TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á
EAST ASIA UNIVERSITY OF TECHNOLOGY

Chương 2

Cấu trúc Hệ điều hành

Nội dung:

- Các dịch vụ của HĐH
- Giao diện người sử dụng của HĐH
- Lời gọi HĐH (System Calls)
- Các chương trình hệ thống
- Thiết kế và thực thi HĐH
- Cấu trúc HĐH
- Virtual Machines
- Tạo ra HĐH (Operating System Generation)

Mục tiêu

- Mô tả các dịch vụ mà một HĐH cung cấp cho các user, tiến trình và các hệ thống khác.
- Thảo luận các cách xây dựng HĐH khác nhau.
- Giải thích các HĐH được cài đặt và khởi động như thế nào.

2.1. Các dịch vụ của hệ điều hành

Một tập các dịch vụ của HĐH cung cấp các chức năng hữu dụng với các user:

- **Giao diện người sử dụng** (user interface, UI) – hầu như tất cả các HĐH có một giao diện người sử dụng
  - Khác nhau giữa giao diện dòng lệnh (Command-Line Interface, CLI), giao diện đồ họa (Graphics User Interface, GUI), Batch
- **Thực hiện chương trình** (Program execution) – khả năng của hệ thống để nạp một chương trình vào bộ nhớ và chạy nó, dừng chương trình (bình thường hoặc bất thường).
- **Thực hiện vào-ra** (I/O operations) – vì chương trình của người sử dụng không thể thực hiện trực tiếp các hoạt động vào/ra, HĐH phải cung cấp một số phương pháp để thực hiện vào/ra (1 file hoặc 1 thiết bị vào/ra).
- **Thao tác với hệ thống file** (File-system manipulation) – vì các chương trình cần đọc, ghi, tạo, xoá, tìm kiếm, liệt kê thông tin, quản lý quyền với các file và thư mục.

Các dịch vụ của hệ điều hành (tiếp)

- **Giao tiếp** (Communications) – trao đổi thông tin giữa các tiến trình đang thực hiện trên cùng 1 máy tính hoặc trên các máy tính khác nhau được nối mạng.
  - Giao tiếp có thể thông qua bộ nhớ chia sẻ (*shared memory)* hoặc chuyển thông điệp (*message passing)*: các gói tin được chuyển bởi HĐH.
- **Phát hiện lỗi** (Error detection) – HĐH cần phải thường xuyên quan tâm đến các lỗi có thể xảy ra
  - Có thể lỗi trong CPU và bộ nhớ, trong các thiết bị vào-ra, hoặc trong chương trình của người sử dụng.
  - Với mỗi loại lỗi, HĐH cần có hành động thích hợp để đảm bảo sự tính toán phù hợp và đúng đắn.
  - Tính năng sửa lỗi có thể tăng đáng kể khả năng sử dụng hiệu quả HĐH của người sử dụng và các lập trình viên.

Các dịch vụ của hệ điều hành (tiếp)

Có một tập chức năng khác của HĐH để đảm bảo sự hoạt động hiệu quả của chính nó thông qua chia sẻ tài nguyên:

- **Resource allocation** – các tài nguyên phải được phân phối cho mỗi user/tiến trình khi chúng chạy đồng thời
  - Có nhiều loại tài nguyên: một số (như các chu kỳ CPU, bộ nhớ chính, file) có thể có mã phân phối đặc biệt, số khác (như các thiết bị vào/ra) có thể có mã yêu cầu và giải phóng chung.
- **Accounting** – theo dõi và ghi lại loại tài nguyên và lượng sử dụng (tài nguyên) của user nhằm mục đích thống kê.
- **Protection & Security** – người chủ thông tin trên một hệ thống máy tính nhiều người sử dụng hoặc nối mạng có thể muốn kiểm soát sự sử dụng thông tin đó, các tiến trình đồng thời không nên can thiệp lẫn nhau
  - Protection gồm sự đảm bảo rằng tất cả sự truy nhập đến các tài nguyên hệ thống được kiểm soát.
  - Security của hệ thống từ bên ngoài yêu cầu thẩm định người sử dụng, chống các thiết bị vào/ra bên ngoài (modem, NIC) cố truy nhập không hợp lệ.

