HỆ ĐIỀU HÀNH

CHƯƠNG 1

TỔNG QUAN HỆ ĐIỀU HÀNH

thanhlv (thanh.cntt.dhv@gmail.com)

Nghệ An, 2023

Nội dung giảng dạy

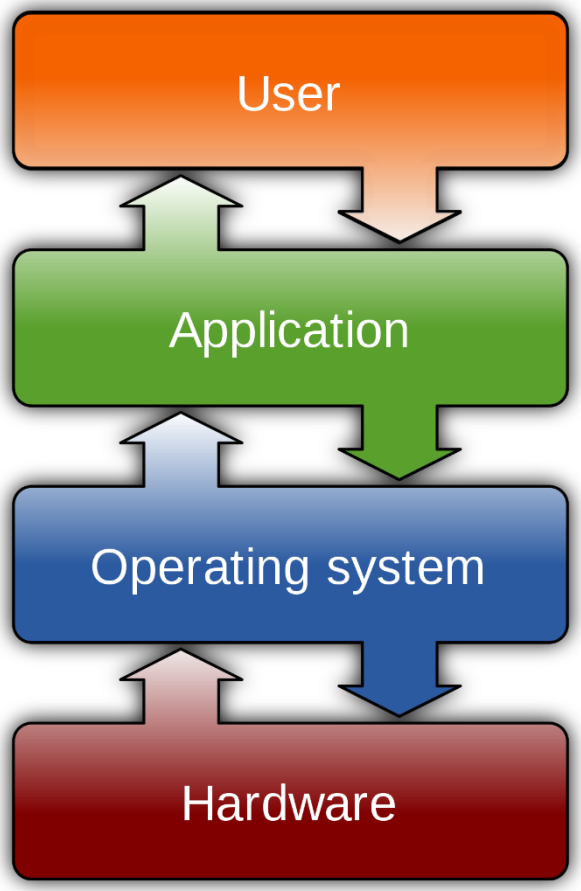
- 1.1. Khái niệm
- 1.2. Phân loại
- 1.3. Ảo hóa
- 1.4. Điện toán đám mây
- 1.5. Các thành phần hệ điều hành
- 1.6. Các dịch vụ của hệ điều hành
- 1.7. System Call
- 1.8. Kiến trúc hệ điều hành

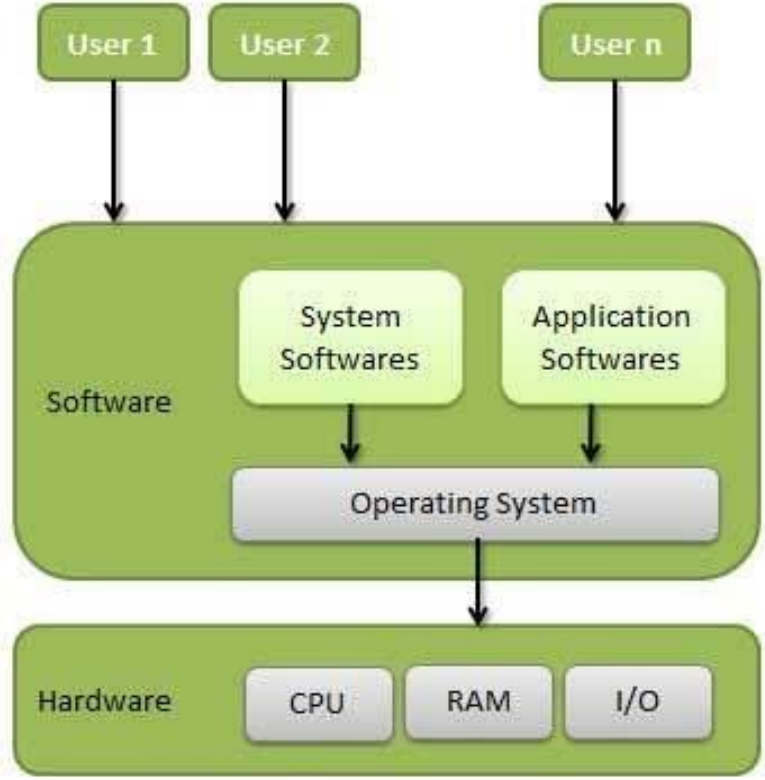
1.1. Khái niệm

- Là một chương trình quản lý tài nguyên của máy tính, đóng vai trò như một lớp trung gian giữa người sử dụng máy tính và phần cứng của máy tính
- Mục tiêu:
  - Cung cấp phương tiện giao tiếp giữa người dùng và máy tính
  - Nhận và thực thi các yêu cầu của người dùng một cách hiệu quả, nhanh chóng và dễ dàng thông qua các chương trình ứng dụng
  - Quản lý và sử dụng tài nguyên máy tính một cách hiệu quả
- Bao gồm mã nguồn, dữ liệu, các tham số hệ thống,… đã được cài đặt sẵn trên hệ thống máy tính
  - Người dùng muốn giao tiếp với hệ điều hành → cần phải có ngôn ngữ chung:
    - Giao tiếp dòng lệnh: thể hiện yêu cầu thông qua các lệnh
    - Giao tiếp biểu tượng: chọn các biểu tượng để thực hiện yêu cầu

1.1. Khái niệm

- Vị trí của hệ điều hành





1.1. Khái niệm

- Tính chất cơ bản của hệ điều hành:
  - Độ tin cậy cao: Mọi hoạt động, thông báo của hệ điều hành phải chuẩn xác
  - An toàn: Dữ liệu và chương trình phải được bảo vệ
    - Không bị thay đổi ngoài ý muốn trong mọi chế độ làm việc
    - Hạn chế truy nhập bất hợp pháp
  - Hiệu quả: Các tài nguyên phải được khai thác triệt để
  - Thuận tiện:
    - Dễ sử dụng
    - Có hệ thống trợ giúp phong phú

1.1. Khái niệm

- Hệ điều hành Windows

Windows 1.0 (1985)

Windows NT (1993)

Windows 95/98 (1995)

Windows 2000 (1999)

Windows XP (2004)

Windows Vista (2005)

Windows 7 (2009)

Windows 8 (2012)

1.1. Khái niệm

- Hệ điều hành Linux

1.1. Khái niệm

- Hệ điều hành Android

Cupcake
Android 1.5

Donut
Android 1.6

Eclair
Android 2.0/2.1

Froyo
Android 2.2.x

Gingerbread
Android 2.3.x

Honeycomb
Android 3.x

Ice Cream Sandwich
Android 4.0.x

Jelly Bean
Android 4.1.x

KitKat
Android 4.4.x

Lollipop
Android 5.0

1.1. Khái niệm

- Hệ điều hành Android

Android 8.0

Android 7.0 Nougat

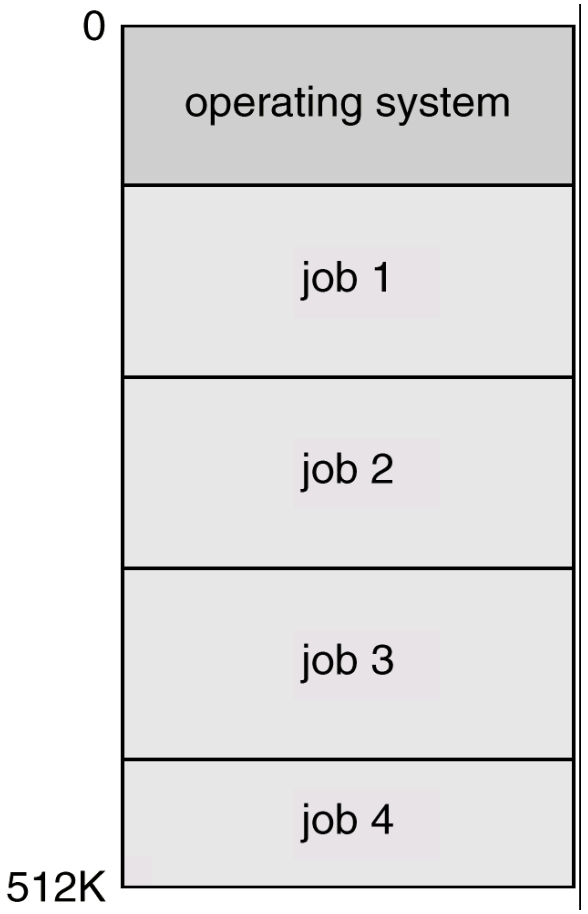
Android 10

1.2. Phân loại

- Hệ thống xử lý theo lô
  - Thực hiện các chương trình lần lượt theo những chỉ thị đã được xác định trước
  - Khi một chương trình kết thúc, hệ thống tự động thực hiện chương trình tiếp theo mà không cần sự can thiệp từ bên ngoài
  - Toàn bộ hệ thống máy tính phục vụ 1 chương trình từ lúc bắt đầu khi chương trình được đưa vào bộ nhớ đến khi kết thúc chương trình
  - Khi 1 chương trình được đưa vào bộ nhớ và thực hiện → nó chiếm giữ mọi tài nguyên hệ thống nên không thể đưa chương trình khác vào bộ nhớ
  - Vấn đề: khi chương trình truy nhập thiết bị vào/ra, processor rơi vào trạng thái chờ đợi

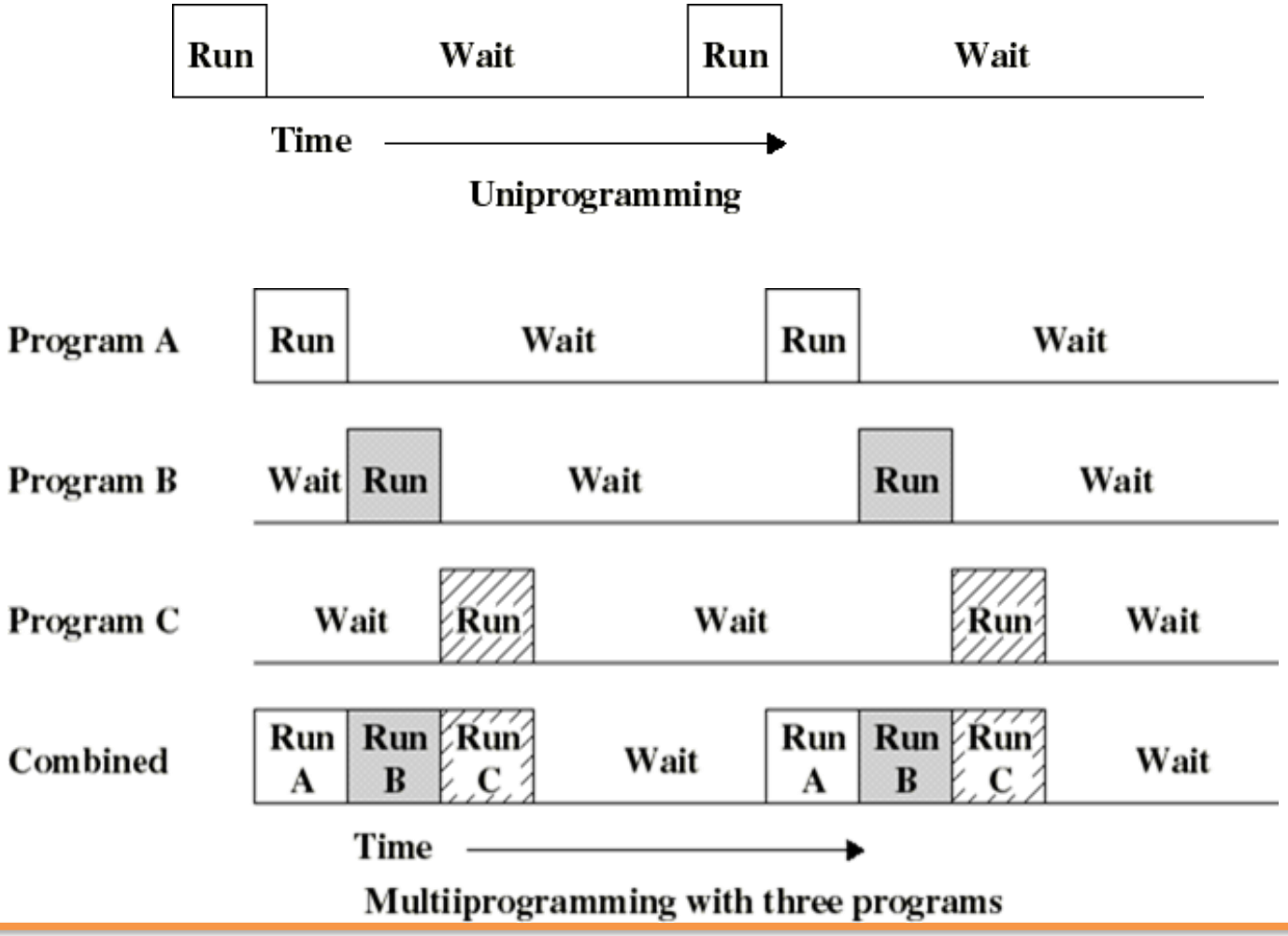
1.2. Phân loại

- Hệ thống đa chương trình (multiprogramming)
  - Tại một thời điểm có thể có nhiều chương trình có mặt đồng thời trong bộ nhớ
  - Một chương trình được thực hiện và chiếm giữ CPU cho đến khi (1) có yêu cầu vào/ra, hoặc (2) kết thúc
  - Khi (1) hoặc (2) xảy ra, chương trình khác sẽ được thực hiện
  - Ưu điểm: Tận dụng thời gian rỗi của CPU



