Chia sẻ thời gian

  + Môi trường đa chương trình

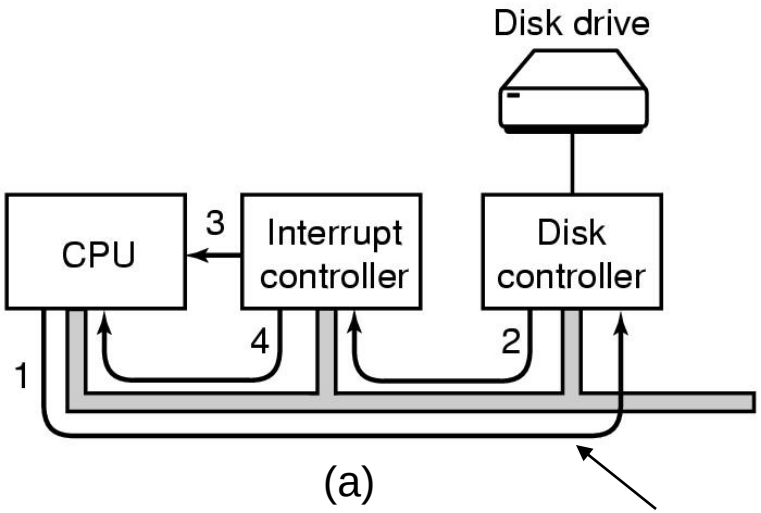
  + Có thể tương tác

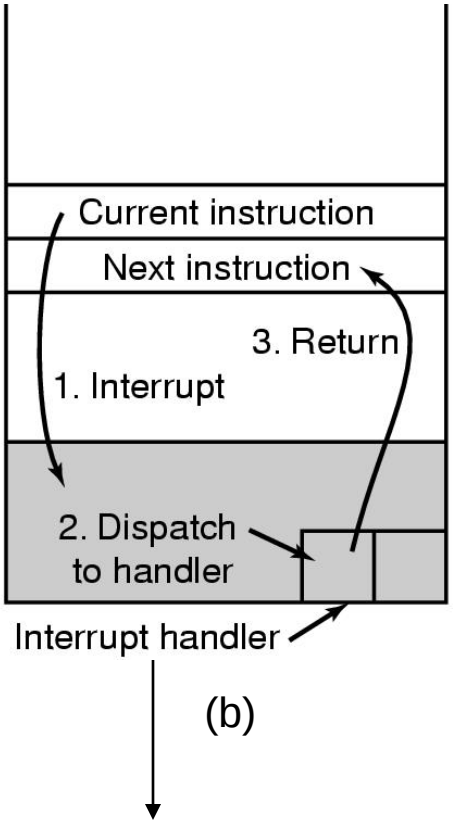
- Đa xử lí
- Xử lí ngoại tuyến (offline)



NGẮT

read (fd,buffer,nbytes)