2.2. Giao diện người sử dụng của HĐH

a) **Command-Line Interface (CLI)**

CLI cho phép nhập lệnh trực tiếp để HĐH thực hiện

- Đôi khi CLI được thực thi trong kernel, đôi khi bởi chương trình hệ thống
- Đôi khi hệ thống có nhiều CLI – **shells**, vd: UNIX, LINUX
- Chức năng chính là tìm nạp 1 lệnh từ người sử dụng rồi thực hiện nó
  - Đôi khi các lệnh là built-in,
  - Đôi khi chỉ là tên của các chương trình (vd UNIX):
    - Vd lệnh **rm file.txt**
    - Nếu sau cần thêm lệnh mới thì không cần sửa đổi shell

Giao diện người sử dụng của HĐH (tiếp)

b) **Graphical User Interface (GUI)**

- Giao diện **desktop** thân thiện
  - Thường dùng chuột, bàn phím, màn hình
  - **Icons** đại diện cho các file, chương trình, hành động,…
  - Các nút chuột khác nhau trên các đối tượng gây các hành động khác nhau (cung cấp thông tin, lựa chọn, thực hiện chức năng, mở thư mục)
  - Được phát minh tại Xerox PARC những năm 1970
- Hiện nay nhiều HĐH bao gồm cả giao diện CLI và GUI
  - Microsoft Windows có GUI với CLI "command" shell
  - Apple Mac OS X có giao diện GUI là "Aqua" với UNIX kernel bên dưới và có các shell.
  - Solaris là CLI với các giao diện GUI tùy chọn (Java Desktop, KDE)

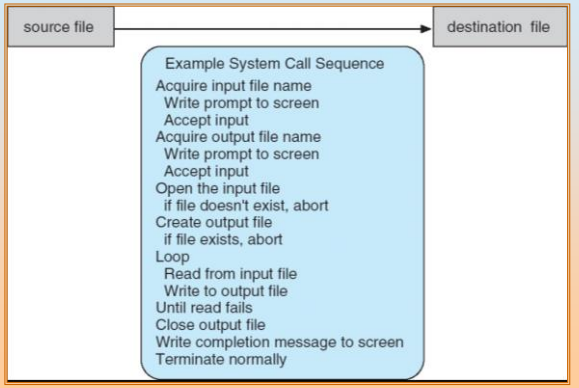
2.3. System Calls

- Cung cấp giao diện lập trình cho các dịch vụ của HĐH.
- Thường được viết bằng một ngôn ngữ bậc cao (C, C++)
- Hầu hết được truy nhập bởi các chương trình thông qua một giao diện lập trình ứng dụng (**Application Program Interface - API)** bậc cao, ít khi sử dụng trực tiếp system call.
- Ba API phổ biến nhất là Win32 API cho Windows, POSIX API cho các hệ thống trên nền POSIX (gồm hầu hết các phiên bản của UNIX, Linux, Mac OS X), và Java API cho Java virtual machine (JVM)
- Tại sao lại sử dụng các API thay vì các system call?

  (Chú ý: tên của các system-call được sử dụng ở đây là tổng quát)