1.2. Phân loại

- Hệ thống đa chương trình (multiprogramming)

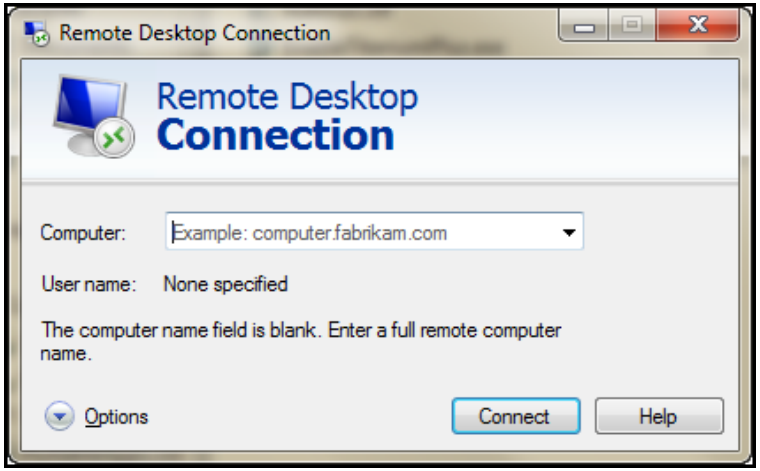


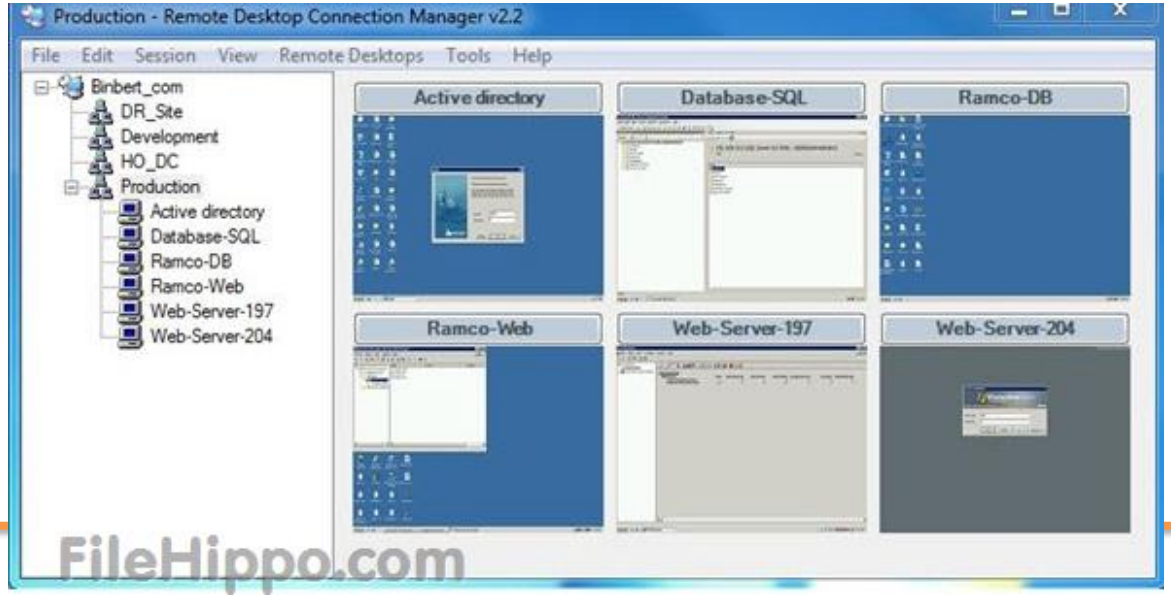
1.2. Phân loại

- Hệ thống phân chia thời gian (time sharing):
  - Nhằm tăng hiệu suất sử dụng các tài nguyên trong hệ thống
  - Cho phép nhiều người dùng chia sẻ (sử dụng) máy tính tại 1 thời điểm bằng cách phân chia thời gian sử dụng các tài nguyên
  - CPU luân phiên thực thi giữa các công việc
    - Mỗi công việc được chia một phần nhỏ thời gian CPU (time slice, quantum time)
    - Cung cấp tương tác giữa user và hệ thống với thời gian đáp ứng (response time) nhỏ (ms → s)
  - Một công việc chỉ được chiếm CPU khi nó nằm trong bộ nhớ chính
  - Khi cần thiết, một công việc nào đó có thể được chuyển từ bộ nhớ chính ra thiết bị lưu trữ, nhường bộ nhớ chính cho công việc khác
  - Bộ nhớ ảo (Virtual memory) cho phép thực thi một tiến trình không hoàn toàn nằm trong bộ nhớ

1.2. Phân loại

- Hệ thống phân chia thời gian (time sharing – lệnh mstsc):



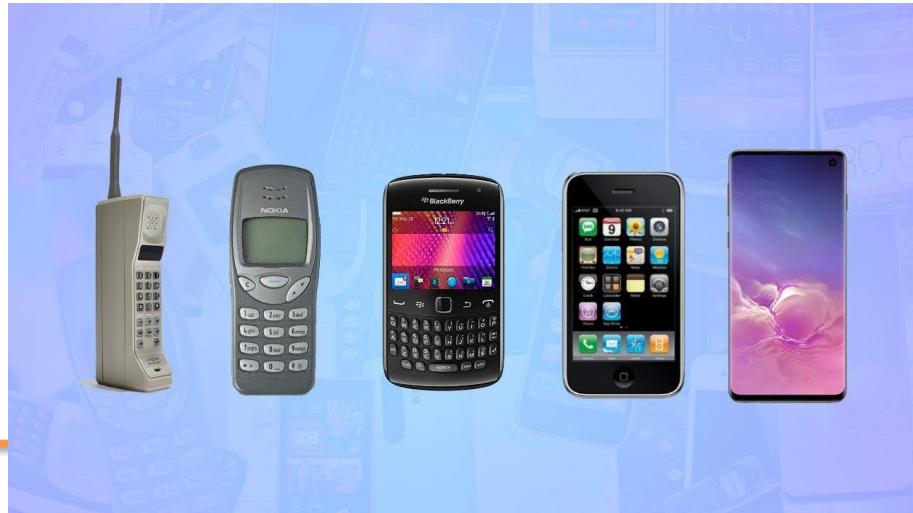


1.2. Phân loại

- Hệ thống để bàn (desktop)
  - Máy tính cá nhân (personal computer): hệ thống máy tính được dành cho một người dùng
  - Các thiết bị xuất/nhập: bàn phím, chuột, màn hình, máy in
  - Tiện lợi và phản ứng nhanh đối với người dùng
  - Hoạt động dựa theo các kỹ thuật được phát triển cho các hệ thống lớn
  - Có thể chạy nhiều họ hệ điều hành khác nhau (Windows, Linux, MacOS,…)

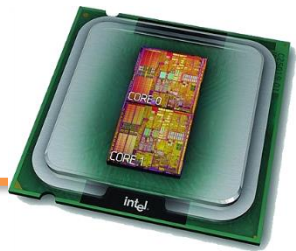
1.2. Phân loại

- Hệ thống cầm tay (mobile)
  - Có kích thước nhỏ, nhẹ, bộ nhớ và CPU có khả năng giới hạn
  - Có các tính năng chuyên biệt (nghe/ gọi, nhắn tin, GPS,…)
  - Có khả năng hoạt động liên tục không ngừng nghỉ để đảm bảo liên lạc, kết nối luôn thông suốt
  - Hệ điều hành dành cho hệ thống cầm tay: iOS, Android, Windows Phone,…



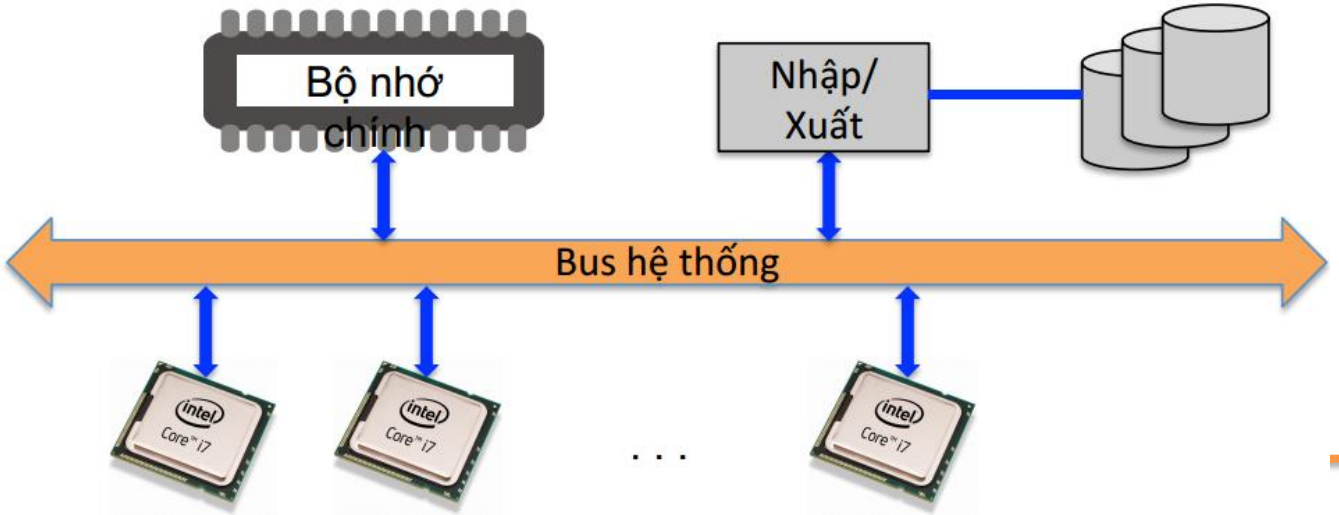
1.2. Phân loại

- Hệ thống song song (hệ thống đa xử lý - multi processor)
  - Là các hệ thống đa xử lý với nhiều CPU được kết nối chặt chẽ với nhau
  - Các processors chia sẻ bộ nhớ, thiết bị ngoại vi; việc giao tiếp giữa các processors diễn ra thông qua bộ nhớ được chia sẻ
  - Lợi ích của hệ thống song song:
    - Tăng năng lực xử lý: nhiều công việc được hoàn thành/đơn vị thời gian
    - Kinh tế: chia sẻ ngoại vi, thiết bị lưu trữ,…
    - Độ tin cậy cao: hỏng một VXL không ảnh hưởng đến hệ thống



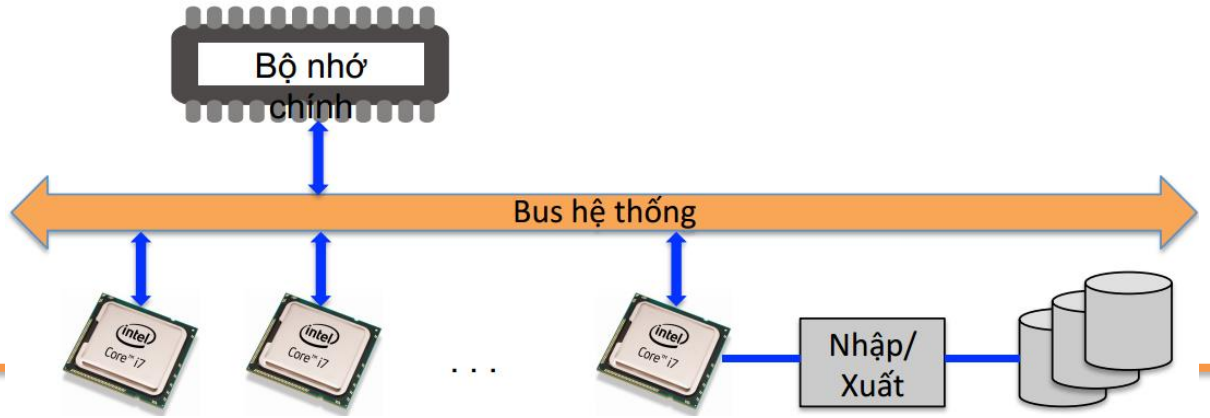
1.2. Phân loại

- Hệ thống song song: Đa xử lý đối xứng (Symmetric MultiProcessing – SMP)
  - Mỗi bộ xử lý chạy một tiến trình/tiểu trình
  - Các VXL giao tiếp với nhau thông qua một bộ nhớ dùng chung
  - Có cơ chế chịu lỗi và khả năng cân bằng tải tối ưu
  - Hầu hết các hệ điều hành hiện đại đều hỗ trợ SMP