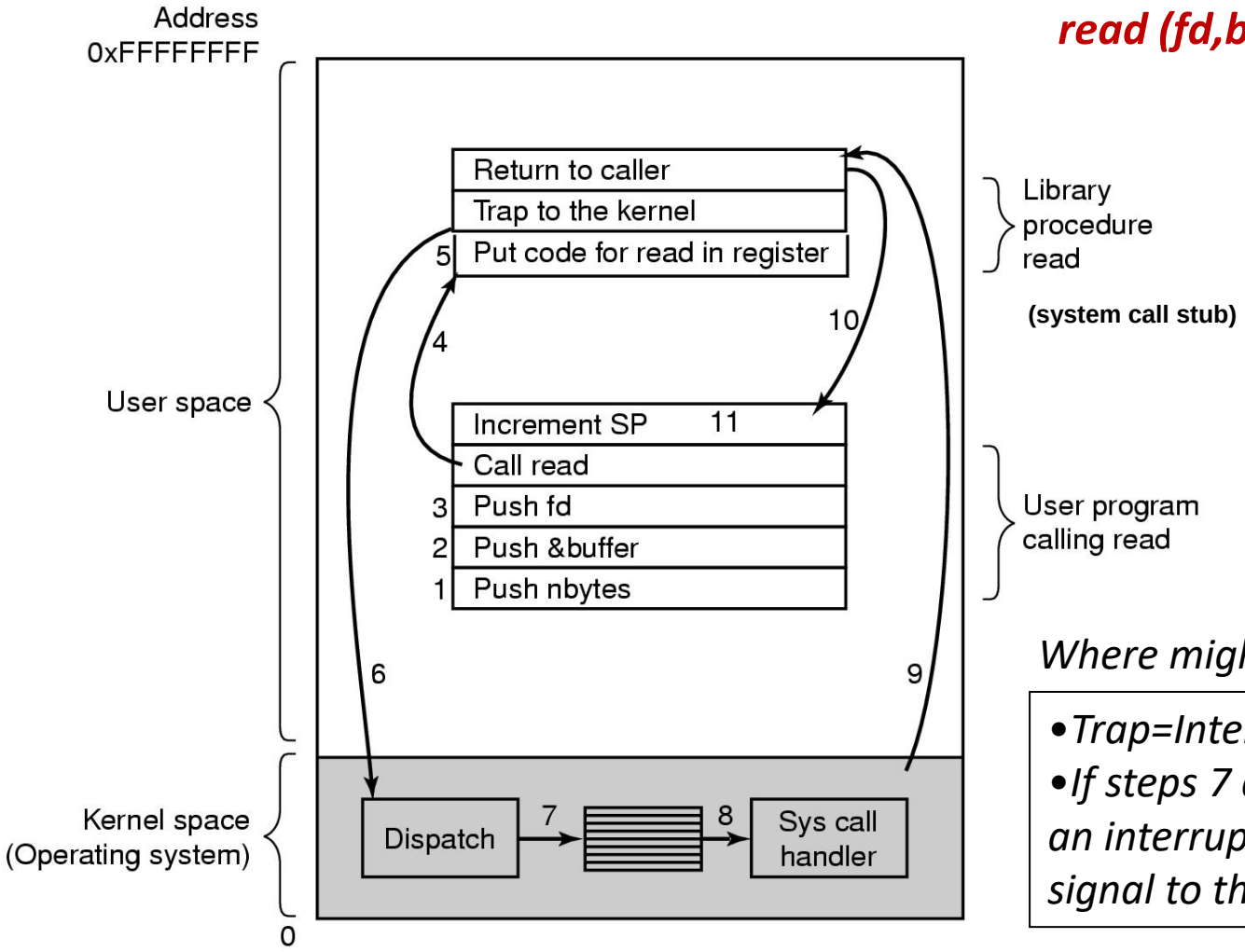
Steps 3 and 4 in (a)

(a) Các bước bắt đầu I / O thiết bị và bị gián đoạn

(b) Cách CPU bị ngắt

*"Interrupt handler" is a part of the Code for the **device driver***

HOẠT ĐỘNG MỘT CUỘC GỌI HỆ THỐNG



read (fd,buffer,nbytes)

Where might an interrupt be used?

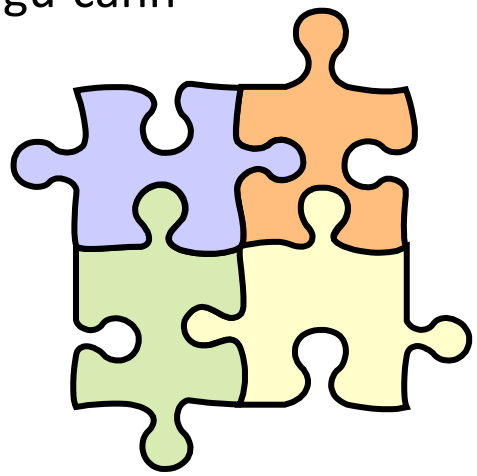
- *Trap=Interrupt*
- *If steps 7 and 8 take long, an interrupt is used later to signal to the CPU when job is done*

CÁC CUỘC GỌI HỆ THỐNG

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

THÀNH PHẦN Os CHÍNH

- Bootstrapping
- Quản lý nguồn tài nguyên
  - Quản lý CPU
  - Quản lý bộ nhớ
  - Quản lý thiết bị I / O
- Quy trình, quản lý luồng, chuyển đổi ngữ cảnh
- Hệ thống file
- Shell