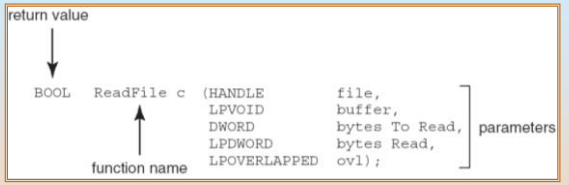
Ví dụ các System Call

- Chuỗi system call để copy nội dung của 1 file tới file khác



Ví dụ API chuẩn

- Xét hàm ReadFile() trong Win32 API – hàm đọc 1 file

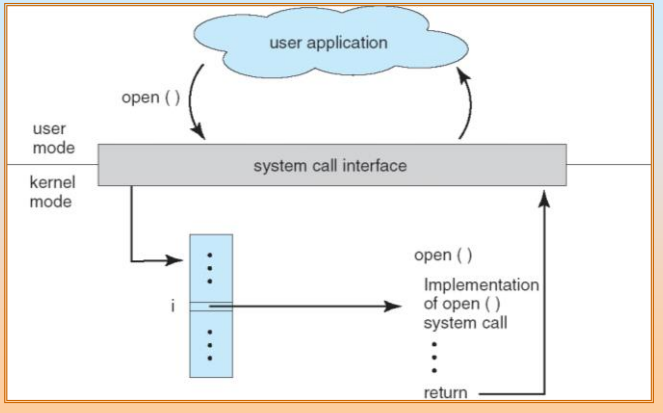


- Các tham số được truyền cho ReadFile()
  - HANDLE file – file cần đọc
  - LPVOID buffer – buffer để dữ liệu được đọc vào và ghi ra
  - DWORD bytesToRead – số byte được đọc vào buffer
  - LPDWORD bytesRead – số byte đọc được trong lần đọc trước
  - LPOVERLAPPED ovl – chỉ ra nếu sử dụng vào/ra kiểu gối chồng.

Thực thi system call

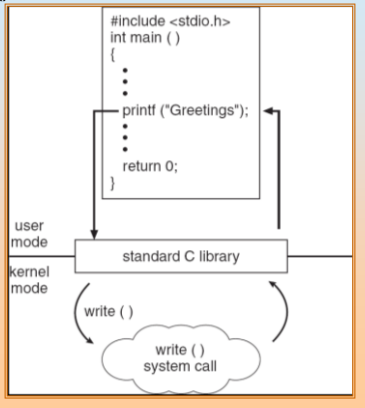
- Thường thì một số được gắn với mỗi system call
  - Giao diện system-call duy trì một bảng được đánh chỉ số theo những số này.
- Giao diện system call gọi system call mong muốn trong kernel HĐH và trả về trạng thái của nó và các giá trị trả về nào đó.
- Người gọi không cần biết gì về system call được thực thi như thế nào
  - Chỉ cần tuân thủ API và hiểu HĐH sẽ làm ra kết quả gì
  - Hầu hết giao diện HĐH ẩn đối với lập trình viên bởi API
    - Được quản lý bởi thư viện hỗ trợ tại giai đoạn chạy (tập các hàm được xây dựng vào các thư viện cùng với trình biên dịch)

Mối quan hệ API – System Call – HĐH



Ví dụ thư viện C chuẩn

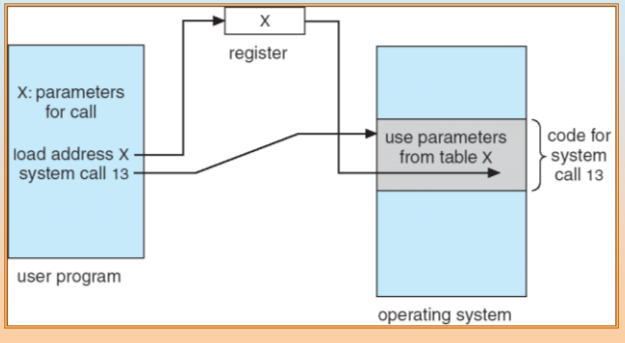
- Chương trình C gọi lời gọi thư viện printf(), mà gọi tới system call write()



Truyền tham số cho System Call

- Thường cần nhiều thông tin hơn là đơn giản chỉ xác định system call mong muốn
  - Kiểu và lượng thông tin chính xác thay đổi theo HĐH và theo lời gọi
- Ba phương thức tổng quát được sử dụng để *truyền tham số* cho HĐH.
  1. Đơn giản nhất: Truyền tham số trong các thanh ghi.
     - Trong một số trường hợp: số tham số nhiều hơn số thanh ghi
  2. Tham số được chứa trong một bảng trong bộ nhớ, và địa chỉ của bảng được truyền như một tham số trong một thanh ghi.
     - Phương pháp này được sử dụng bởi Linux và Solaris
  3. Đẩy *(push,* store) các tham số vào *stack* bằng chương trình, và lấy ra khỏi stack (*pop)* bởi HĐH.
  - Các phương pháp dùng bảng và stack không giới hạn số lượng hay độ dài của các tham số được truyền.