1.2. Phân loại

- Hệ thống song song: Đa xử lý bất đối xứng (Asymmetric multiprocessing – AMP)
  - Một bộ xử lý chính kiểm soát toàn bộ hệ thống
  - Các bộ xử lý khác thực hiện theo lệnh của bộ xử lý chính hoặc theo những chỉ thị đã được định nghĩa trước
  - Mô hình này theo dạng quan hệ chủ tớ: Bộ xử lý chính sẽ lập lịch cho các bộ xử lý khác
  - Thường phổ biến trong các hệ thống cực lớn



1.2. Phân loại

- Hệ thống phân tán (distributed):
  - Mỗi bộ xử lý có bộ nhớ cục bộ riêng, và trao đổi với nhau thông qua các đường truyền thông
  - Các VXL thường khác nhau về kích thước và chức năng (máy cá nhân, máy trạm...)
  - Hệ thống phân tán được sử dụng để:
    - Chia sẻ tài nguyên: cung cấp cơ chế để chia sẻ tập tin, in ấn tại một vị trí xa,...
    - Tăng tốc độ tính toán: một thao tác tính toán được chia làm nhiều phần nhỏ được thực hiện một lúc trên nhiều vị trí khác nhau
    - An toàn: một vị trí trong hệ thống bị hỏng, các vị trí khác vẫn tiếp tục làm việc

1.2. Phân loại

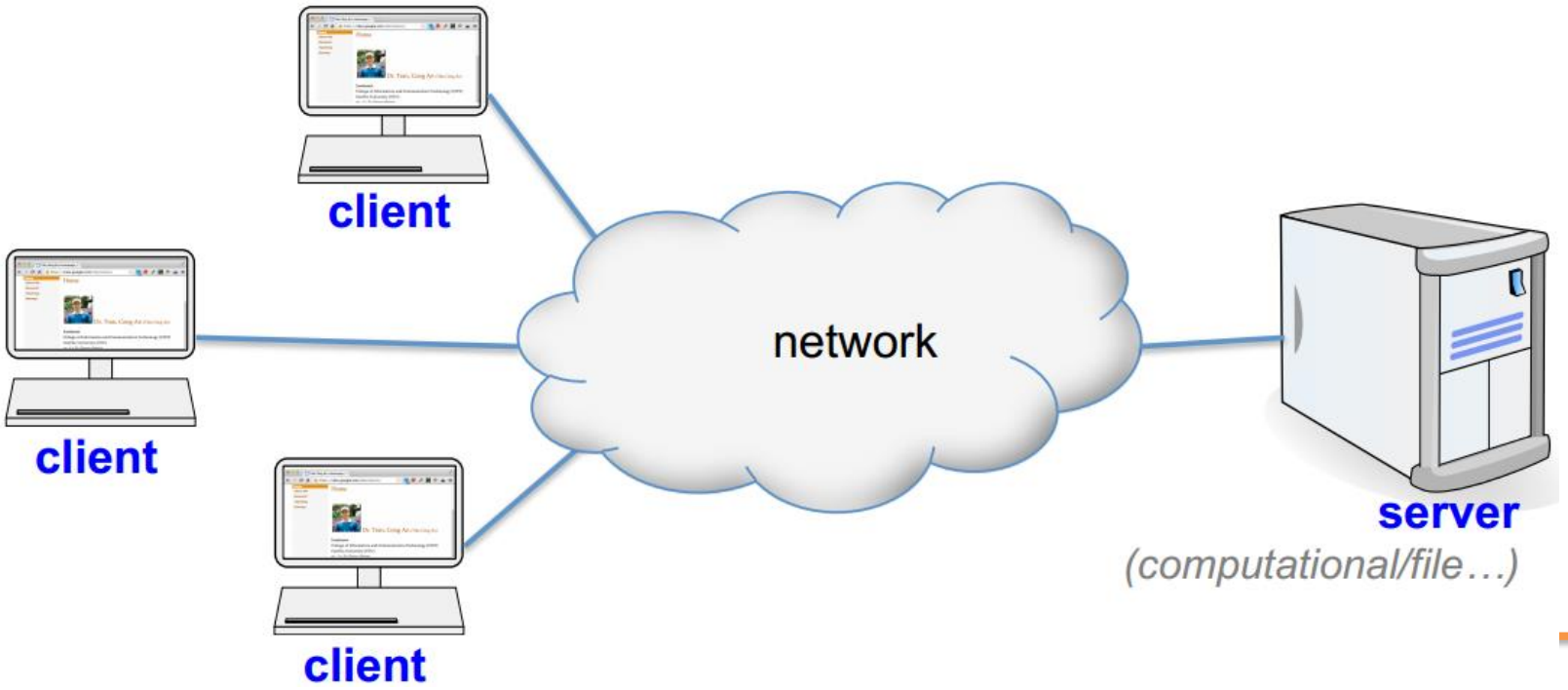
- Hệ thống phân tán (distributed):
  - Hệ điều hành phân tán (Distributed OS) điều khiển các hệ thống máy tính khác nhau trong hệ thống để có thể thực hiện các chức năng chung của hệ thống
    - Cho phép các hệ thống máy tính trao đổi các thông điệp với nhau
    - Gây cảm giác như là một hệ thống duy nhất
  - Một số hệ điều hành phân tán điển hình như AMOEBA, CHORUS,…

amoebaOS

Chorus

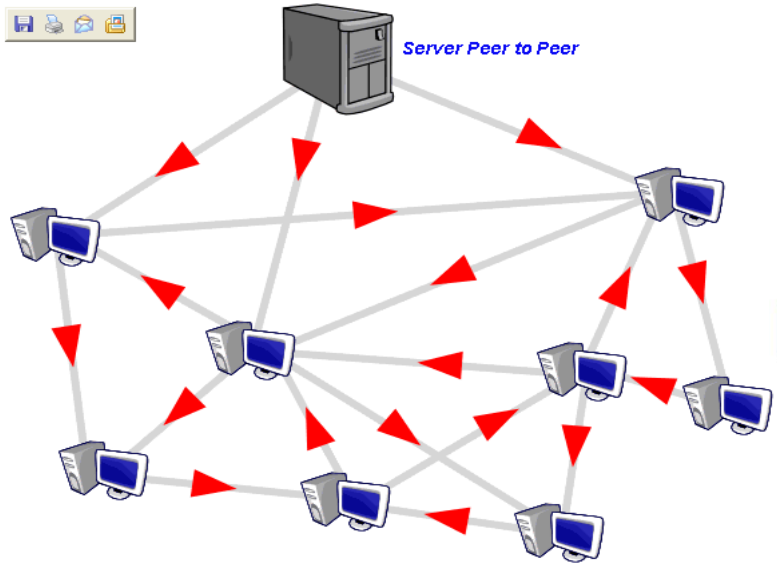
1.2. Phân loại

- Hệ thống phân tán: Client – Server
  - Máy con (Client - đóng vai trò là máy khách) gửi một yêu cầu (request) thông qua một giao thức (WWW, FTP, Telnet, email,…) đến máy chủ (Server - đóng vai trò máy cung cấp dịch vụ), máy chủ sẽ xử lý và trả kết quả về cho máy khách

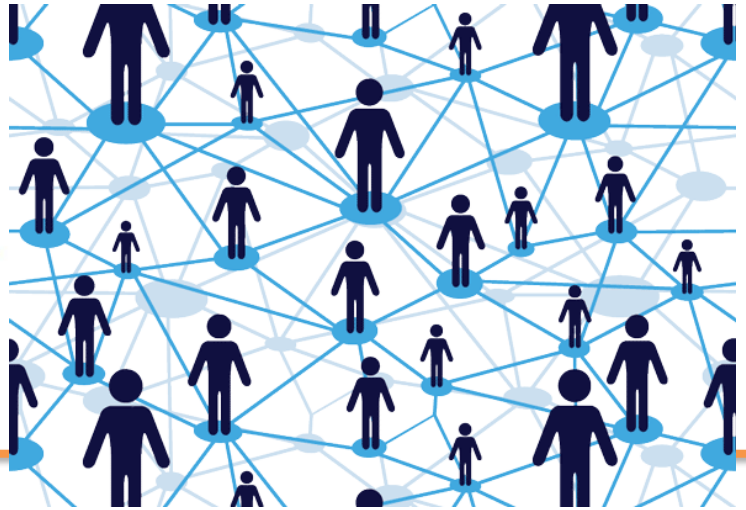


1.2. Phân loại

- Hệ thống phân tán: Peer to Peer (mạng ngang hàng - P2P)
  - P2P không phân biệt clients và servers, tất cả các node là ngang hàng, một máy có thể là client, server hay cả hai (download lẫn upload)
  - Tận dụng được tài nguyên của tất cả các máy trong mạng



BitTorrent™



1.2. Phân loại

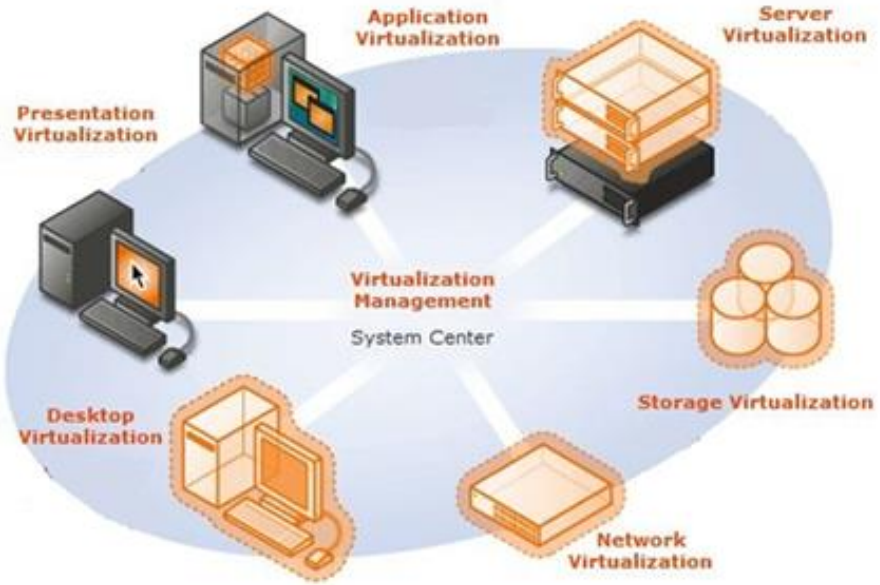
- Hệ thống thời gian thực (Real Time)
  - Sử dụng trong các thiết bị chuyên dụng như điều khiển các thí nghiệm khoa học, quân sự, điều khiển trong y khoa, game thời gian thực
  - Đảm bảo giải quyết bài toán (tiến trình) không muộn hơn một thời điểm xác định
  - Mỗi tiến trình được gắn với một thời gian xác định phải hoàn thành gọi là DeadTime
  - Hoàn thiện bài toán muộn hơn DeadTime thì không còn ý nghĩa

1.2. Phân loại

- Hệ thống thời gian thực (Real Time)
  - Hard real-time:
    - Hạn chế (hoặc không có) bộ nhớ phụ, tất cả dữ liệu nằm trong bộ nhớ chính
    - Yêu cầu về thời gian đáp ứng/xử lý rất nghiêm ngặt, thường sử dụng trong điều khiển công nghiệp, quân sự, robotics,…
  - Soft real-time:
    - Thường được dùng trong lĩnh vực multimedia (đa phương tiện), virtual reality (thực tế ảo) với yêu cầu mềm dẻo hơn về thời gian đáp ứng

1.3. Ảo hóa

- Ảo hóa là công nghệ được thiết kế để tạo ra tầng trung gian giữa hệ thống phần cứng máy tính và phần mềm chạy trên nó
  - Ảo hóa hệ điều hành
  - Ảo hóa máy chủ
  - Ảo hóa lưu trữ
  - Ảo hóa kết nối mạng
  - Ảo hóa ứng dụng
  - …