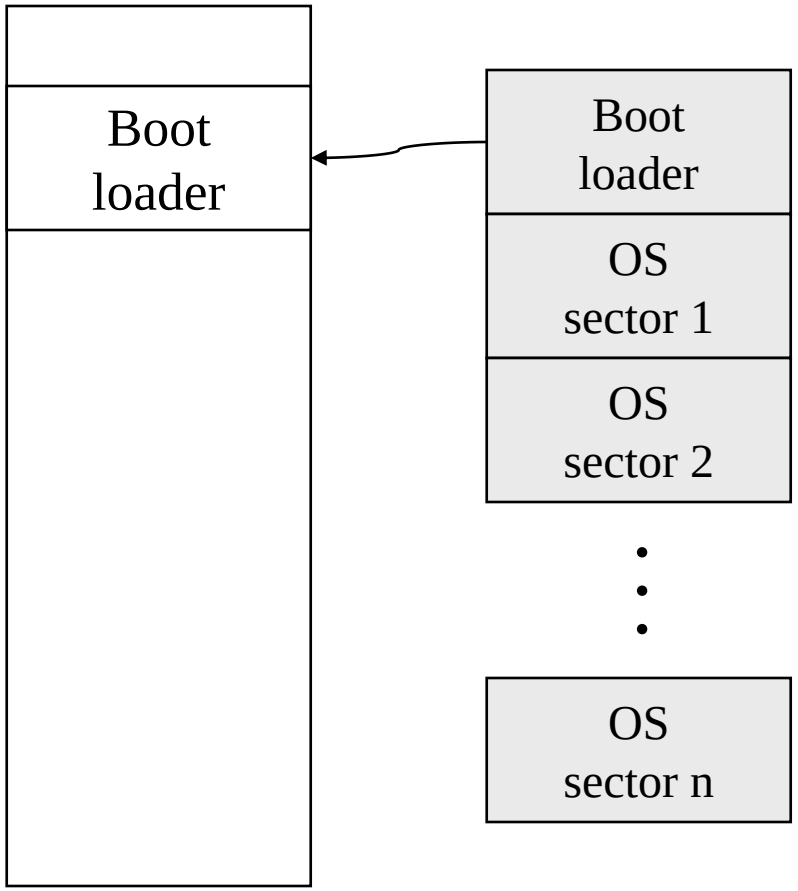


BOOTSTRAPPING

Is this done in user-level or kernel-level?

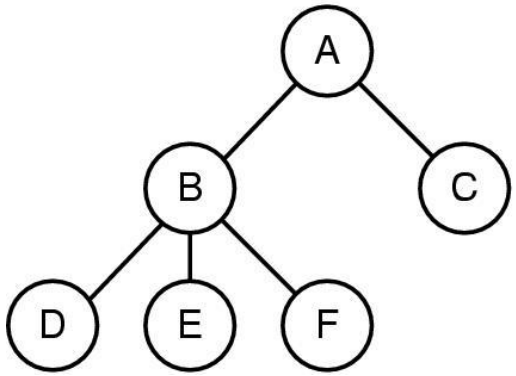
- Bật nguồn cho máy tính
- Reset bộ xử lí
  - Đặt trạng thái biết trước
  - Chuyển (jump) mã ROM

1. Nạp vào bộ khởi động tải từ bộ nhớ ổn định (Ổ cứng/mạng)
2. Chuyển đến bộ khởi động
3. Tải phần còn lại của hệ điều hành
4. Khởi tạo và chạy



TIẾN TRÌNH

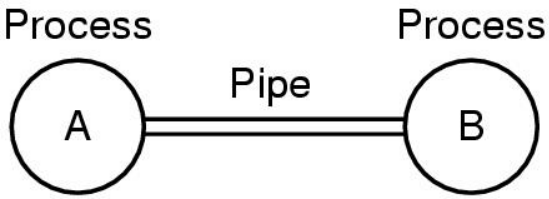
- **Chương trình = mã**
- **Process (Tiến trình) = chương trình đang chạy.** Bao gồm:
  Mã, dữ liệu, ngăn xếp, bộ đếm chương trình, giá trị thanh ghi, bất kỳ thứ gì mô tả "trạng thái" hiện tại của tiến trình
- Một cây tiến trình
  - Tiến trình A đã tạo hai tiến trình con, B và C
  - Tiến trình B đã tạo ra ba tiến trình con D, E và F
- Một hệ điều hành phải quản lý / kết hợp nhiều tiến trình cùng một lúc!