Truyền tham số thông qua Bảng



Các loại System Calls

- Điều khiển tiến trình (Process control)
  - kết thúc, bỏ dở (abort)
  - nạp, thực hiện
  - tạo, chấm dứt tiến trình
  - lấy, thiết lập các thuộc tính của tiến trình
  - chờ đợi
  - đợi sự kiện, báo hiệu sự kiện
  - phân phối và giải phóng bộ nhớ

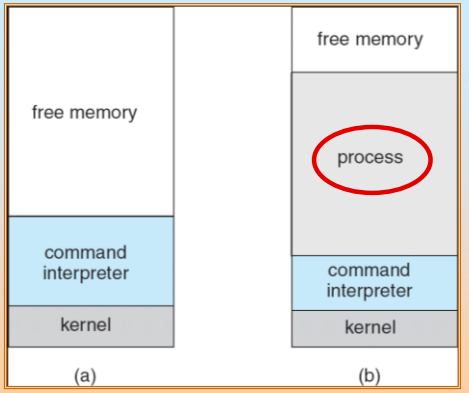
Các loại System Calls (tiếp)

- Quản lý file (File management)
  - tạo file, xóa file
  - mở, đóng
  - đọc, ghi, định vị
  - lấy/ thiết lập thuộc tính file
- Quản lý thiết bị (Device management)
  - yêu cầu thiết bị, giải phóng thiết bị
  - đọc, ghi, định vị
  - lấy/ thiết lập các thuộc tính thiết bị
  - gắn kết (attack), tháo gỡ (detach) logic các thiết bị

Các loại System Calls (tiếp)

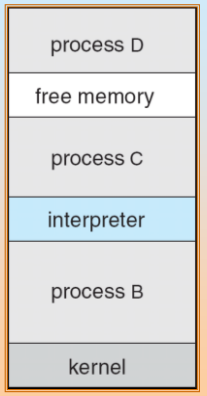
- Duy trì thông tin (Information maintenance)
  - lấy/ thiết lập giờ hoặc ngày
  - lấy/ thiết lập dữ liệu hệ thống
  - lấy/ thiết lập thuộc tính của tiến trình, file, thiết bị
- Giao tiếp (Communications)
  - tạo, xóa kết nối giao tiếp
  - gửi, nhận thông điệp
  - truyền thông tin trạng thái
  - gắn kết, tháo gỡ logic các thiết bị ở xa (remote device)

MS-DOS



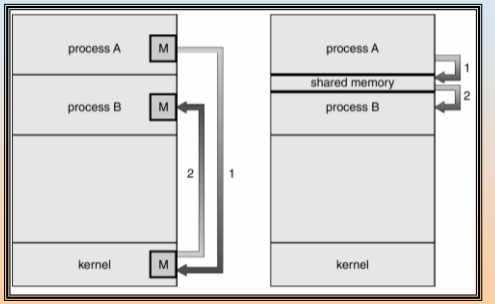
At System Start-up Running a Program

UNIX



Các phương thức giao tiếp

- Sự giao tiếp có thể thực hiện bằng cách sử dụng phương thức message passing hoặc shared memory.



Message Passing Shared Memory

2.4. Các chương trình hệ thống

- Các chương trình hệ thống cung cấp một môi trường thuận tiện cho việc thực hiện và phát triển chương trình. Chúng có thể được phân loại thành:
  - Thao tác với file: tạo, xóa, copy, đổi tên… các file và thư mục
  - Thông tin trạng thái: ngày giờ, dung lượng bộ nhớ trống, số user…
  - Sửa đổi file: trình soạn thảo văn bản có thể tạo, sửa nội dung file trên đĩa
  - Hỗ trợ ngôn ngữ lập trình: trình biên dịch, trình thông dịch, trình gỡ lỗi…
  - Nạp và thực hiện chương trình: nạp CT đã được biên dịch vào bộ nhớ để thực hiện
  - Giao tiếp: cung cấp cơ chế tạo kết nối ảo giữa các tiến trình, các user, các máy tính để gửi message, duyệt web, gửi email, truyền file…
- Hầu hết cách nhìn nhận của người sử dụng về HĐH được xác định bởi các chương trình hệ thống, không thực sự bởi các system call.