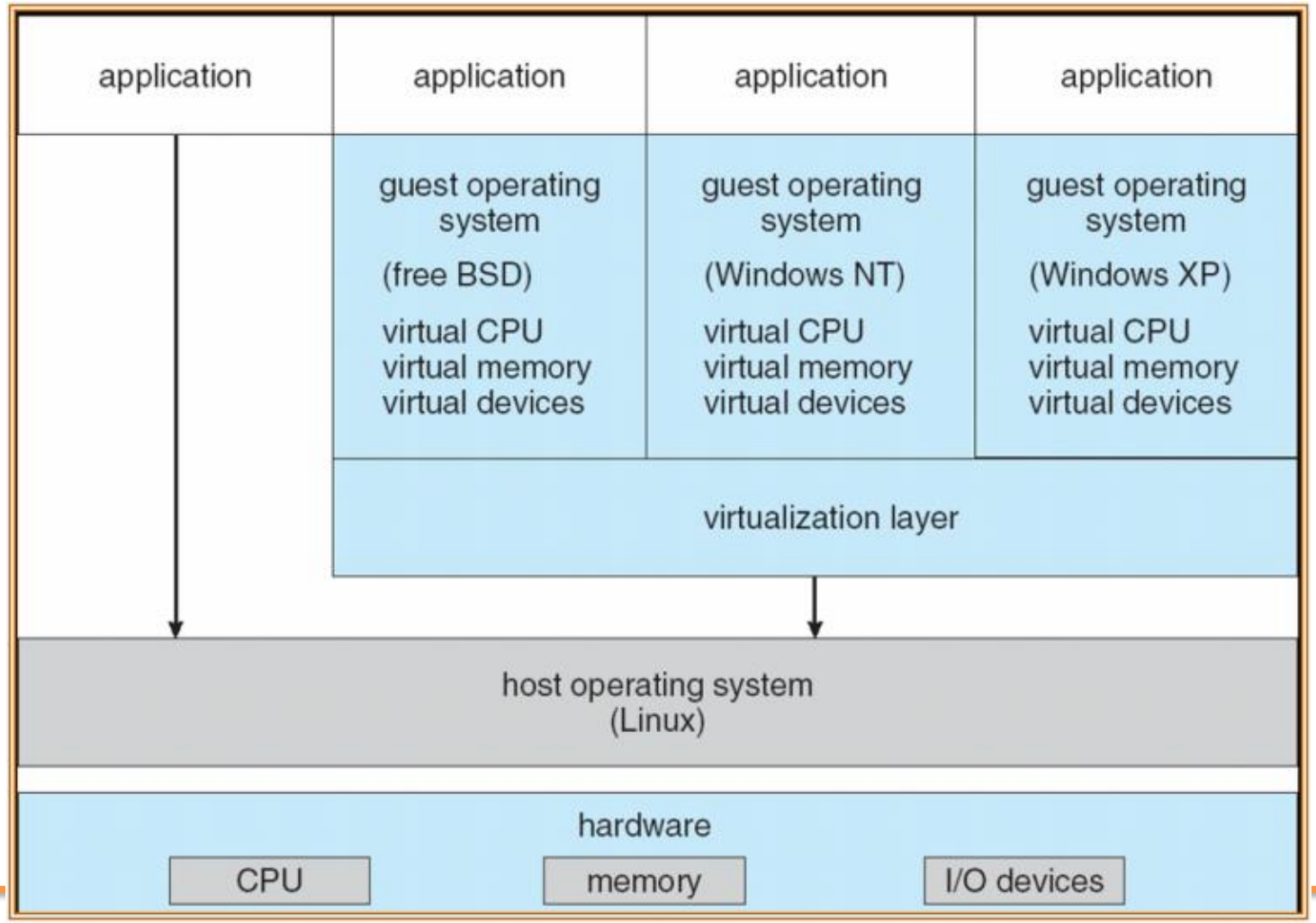


1.3. Ảo hóa

- Máy ảo:
  - Từ một máy thực đơn lẻ có thể tạo ra nhiều máy ảo độc lập. Mỗi máy ảo như vậy đều có một thiết lập thành các hệ thống riêng lẻ: hệ điều hành riêng và các ứng dụng riêng
  - Ưu điểm:
    - Phát triển các ứng dụng cho nhiều hệ điều hành khác nhau mà không cần có nhiều hệ thống vật lý khác nhau
    - Hữu ích cho các nhà phát triển, nếu hệ điều hành hoặc ứng dụng thử nghiệm bị lỗi thì chỉ giới hạn trong máy ảo
    - Ví dụ: Vmware, HyperV,…

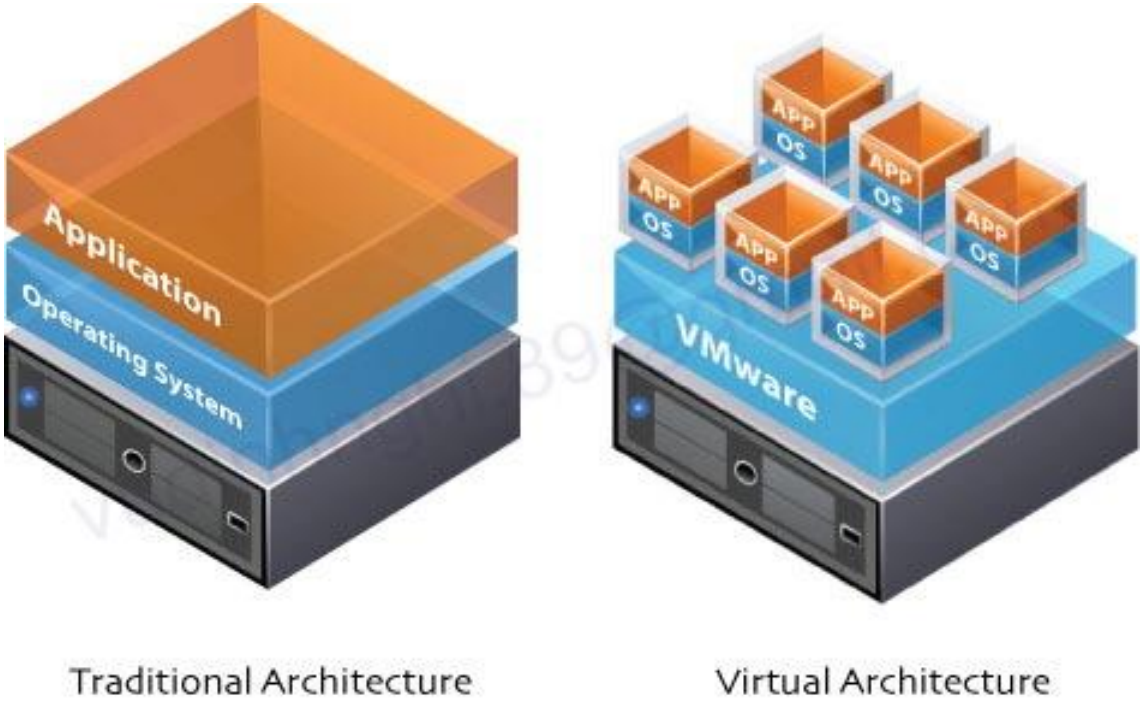
1.3. Ảo hóa

- Máy ảo:



1.3. Ảo hóa

- Máy ảo:



Traditional Architecture

Virtual Architecture

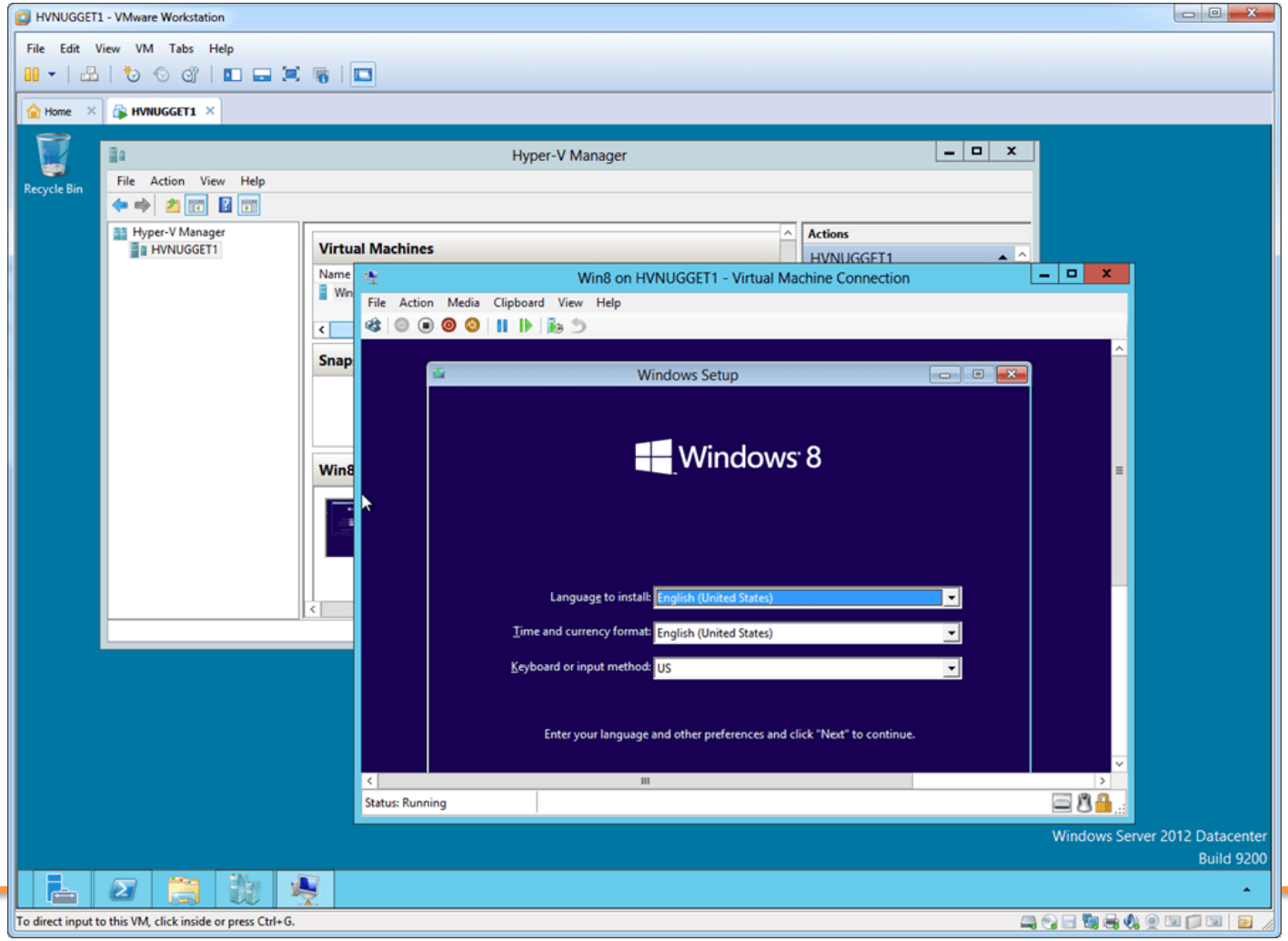
VirtualBox

Microsoft Hyper-V

vmware

1.3. Ảo hóa

- Máy ảo:

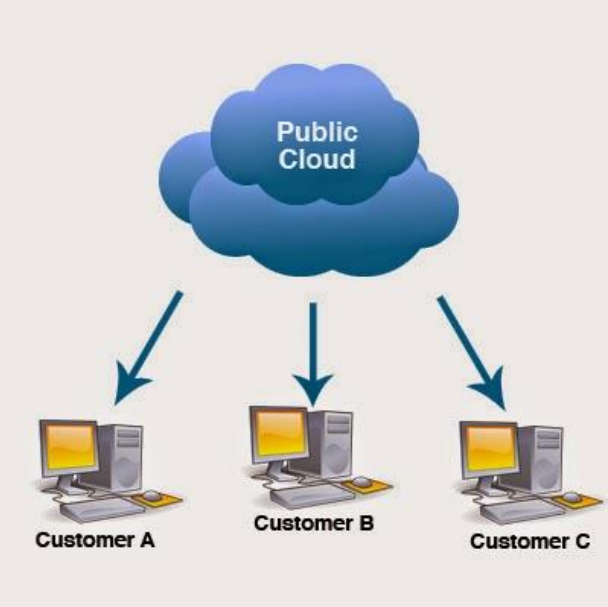


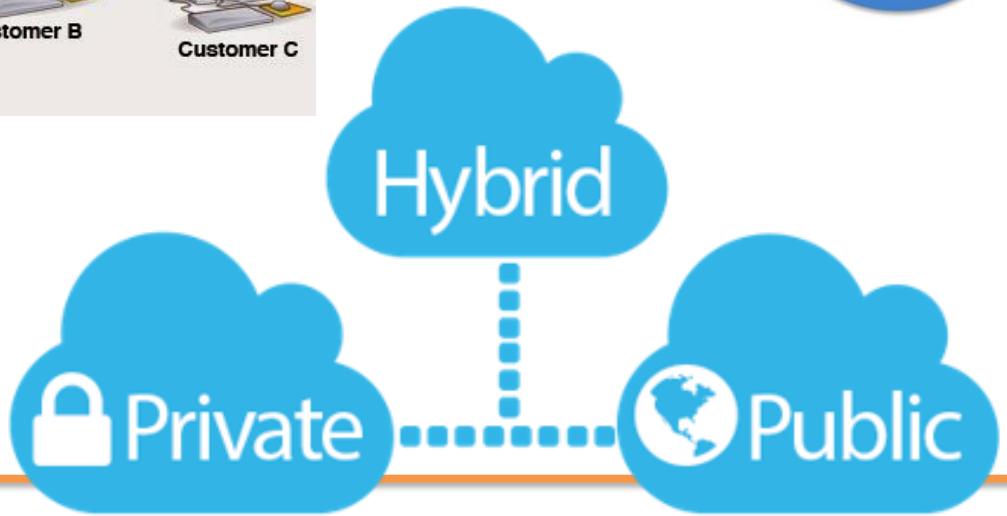
1.4. Điện toán đám mây

- Giúp cung cấp các tài nguyên tính toán, lưu trữ, ứng dụng như là các dịch vụ thông qua mạng
- Các dịch vụ sẽ nằm tại các máy chủ ảo (đám mây) trên Internet thay vì trong máy tính gia đình và văn phòng để mọi người kết nối và sử dụng mỗi khi cần
- Các loại cloud Computing:
  - Public cloud
  - Private cloud
  - Hybrid cloud (public cloud + private cloud)