MỘT VÍ DỤ TIẾN TRÌNH (shell)

- **Lệnh shell**

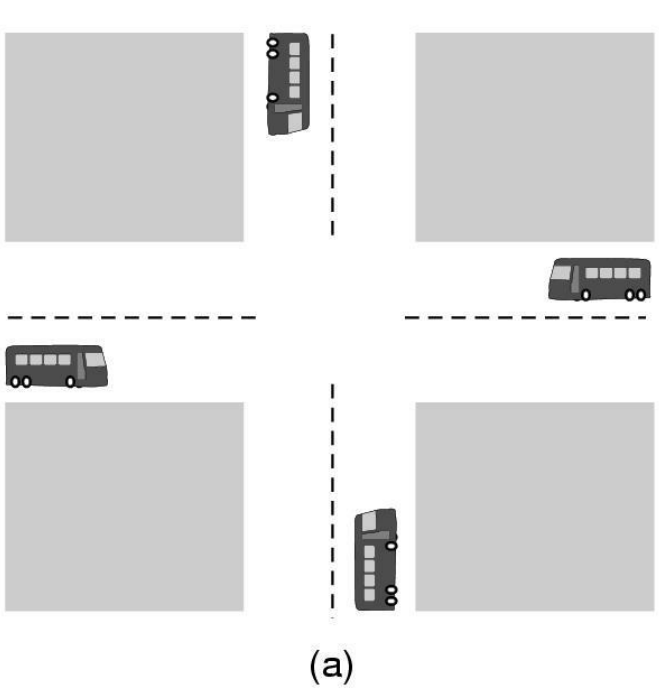
```
while (TRUE) {                                  /* repeat forever */
   type_prompt( );                              /* display prompt */
   read_command (command, parameters)           /* input from terminal */

    if (fork() != 0) {                          /* fork off child process */
     /* Parent code */
        waitpid( -1, &status, 0);               /* wait for child to exit */
    } else {
   /* Child code */
     exec (command, parameters, 0);             /* execute command */
    }
}
```



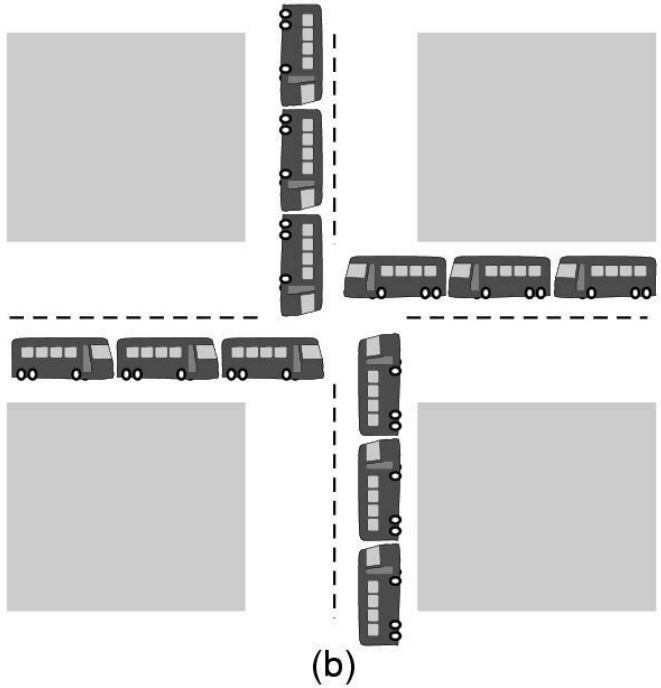
Hai tiến trình trao đổi qua đường ống

MỘT VÍ DỤ TIẾN TRÌNH (shell)

- **Bế tắc - Deadlock**



(a) Một tiềm năng deadlock

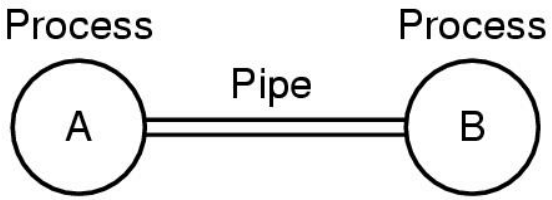


(b) Một deadlock có thực.