2.5. Thiết kế và thực thi HĐH

- Thiết kế và thực thi HĐH không có giải pháp hoàn hảo, nhưng một số phương pháp đã chứng minh thành công
- Cấu trúc bên trong của các HĐH khác nhau có thể rất khác nhau
- Bắt đầu từ việc xác định các mục tiêu và đặc điểm
- Bị tác động bởi sự lựa chọn phần cứng, loại HĐH: chia sẻ thời gian, đơn người dùng, đa người dùng, phân tán, thời gian thực…
- *User* goals và *System* goals
  - User goals – HĐH cần dễ sử dụng, dễ học, đáng tin cậy, an toàn, nhanh.
  - System goals – HĐH cần dễ thiết kế, thực thi và duy trì, cũng như linh hoạt, đáng tin cậy, không có lỗi, hiệu quả.

Thiết kế và thực thi HĐH (tiếp)

- Nguyên lý quan trọng là sự tách biệt:

  Policy (chính sách): *Cái gì* sẽ được làm? - What
  Mechanism (cơ chế): Làm nó *như thế nào*? - How

  - Sự tách biệt chính sách với cơ chế cho phép sự linh hoạt tối đa nếu sau này các quyết định chính sách được thay đổi
  - VD: Các HĐH vi nhân (như UNIX, Solaris) tách biệt cơ chế và chính sách bằng cách thực thi một tập cơ bản các khối tạo dựng ban đầu, hầu như độc lập với chính sách; cho phép các cơ chế và chính sách tiên tiến hơn có thể được thêm vào thông qua các môđun kernel do người sử dụng tạo hoặc do chính chương trình của người sử dụng. Trong phiên bản mới nhất của Solaris, tùy vào bảng nào được nạp, hệ thống có thể là chia sẻ thời gian, xử lý theo lô, thời gian thực, chia sẻ công bằng, hay dạng kết hợp bất kỳ.

Thực thi hệ thống

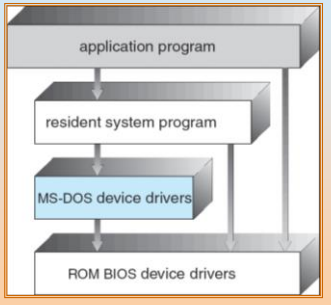
- Trước đây, HĐH được viết bằng ngôn ngữ assembly, hiện nay có thể viết bằng những ngôn ngữ bậc cao (UNIX, PS/2, Windows NT chủ yếu viết bằng C).
- Mã được viết bằng ngôn ngữ bậc cao:
  - có thể viết nhanh hơn.
  - cô đọng hơn.
  - dễ hiểu và dễ gỡ rối.
- Một HĐH được viết bằng một ngôn ngữ bậc cao sẽ dễ dàng hơn khi chuyển sang phần cứng mới.

2.6. Cấu trúc hệ điều hành

a) Cấu trúc đơn giản

Hệ điều hành MS-DOS

- MS-DOS – được viết để cung cấp hầu hết các chức năng trong một không gian nhỏ nhất
  - Không chia thành các module
  - Dù MS-DOS có một vài cấu trúc, giao diện của nó và các mức chức năng không được phân định rõ ràng.

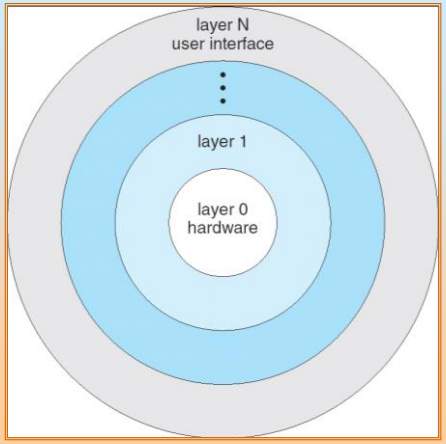


Cấu trúc lớp của MS-DOS

b) Phương pháp phân lớp

- HĐH được chia thành các lớp (layer, level), mỗi lớp được xây dựng trên đỉnh của các lớp thấp hơn. Lớp ở đáy (layer 0) là phần cứng; lớp cao nhất (layer N) là user interface.
- Bằng cách chia thành các lớp như trên, mỗi lớp chỉ sử dụng các chức năng và dịch vụ của các lớp dưới.