1.4. Điện toán đám mây

- Một số hình ảnh minh họa:





1.4. Điện toán đám mây

- Một số dịch vụ phổ biến:
  - Google Drive: dịch vụ lưu trữ đám mây của Google
  - OneDrive: dịch vụ lưu trữ đám mây của Microsoft
  - Mediafile.com: dịch vụ chia sẻ file với tốc độ tải file cao
  - Dropbox.com: dịch vụ rất uy tín về độ bảo mật và tiện ích
  - Fshare.vn: dịch vụ của FPT (Việt Nam)

Google Drive

Dropbox

OneDrive

FSHARE

MediaFire

1.5. Các thành phần hệ điều hành

- Quản lý tiến trình (Process Management)
- Thành phần quản lý và phân phối tài nguyên:
  - Quản lý bộ nhớ chính (Main Memory Management)
  - Quản lý File (File Management)
  - Quản lý hệ thống vào-ra (I/O System Management)
  - Quản lý bộ nhớ thứ cấp (Secondary Storage Management)
- Hệ thống bảo vệ (Protection System)
- Hệ thống thông dịch lệnh (Command-Interpreter System)
- Nhân Hệ điều hành (Kernel)

Process Management

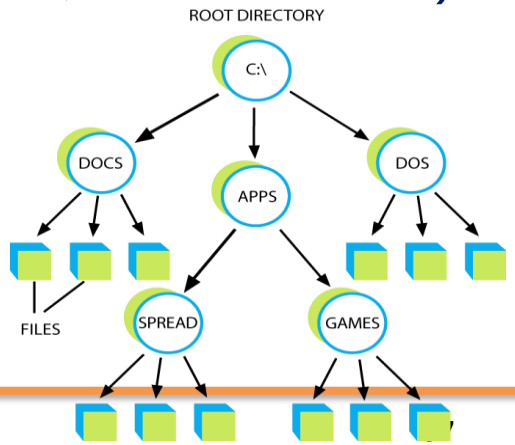
- Tiến trình là một chương trình đang thực thi
- Một tiến trình cần các tài nguyên của hệ thống như CPU, bộ nhớ, file, thiết bị I/O,… để hoàn thành công việc
  - Tài nguyên được cấp khi tiến trình được khởi tạo hay khi đang thi hành
  - Tiến trình kết thúc, tài nguyên được trả về
- HĐH chịu trách nhiệm đối với việc quản lý tiến trình:
  - Tạo và xoá tiến trình của người sử dụng và của hệ thống
  - Tạm ngừng và tiếp tục (suspend/resume) lại tiến trình
  - Cung cấp các cơ chế:
    - Đồng bộ hoạt động các tiến trình (synchronization)
    - Giao tiếp giữa các tiến trình (interprocess communication)
    - Khống chế deadlock

Main Memory Management

- Bộ nhớ chính là một mảng lớn words (các từ) hay bytes có kích thước lớn, mỗi word hay byte có địa chỉ riêng
  - Chứa dữ liệu có thể truy nhập nhanh chóng
  - Được chia sẻ bởi CPU và các thiết bị vào\ra
- Main memory là một thiết bị lưu trữ tạm
  - Mất nội dung bên trong khi hệ thống ngừng hoạt động
- Hoạt động quản lý bộ nhớ chính của HĐH
  - Lưu lại dấu vết của các phần bộ nhớ đang được sử dụng và được sử dụng bởi tiến trình nào
  - Quyết định xem những tiến trình nào được nạp khi có bộ nhớ trống
  - Phân phối và thu hồi bộ nhớ cho các tiến trình

File Management

- Một file là một tập hợp các thông tin liên quan được tổ chức một cách lôgic, được lưu trữ trên thiết bị nhớ phụ (đĩa, usb,…)
- Các file có thể đại diện cho dữ liệu và chương trình (cả source, data, forms,…)
- Hoạt động quản lý file của Hệ điều hành:
  - Tạo/ xóa một tập tin/ thư mục
  - Hỗ trợ các thao tác khác với file (phân quyền, đổi tên file,…)
  - Ánh xạ file trên các thiết bị lưu trữ phụ



I/O System Management

- HĐH phải ra các chỉ thị điều khiển thiết bị, kiểm soát các ngắt và lỗi từ thiết bị I/O
- HĐH phải cung cấp một cách giao tiếp đơn giản và tiện dụng giữa các thiết bị và phần còn lại của hệ thống
- Hệ thống vào/ra bao gồm:
  - Một thành phần quản lý bộ nhớ I/O (buffer-caching system)
  - Một giao diện device driver chung đến các trình điều khiển thiết bị (general device-driver interface)
  - Trình điều khiển thiết bị các thiết bị I/O riêng biệt (device driver)
- Chỉ device driver biết các tính chất đặc biệt của thiết bị mà nó điều khiển

Secondary Storage Management

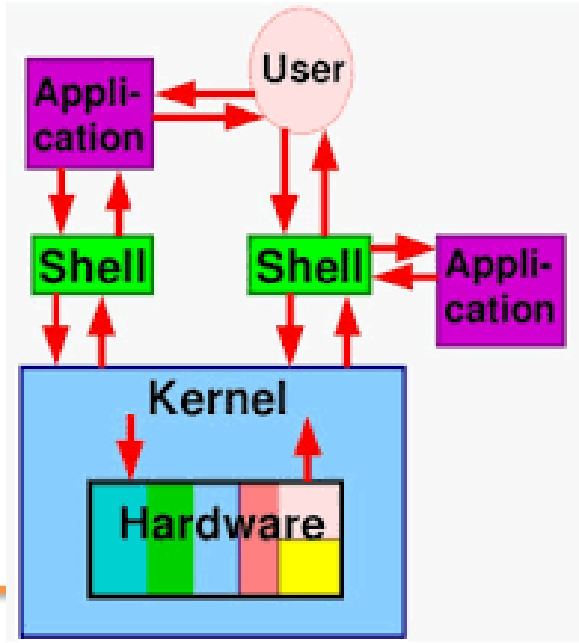
- Dữ liệu trên bộ nhớ chính (primary storage):
  - Tạm thời, dung lượng nhỏ
  - Để lưu trữ lâu dài, dung lượng lớn → cần sử dụng bộ nhớ thứ cấp (secondary storage)
  - Hầu hết các hệ thống máy tính hiện đại sử dụng ổ đĩa như là các phương tiện lưu trữ dữ liệu cơ sở cho cả chương trình và dữ liệu (SSD, HDD)
- Tốc độ chung của toàn bộ hệ thống phụ thuộc rất nhiều vào hệ thống đĩa và các giải thuật sử dụng trên nó
- Hoạt động quản lý hệ thống đĩa của hệ điều hành
  - Quản lý các không gian còn trống
  - Cấp phát không gian lưu trữ
  - Lập lịch ổ đĩa (disk scheduling)

Protection System

- Khi hệ thống cho phép nhiều user cùng làm việc hay nhiều quá trình cùng thực hiện, hệ thống cần phải:
  - Kiểm soát quá trình người dùng đăng nhập/xuất và sử dụng hệ thống
  - Kiểm soát việc truy cập các tài nguyên trong hệ thống
  - Bảo đảm chỉ những người dùng/quá trình đủ quyền hạn mới được phép sử dụng các tài nguyên tương ứng

Command-Interpreter System (Shell)

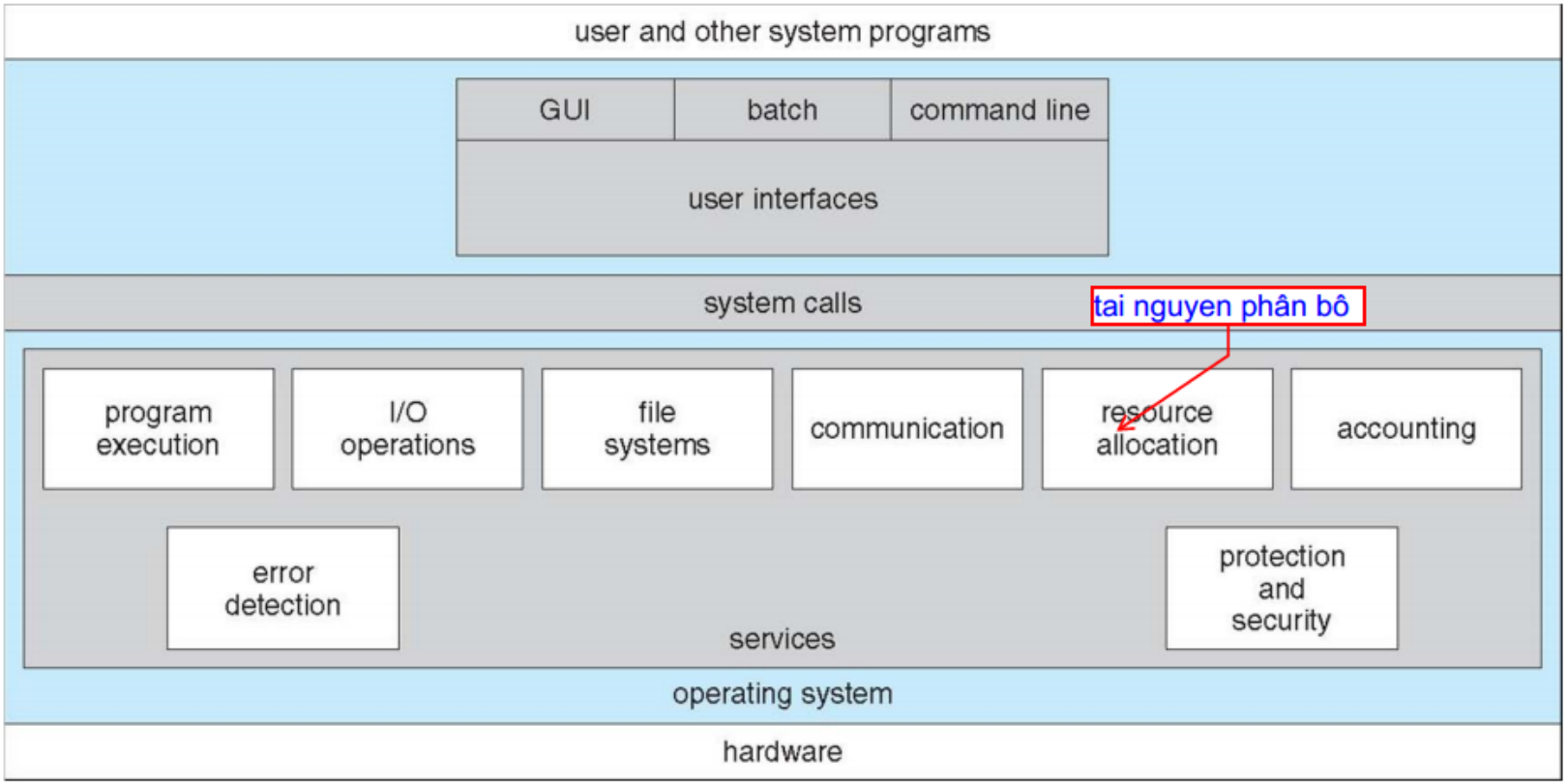
- Là giao diện giữa người dùng và hệ điều hành
  - Ví dụ: command-line, mouse-based window-and-menu
- Nhận và thực hiện các câu lệnh điều khiển của người dùng để thực hiện các tác vụ như: quản lý quá trình, quản lý vào/ra, quản lý bộ nhớ, truy cập hệ thống tập tin,…
- Được cài đặt trong kernel (DOS) hoặc qua các chương trình hệ thống (Widows, Linux)
- Liên hệ chặt chẽ với các thành phần khác của hệ điều hành để thực thi các yêu cầu của người dùng
- Các nhóm lệnh của trình thông dịch lệnh:
  - Tạo, hủy, xem thông tin quá trình, hệ thống
  - Điều khiển truy cập I/O
  - Quản lý, truy cập hệ thống lưu trữ thứ cấp
  - Quản lý, sử dụng bộ nhớ
  - Truy cập hệ thống file
  - …