Một hệ điều hành phải đối phó với việc phát hiện, tránh và ngăn chặn bế tắc

MỘT VÍ DỤ TIẾN TRÌNH (shell)

- **Lệnh shell**



Hai tiến trình trao đổi qua đường ống

QUẢN LÍ CPU

Is this done in user-level or kernel-level?

- Mong muốn
  Chia sẻ thời gian
  Nhiều phân bổ CPU

- Bài toán
  Không lãng phí tài nguyên CPU
  Đồng bộ hóa và loại trừ lẫn nhau
  Công bằng
  Tránh ngẹt (Dead Free)

Analogy: Single Video Game in a house with multiple kids

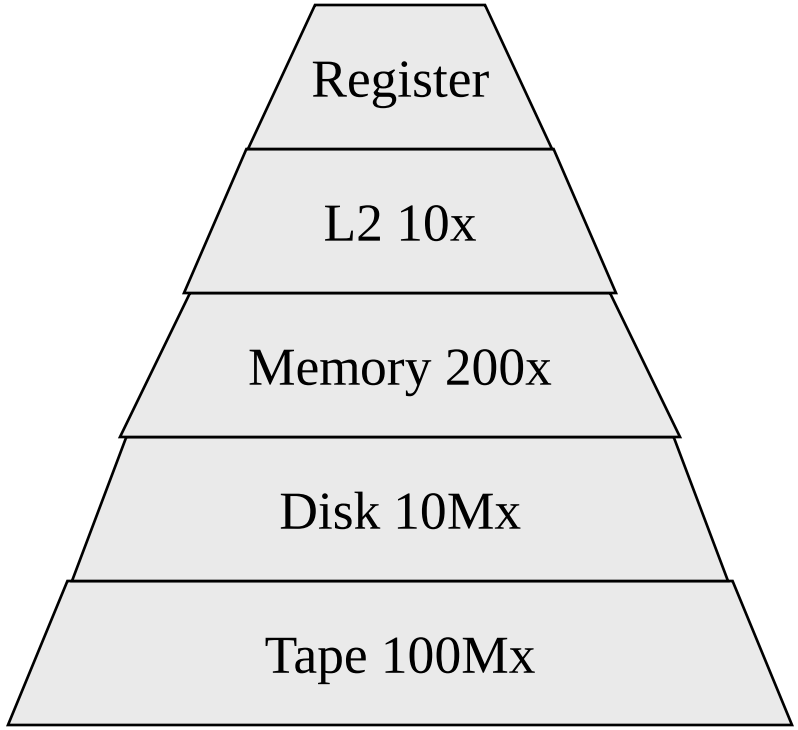
What happens if a process executes
while (1) ; ?

QUẢN LÍ BỘ NHỚ

- Mong muốn
  Hỗ trợ các chương trình để chạy
  Phân bổ và quản lý
  Chuyển từ /tới bộ nhớ thứ hai

- Bài toán
  Hiệu quả & tiện lợi
  Công bằng
  Bảo vệ

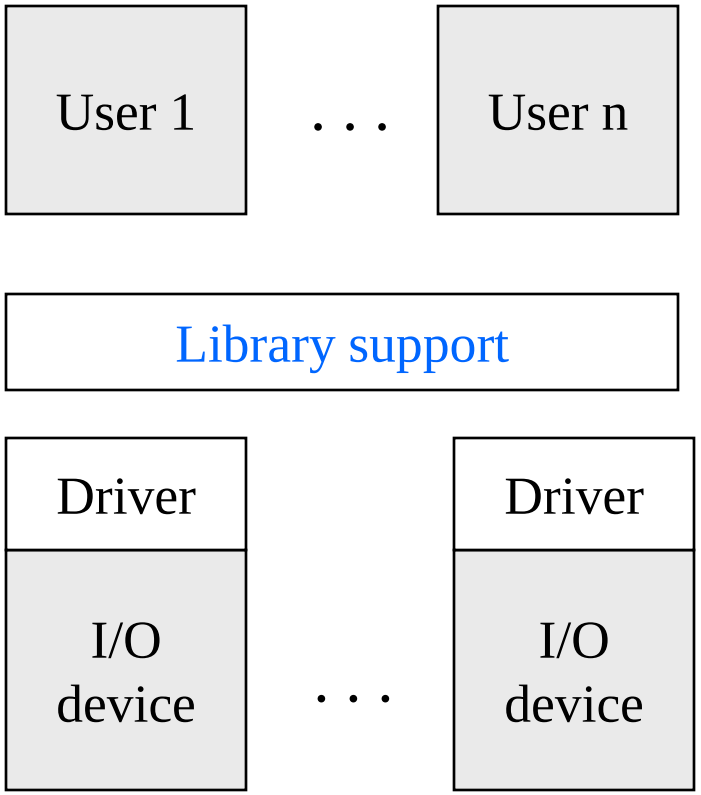
Is this done in user-level or kernel-level?