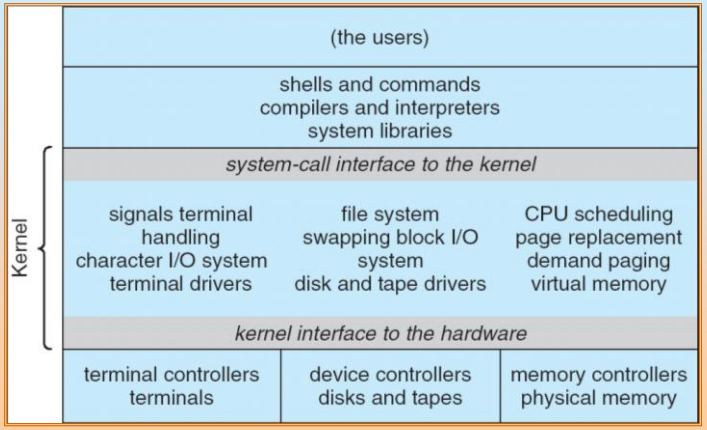
HĐH được phân lớp



Cấu trúc HĐH UNIX

- UNIX – là HĐH khác mà ban đầu đã bị hạn chế bởi chức năng phần cứng.
- HĐH UNIX bao gồm 2 phần riêng biệt:
  - Systems programs – các chương trình hệ thống
  - The kernel - nhân
    - Bao gồm tất cả các lớp nằm dưới giao diện system-call và nằm trên physical hardware
    - Cung cấp hệ thống file, lập lịch CPU, quản lý bộ nhớ và các chức năng HĐH khác; rất nhiều chức năng cho 1 mức.

Cấu trúc lớp HĐH UNIX (tiếp)



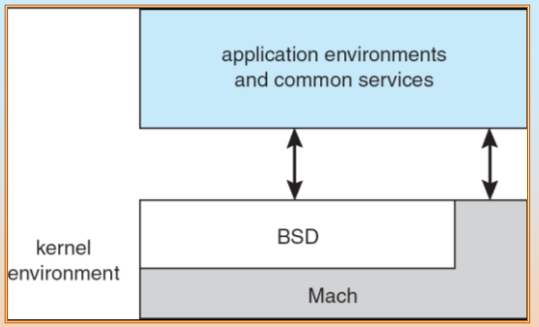
c) Cấu trúc vi nhân (Microkernel)

- Vd: Windows XP
- Chuyển rất nhiều thành phần không thiết yếu từ kernel vào trong user space ⇒ microkernel
- Sự giao tiếp diễn ra giữa các module của người sử dụng bằng phương thức message passing.
- Các lợi điểm:

  - dễ dàng mở rộng hệ điều hành mà không phải thay đổi kernel
  - dễ dàng mang một HĐH đặt vào những kiến trúc khác
  - đáng tin cậy hơn (ít mã lệnh chạy trong kernel mode)
  - an toàn hơn (ít thứ phải bảo vệ hơn)

- Nhược điểm: có thể làm giảm hiệu năng vì quá tải giao tiếp từ user space tới kernel space.

Cấu trúc vi nhân lai của Mac OS X

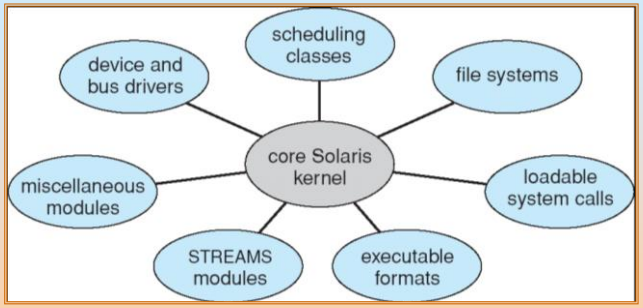


BSD: Berkeley Software Distribution. Mac OS X dựa trên hệ điều hành Darwin, một hệ điều hành cũng dựa trên BSD. Trong khi hạt nhân ở mức độ thấp và các phần mềm khác là mã nguồn mở BSD, hầu hết các phần còn lại của hệ điều hành Mac OS là mã nguồn đóng. Apple được xây dựng Mac OS X và iOS trên top của BSD vì vậy họ sẽ không phải viết hệ điều hành cấp thấp, cũng giống như Google Android được xây dựng trên top của Linux.

d) Modules

- Hầu hết các HĐH hiện đại thực thi các kernel module:
  - Sử dụng phương pháp hướng đối tượng
  - Mỗi thành phần hạt nhân là tách biệt
  - Mỗi thành phần giao tiếp với các thành phần khác qua giao diện đã định trước
  - Mỗi thành phần có thể được nạp vào trong kernel khi cần thiết
- Tổng quát: tương tự như các lớp nhưng phức tạp hơn.