Nhân Hệ điều hành (Kernel)

- Hệ điều hành có dung lượng lớn, vì vậy không thể đưa tất cả vào bộ nhớ
- Các chương trình của hệ điều hành phân thành 2 loại:
  - Nhân HĐH: gồm các chương trình luôn tồn tại trong bộ nhớ để điều khiển máy tính làm việc
  - Các chương trình chỉ được nạp vào bộ nhớ khi cần thiết
- Theo chức năng nhân HĐH thường bao gồm các nhóm chương trình:
  - Module chương trình tải (loader) có chức năng đưa một chương trình vào bộ nhớ và cho phép chương trình đã tải nhận điều khiển hay không
  - Module chương trình điều phối chính: đảm nhận việc lựa chọn các bước làm việc của toàn bộ hệ thống
  - Module chương trình lập lịch: chọn tiến trình làm việc tiếp theo
  - Các thông tin về hệ thống (các tham số hệ thống)

1.6. Các dịch vụ của hệ điều hành



- User interface (UI) - Hầu hết tất cả các hệ điều hành điều có một giao diện người dùng: Command-Line (CLI), Graphics User Interface (GUI), Batch

1.6. Các dịch vụ của hệ điều hành

- Thực hiện chương trình (Program execution): Nạp chương trình vào bộ nhớ và thực thi, kết thúc nó
- Thực hiện vào/ra (I/O operations): Cung cấp các phương tiện để thực hiện các thao tác I/O
- Thao tác với hệ thống file (File-system manipulation): Cung cấp các phương tiện để chương trình có thể đọc, ghi, tạo, xóa, liệt kê, tìm kiếm, quản lý quyền truy cập trên tập tin/thư mục
- Giao tiếp (Communications): Trao đổi thông tin giữa các tiến trình đang thực hiện trên cùng 1 máy tính hoặc trên các hệ thống khác nhau được nối mạng thông qua mạng máy tính (shared memory hoặc message passing)
- Phát hiện lỗi (Error detection): Bảo đảm việc tính toán đúng đắn bằng cách phát hiện các lỗi trong CPU và bộ nhớ, trong các thiết bị vào/ra, hoặc trong chương trình của người sử dụng

Các chức năng mở rộng của HĐH

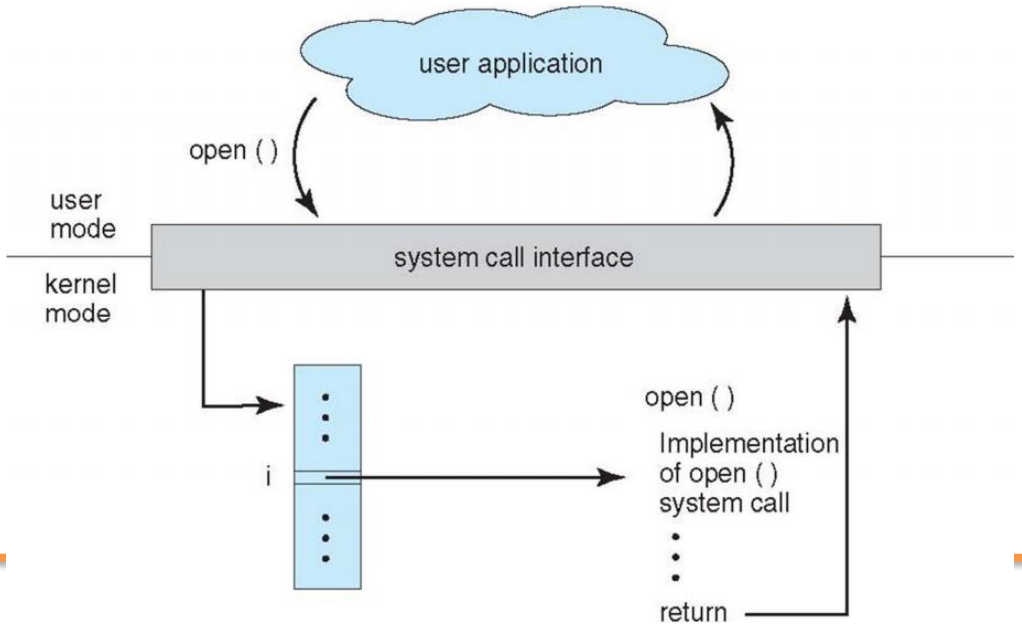
- Các chức năng bổ sung thêm để đảm bảo cho các hoạt động của hệ thống hiệu quả hơn
  - Cấp phát tài nguyên (Resource allocation): Phân phối tài nguyên cho nhiều người sử dụng hoặc cho nhiều công việc chạy cùng lúc
  - Thống kê (Accounting): Lưu giữ thông tin về loại và số lượng tài nguyên sử dụng, nhằm sử dụng cho thống kê, tính toán (giá thành sử dung), nghiên cứu (cải tiến hệ thống)
  - Bảo vệ và an ninh (Protection and Security):
    - Protection: Đảm bảo tất cả các truy cập đến các nguồn tài nguyên của hệ thống điều được kiểm soát
    - Security: Đảm bảo an toàn cho hệ thống từ các truy cập bên ngoài

1.7. System Call

- System call (hàm thư viện của hệ điều hành): Là giao diện giữa tiến trình và hệ điều hành dùng để gọi các dịch vụ của hệ điều hành
- Các lời gọi hệ thống thường được cài đặt bằng các ngôn ngữ bậc cao (C, C++), gọi là các giao diện lập trình ứng dụng (Application Programming Interface - API)
- Phụ thuộc vào từng loại hệ điều hành
- Một số API phổ biến:
  - Windows API (cho HĐH Windows)
  - POSIX API (cho POSIX-Based systems như Linux, Unix, MacOS)
  - Java API (cho Java Virtual Machine)

1.7. System Call

- Giao diện lời gọi hệ thống (System call interface): cung cấp giao diện trực tiếp đến các lời gọi hệ thống bên trong nhân
- Giao diện lời gọi hệ thống gọi system call được chỉ định trong nhân HĐH (OS kernel) và trả về trạng thái + giá trị (nếu có) của lời gọi hệ thống

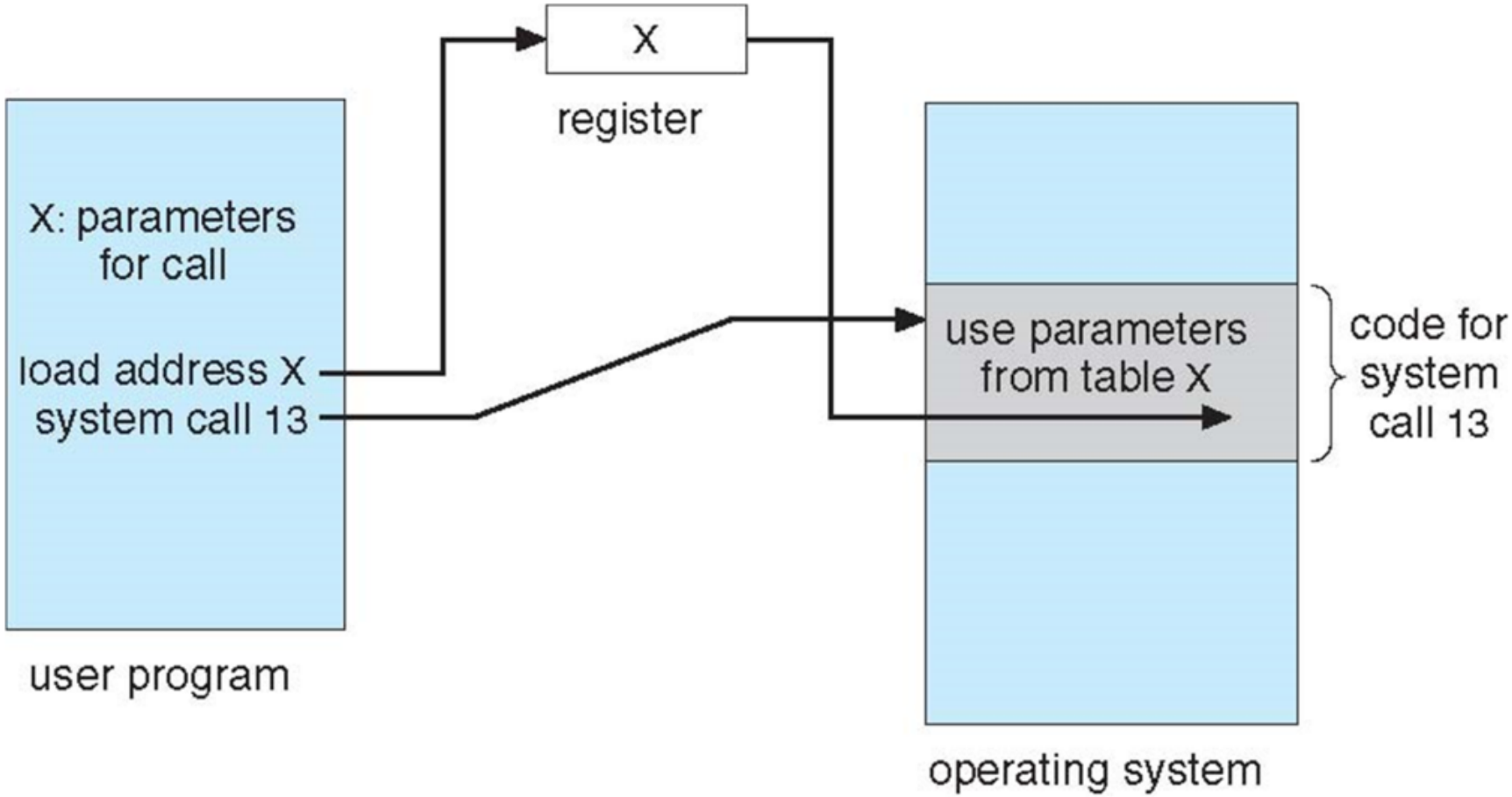


1.7. System Call

- Một lời gọi hệ thống thường kèm theo các tham số
- Ba phương thức tổng quát được sử dụng để truyền tham số giữa tiến trình và hệ điều hành:
  - Truyền qua thanh ghi: giới hạn số lượng tham số vì số thanh ghi tương đối ít
  - Truyền qua bộ nhớ: các tham số được lưu trong một bảng tham số trong bộ nhớ, và địa chỉ của bảng được truyền như một tham số trong một thanh ghi
  - Truyền qua stack: đẩy (push) các tham số vào stack bằng chương trình, và lấy ra (pop) khỏi stack bởi hệ điều hành

1.7. System Call

- Truyền tham số qua bảng tham số lưu trong bộ nhớ:



1.7. System Call

- UNIX/Win32 API (Application Programming Interface)

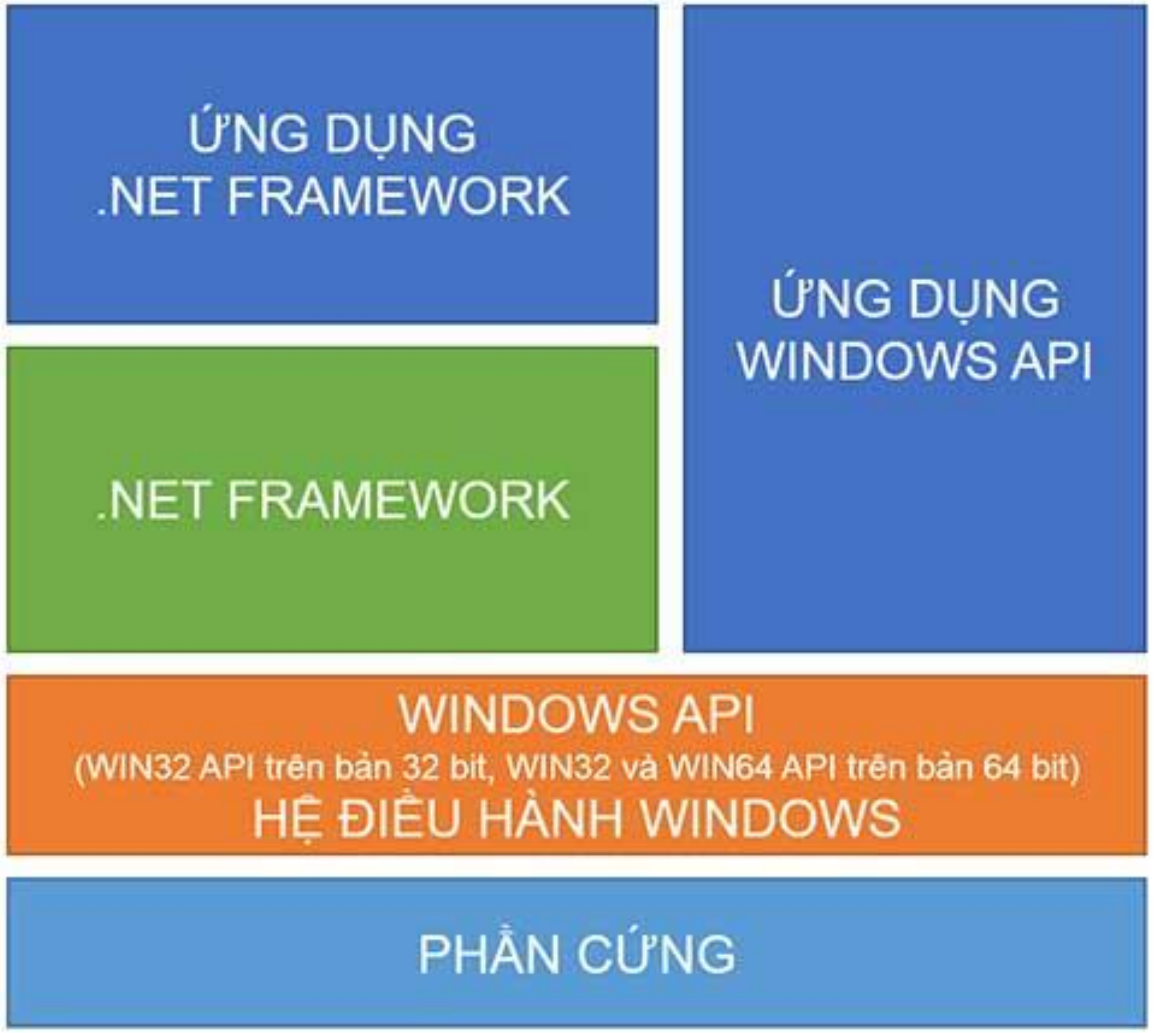
UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	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