QUẢN LÍ THIẾT BỊ VÀO RA

Is this done in user-level or kernel-level?

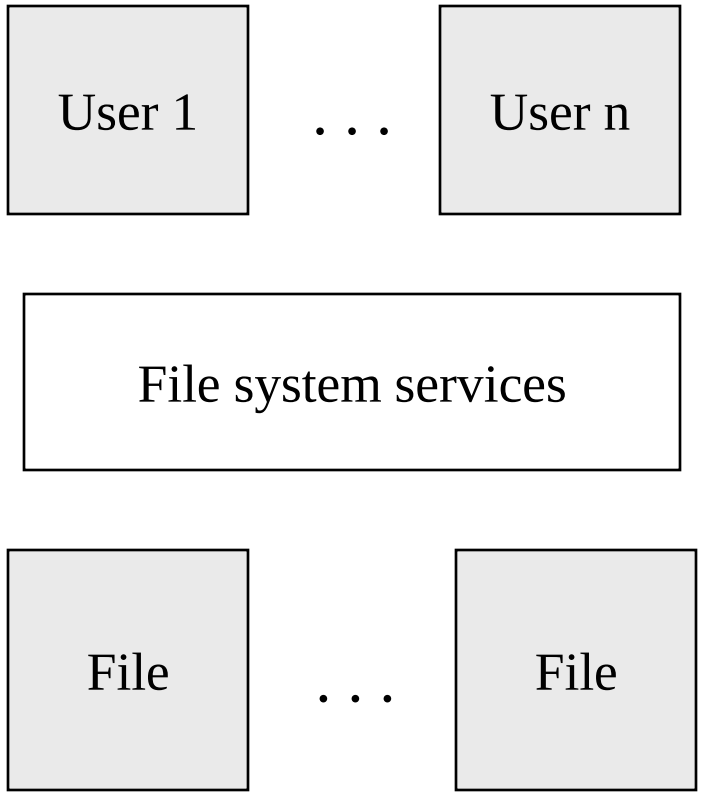
- Mong muốn
  - Tương tác giữa các thiết bị và ứng dụng
  - Có khả năng cắm và chạy các thiết bị mới
- Bài toán
  - Hiệu quả
  - Công bằng
  - Bảo vệ và chia sẻ



FILE HỆ THỐNG

Is this done in user-level or kernel-level?

- Một hệ thống tệp điển hình
  Mở tệp có xác thực
  Đọc / ghi dữ liệu trong tệp
  Đóng tệp
- Có thể chuyển một số dịch vụ này sang cấp độ người dùng không?



THIẾT LẬP TRADEOFFs

- Tất cả trong Kernel (Window)
  Ưu điểm: hiệu quả?
  Nhược điểm: khó phát triển các dịch vụ mới
- Hầu như tất cả đều ở cấp độ người dùng (Exokernel)
  Ưu điểm: dễ dàng phát triển các ứng dụng mới
  Nhược điểm: bảo vệ
- Phân chia giữa người dùng và Kernel (Unix)
  Kernel: trình điều khiển hiển thị và trình điều khiển chuột
  Người dùng: nhiều dịch vụ khác, bao gồm hệ thống tập tin mới!

TỔNG KẾT

- Tổng quan về phần cứng
- Sơ lược về Unix
- Cuộc gọi hệ thống
- Tổng quan về các thành phần hệ điều hành
- Không gian người dùng so với Không gian Kernel

Thanks