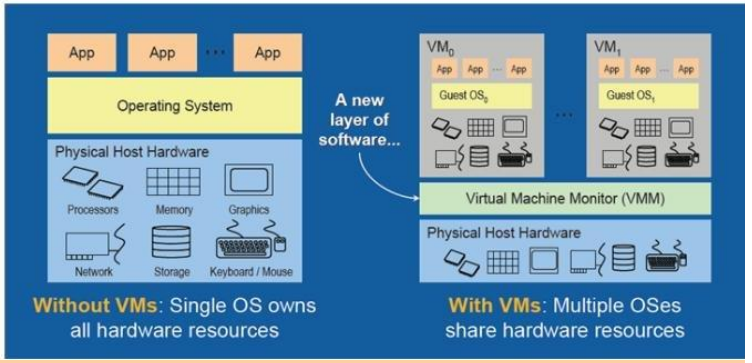
Cấu trúc môđun của HĐH Solaris



2.7. Virtual Machines

- Một *máy ảo* (*virtual machine)* là một chương trình giả lập phần cứng (hardware simulator). Chạy N bản copy của chương trình giả lập này, một máy vật lý trở thành N máy ảo.
- Mỗi máy ảo có thể chạy:
  - một tiến trình đơn dưới một HĐH đơn
  - tất cả tiến trình của một user dưới một HĐH
  - một HĐH chia sẻ thời gian phức tạp (vd: để gỡ rối)
- "HĐH" (Virtual Machine) có 3 phần:
  - Trình giả lập phần cứng - hardware simulator,
  - Tài nguyên (processor, memory) chia sẻ giữa các trình giả lập,
  - HĐH chạy trong mỗi trình giả lập.

Các mô hình hệ thống VM và non-VM

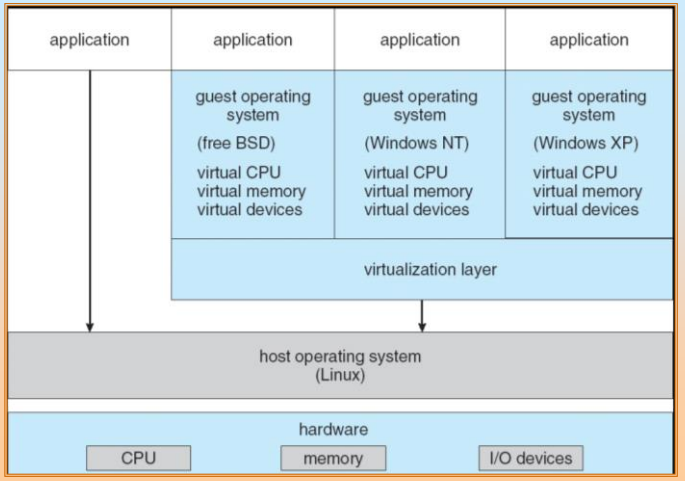


Non-virtual Machine Virtual Machine

Các lợi điểm của Virtual Machine

- Virtual-machine cung cấp sự bảo vệ hoàn toàn các tài nguyên hệ thống vì mỗi máy ảo được tách biệt với các máy ảo khác.
- Rất lý tưởng cho việc nghiên cứu và phát triển các HĐH. Sự phát triển hệ thống ảo không phá vỡ sự hoạt động của hệ thống thật.