1.7. System Call

- Windows API
  - Windows API được Microsoft cung cấp cho người lập trình, để viết các ứng dụng trên nền Windows
  - Windows API là giao diện lập trình nằm ngay trên nền Windows, cung cấp các hàm thao tác trực tiếp với hệ điều hành và phần cứng máy tính
  - Các ứng dụng Windows sẽ thông qua Windows API để thao tác với máy tính
- .NET Framework:
  - Gọi lại các hàm Windows API (nếu gọi một phương thức .NET nào đó, nó sẽ gọi lại các hàm Windows API có chức năng tương ứng để thực hiện, chứ không thao tác trực tiếp đến hệ điều hành)

1.7. System Call

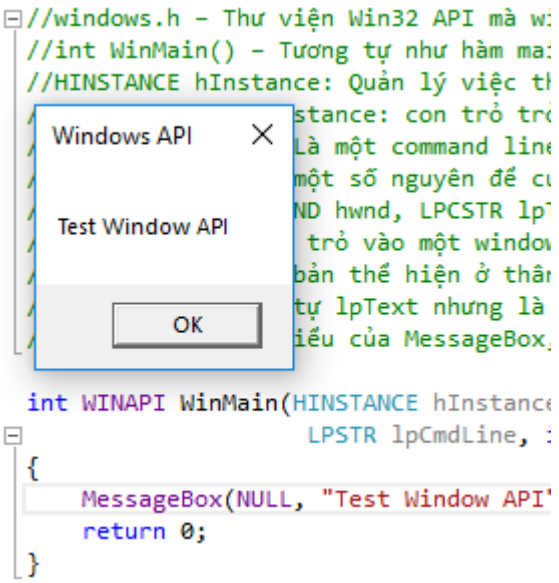
– Windows API và .NET Framework



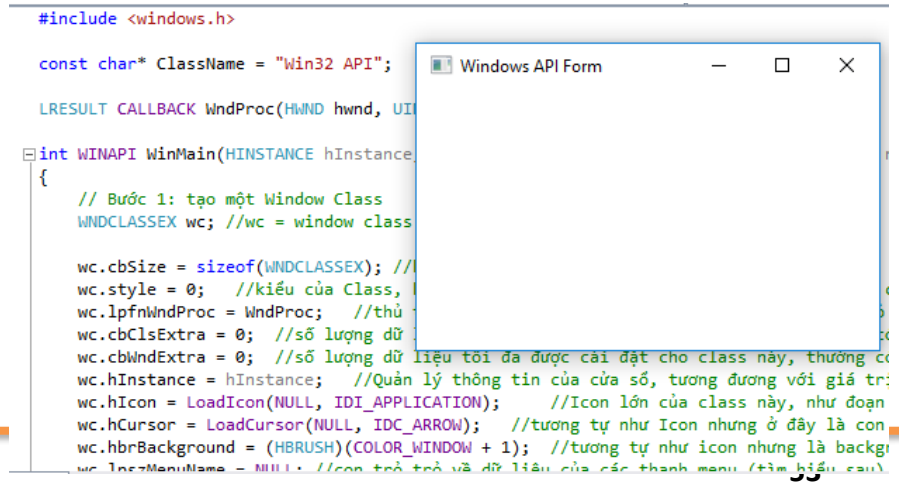
1.7. System Call

- Windows API
  - Ví dụ 1: Hiện message thông báo

```
int WINAPI WinMain(HINSTANCE hInstance,
HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow)
{
        MessageBox(NULL, "Test Window
API", "Windows API", MB_OK);
       return 0;
}
```



  - Ví dụ 2: Hiện form đơn giản



1.7. System Call

- Windows API
  - Ví dụ 3: Hiện form có menu

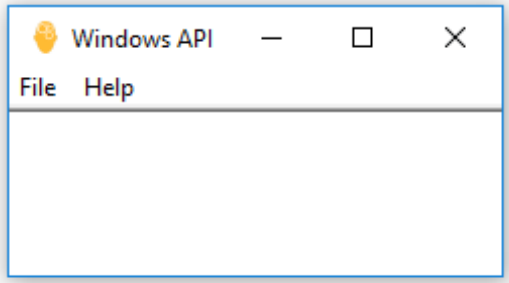
```
const char* ClassName = "Win32 API";

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);
BOOL CALLBACK AboutBox(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    WNDCLASSEX wc;
    HWND hwnd;
    MSG Msg;

    wc.cbSize = sizeof(WNDCLASSEX);
    wc.style = 0;
    wc.lpfnWndProc = WndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon(GetModuleHandle(NULL), MAKEINTRESOURCE(IDI_ICON1));
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
    wc.lpszMenuName = MAKEINTRESOURCE(IDR_MENU1);
    wc.lpszClassName = ClassName;
    wc.hIconSm = LoadIcon(GetModuleHandle(NULL), MAKEINTRESOURCE(IDI_ICON1));

    if (!RegisterClassEx(&wc))
    {
        MessageBox(NULL, "Window Registration Failed!", "Error!", MB_ICONEXCLAMATION | MB_OK);
        return 0;
```

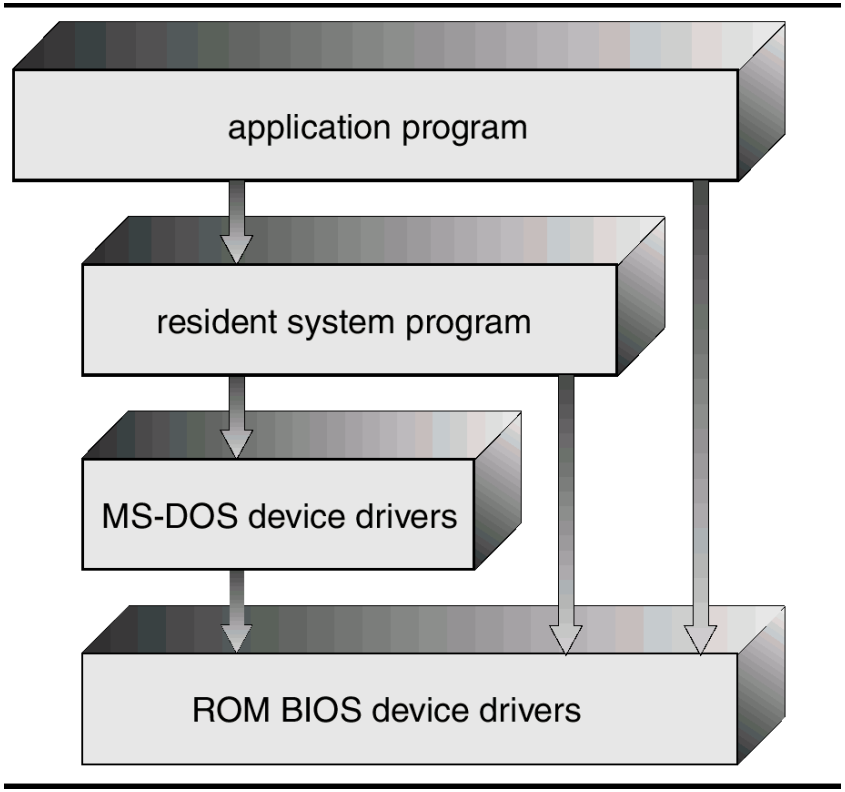


1.7. System Call

- Phân loại:
  - Điều khiển tiến trình (process control): khởi tạo, huỷ, thực thi tiến trình,…
  - Quản lý bộ nhớ (memory control): cấp phát và giải phóng bộ nhớ,...
  - Quản lý file (file management): tạo, xóa, đóng, mở tệp, đọc, thiết lập thuộc tính,...
  - Quản lý thiết bị (device management): thực hiện trao đổi vào/ra, Attach/detach thiết bị,…
  - Duy trì thông tin trạng thái (information maintenance): lấy/thiết lập ngày giờ, đọc, ghi thông tin hệ thống,…
  - Giao tiếp (communication): tạo, xóa kết nối giao tiếp, gửi, nhận thông điệp,…

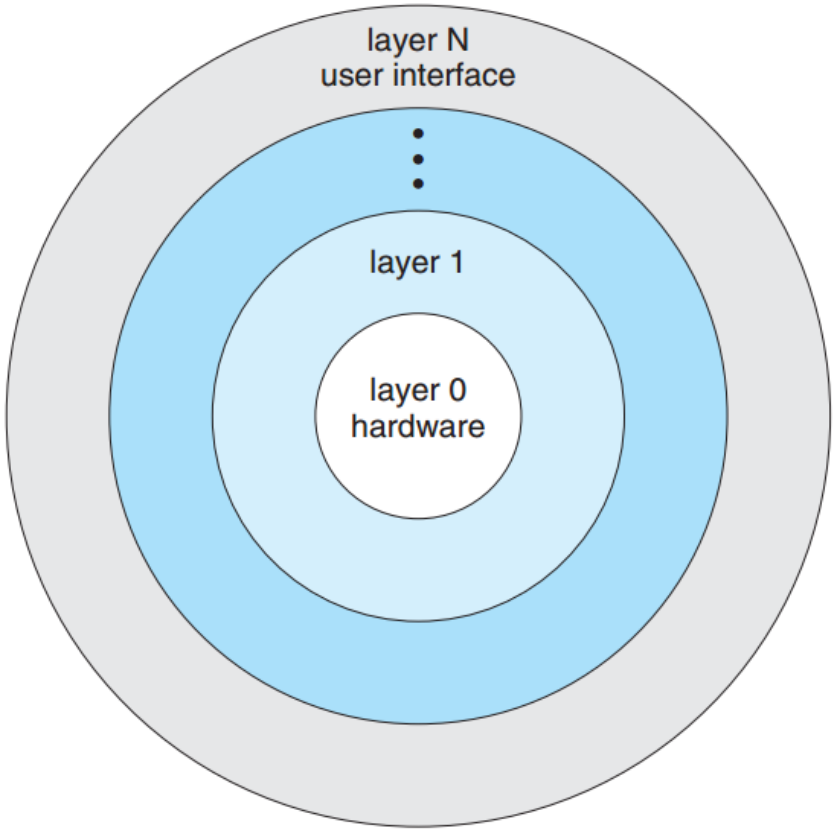
1.8. Kiến trúc hệ điều hành

- Kiến trúc đơn giản:
  - MS-DOS: Không có kiến trúc rõ ràng, không được chia thành các modules (do giới hạn về dung lượng bộ nhớ)
  - Mặc dù MS-DOS được tổ chức có cấu trúc, các lớp chức năng cũng như giao diện giữa chúng không được phân chia tốt



1.8. Kiến trúc hệ điều hành

- Kiến trúc phân tầng:
  - Hệ điều hành được chia thành một số tầng, mỗi tầng được xây dựng trên nền tảng của một tầng khác thấp hơn
  - Tầng thấp nhất là tầng vật lý, tầng cao nhất là giao diện với người dùng
  - Sự phân chia chức năng: mỗi một tầng sẽ sử dụng các hàm (thao tác) và dịch vụ được cung cấp duy nhất bởi tầng phía dưới liền kề nó