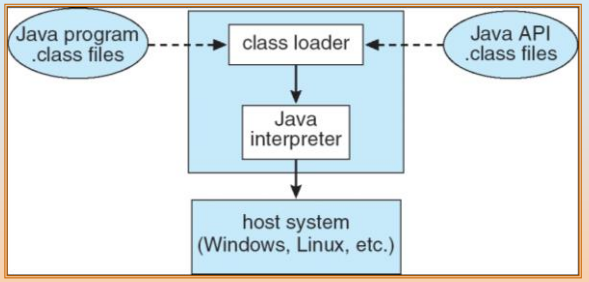
Kiến trúc VMware



Java Virtual Machine

- Các chương trình Java sau khi biên dịch thành các tệp bytecode có tính trung lập nền (platform-neutral bytecode, có tên mở rộng .class), và được thực hiện bởi Java Virtual Machine (JVM).
- JVM bao gồm:
  - trình nạp lớp (class loader)
  - trình xác định lớp (class verifier)
  - trình thông dịch thời gian chạy (runtime interpreter)
- Trình thông dịch Java có thể là:
  - môđun phần mềm thông dịch các bytecode chỉ 1 lần.
  - Just-In-Time (JIT) compiler chuyển các bytecode thành ngôn ngữ máy tự nhiên → làm tăng hiệu năng.

Java Virtual Machine (tiếp)



2.8. System Generation (SYSGEN)

Hệ điều hành được xây dựng dựa trên nguyên tắc nào ?

- Các HĐH được thiết kế để chạy trên bất kỳ loại máy nào; sau đó hệ thống phải được cấu hình cho mỗi máy tính cụ thể. Tiến trình đó được gọi là System generation.
- HĐH thường được phân phối trên các đĩa DVD. Để tạo ra 1 HĐH, chúng ta sử dụng 1 chương trình đặc biệt - SYSGEN.
- Chương trình SYSGEN xác định thông tin liên quan đến cấu hình riêng của hệ thống phần cứng từ 1 file hoặc yêu cầu người sử dụng cung cấp:
  - Sử dụng CPU nào ? Dung lượng bộ nhớ khả dụng ?
  - Thông tin về các thiết bị khả dụng ?
  - Các lựa chọn HĐH nào được yêu cầu ? Những giá trị tham số nào được sử dụng ?

SYSGEN (tiếp)

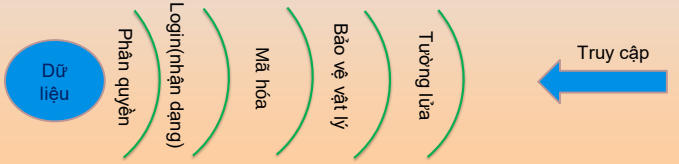
- Các thông tin trên sau đó thường được System generation sử dụng để tạo các bảng thích hợp mô tả hệ thống và để sinh ra hệ thống.
- Sau khi hệ thống được sinh ra, nó phải được phần cứng sử dụng → làm sao để phần cứng biết nơi chứa nhân HĐH (kernel), nạp như thế nào ?
- *Booting* – quá trình khởi động máy tính bằng cách nạp nhân.
- *Bootstrap program* – đoạn mã được chứa trong ROM của hầu hết các hệ thống máy tính để có thể xác định vị trí của nhân, nạp nó vào bộ nhớ, và bắt đầu sự thực hiện của nó.

SYSGEN (tiếp)

- **Nguyên tắc bảo vệ nhiều mức**

+ Tài nguyên có đặc trưng khác nhau ➔ phương pháp bảo vệ khác nhau

+ Nhiều lớp bảo vệ ➔ an toàn



BÀI TẬP

1. Cài đặt VMware Workstation Pro 15/17.
2. Thiết lập cấu hình cho 1 máy ảo và Cài đặt HĐH Linux (Linux Mint/Ubuntu).

   Hoặc cài đặt HĐH Windows XP/7/8/10/Server 2012/2016

a) Link:

https://drive.google.com/drive/folders/1jg7_pNOeqhpiWyAGr3QQERsa0hlzfF5G

b) Link – Linux Mint 20.2: https://linuxmint.com/edition.php?id=288

c) Link – Ubuntu 20.04.3 LTS: https://ubuntu.com/download/desktop

d) Link – CentOS 8.4.2105:

http://centos-hcm.viettelidc.com.vn/8.4.2105/isos/x86_64/

d) Link – Fedora 35 Workstation:

https://getfedora.org/en/workstation/download/

BÀI TẬP

3. Nhiệm vụ của HĐH trong việc quản lý tiến trình ?
4. Nhiệm vụ của HĐH trong việc quản lý bộ nhớ ?
5. Nhiệm vụ của HĐH trong việc quản lý tập tin ?
6. Nhiệm vụ của HĐH trong việc quản lý vào/ra ?
7. Cấu trúc của hệ thống MS- DOS ?