1.8. Kiến trúc hệ điều hành

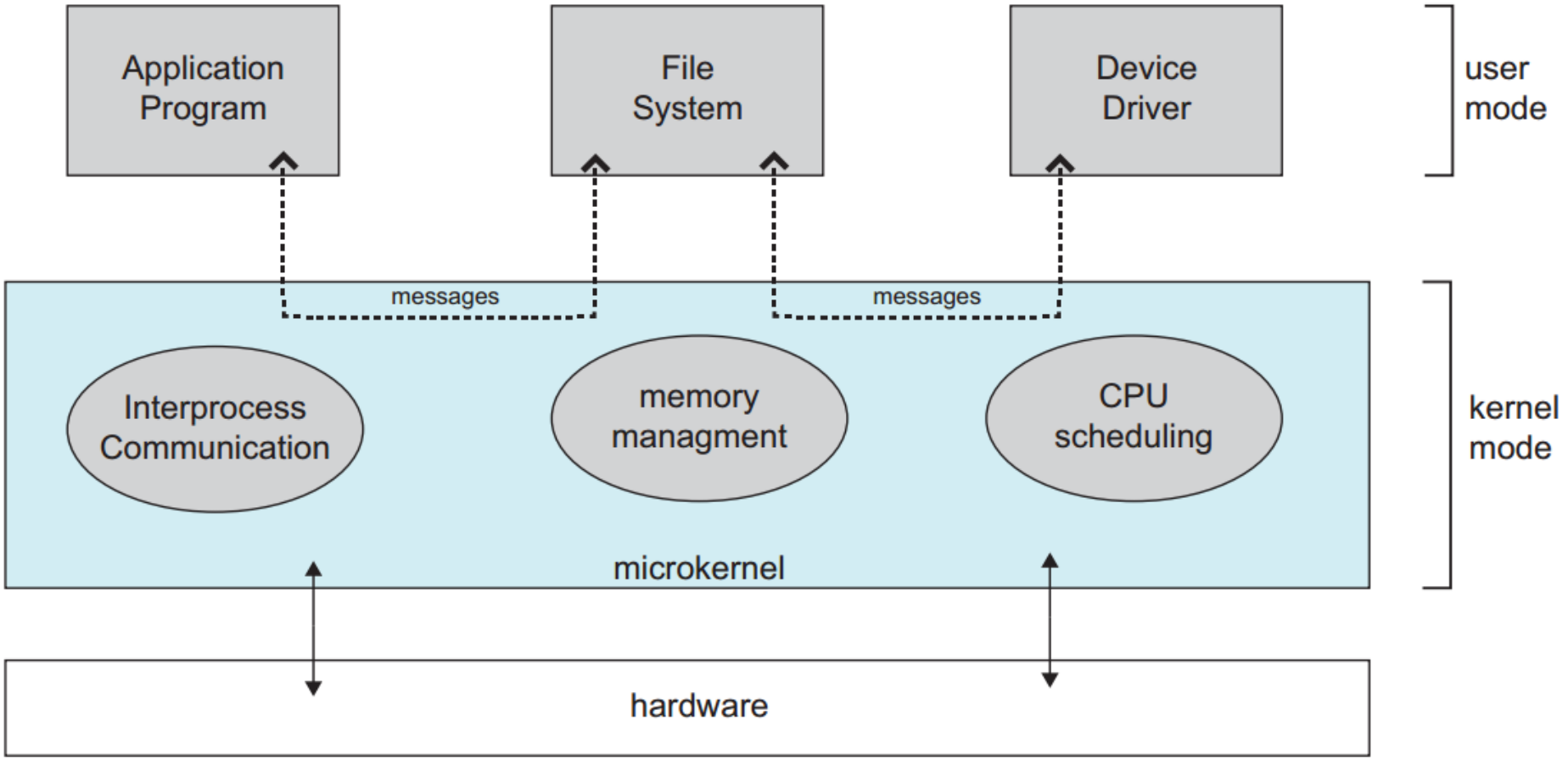
- Kiến trúc phân tầng:
  - Ưu điểm:
    - Tính module → đơn giản hóa trong việc thiết kế, cài đặt, gỡ rối và kiểm tra hệ thống
    - Đơn giản hóa được thể hiện qua việc có thể sửa đổi, cải tiến tại từng tầng, không ảnh hưởng đến các tầng khác
  - Nhược điểm:
    - Cần phải định nghĩa cẩn thận chức năng các tầng vì mỗi tầng chỉ có thể sử dụng các tầng dưới nó
    - Đôi khi khó khăn trong việc xác định một chức năng của HĐH nằm tại tầng nào
    - Tăng chi phí cho việc gọi các lời gọi hệ thống thông qua nhiều tầng

1.8. Kiến trúc hệ điều hành

- Kiến trúc vi nhân:
  - Di chuyển nhiều chức năng từ nhân lên mức người dùng, giữ lại các phần chính yếu: quản lý quá trình, bộ nhớ, giao tiếp giữa các quá trình → nhân nhỏ hơn
  - Việc giao tiếp giữa các module người dùng được thực hiện bằng cách sử dụng cơ chế chuyển thông điệp gián tiếp thông qua nhân
  - Lợi ích:
    - Dễ dàng mở rộng hệ điều hành như các dịch vụ mới (đưa vào không gian người dùng)
    - Dễ dàng chuyển đổi hệ điều hành sang các kiến trúc mới (do nhân nhỏ hơn)
    - Tin cậy hơn và an toàn hơn (ít mã lệnh chạy ở mức nhân hơn)
  - Nhược điểm: Chi phí cho việc giao tiếp giữa các tiến trình trong không gian người dùng và nhân

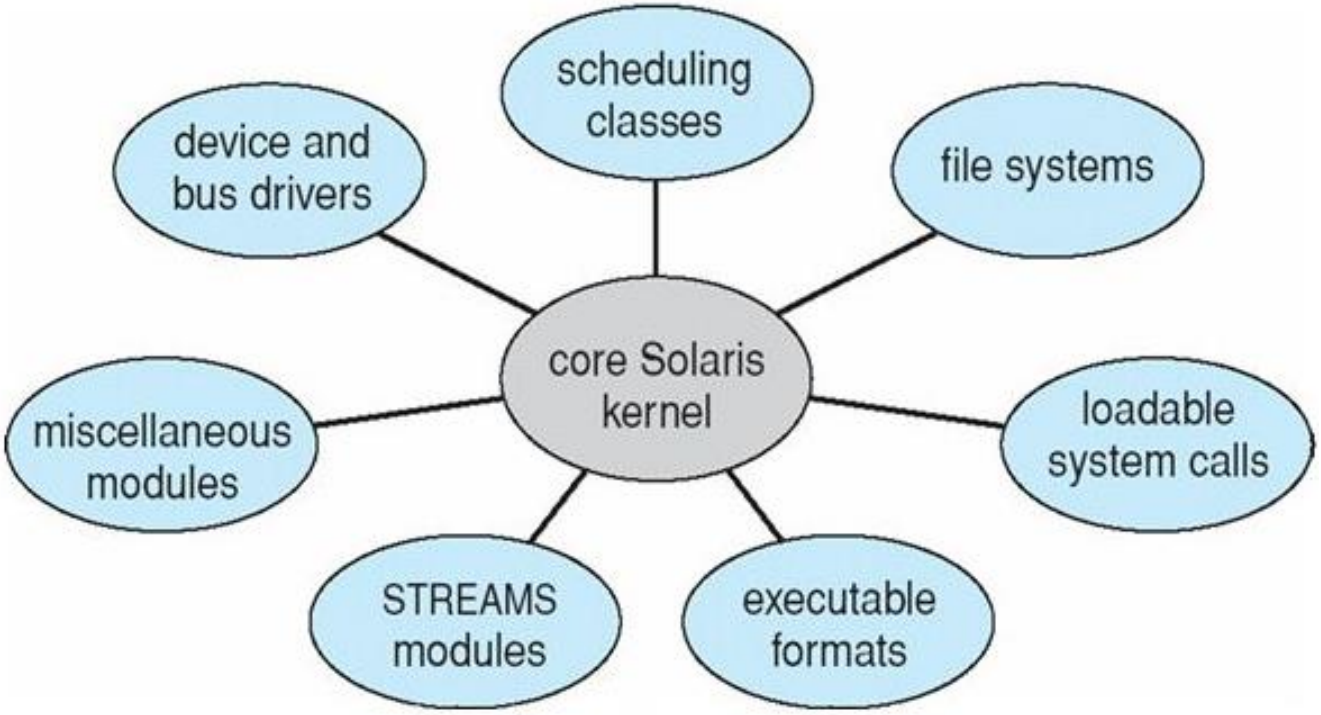
1.8. Kiến trúc hệ điều hành

- Kiến trúc vi nhân:



1.8. Kiến trúc hệ điều hành

- Kiến trúc module:
  - Dùng cách tiếp cận hướng đối tượng (object-oriented)
  - Các module dễ dàng liên lạc trực tiếp với nhau
  - Nhân là nhỏ, các module được nạp vào trong nhân khi cần thiết

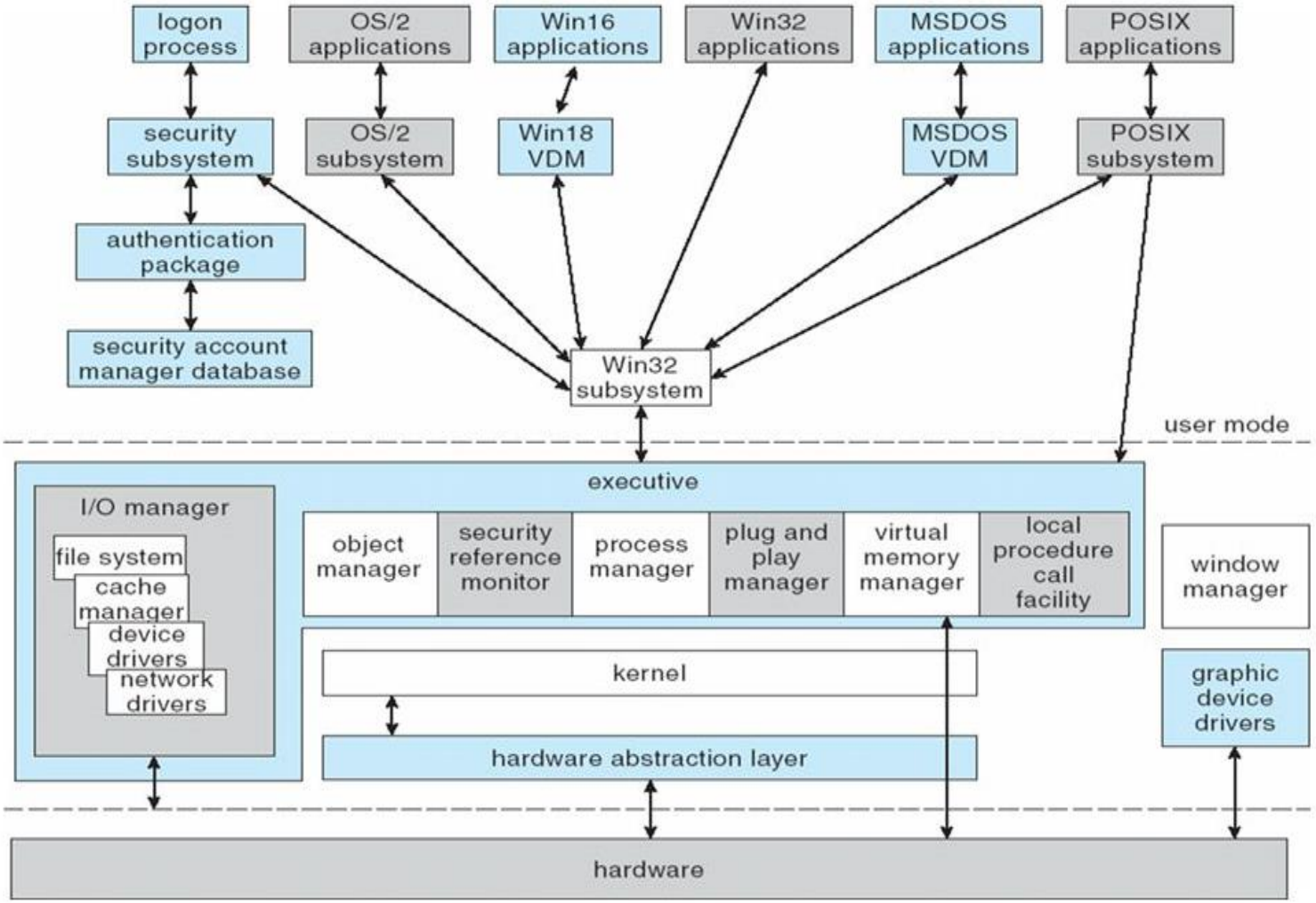


1.8. Kiến trúc hệ điều hành

- Kiến trúc Hybrid System:
  - Hầu hết các hệ điều hành không thật sự dùng một kiểu cấu trúc đơn nhất
  - Kết hợp nhiều hướng tiếp cận nhằm đạt được hiệu suất, độ an toàn, khả năng linh hoạt cao nhất có thể

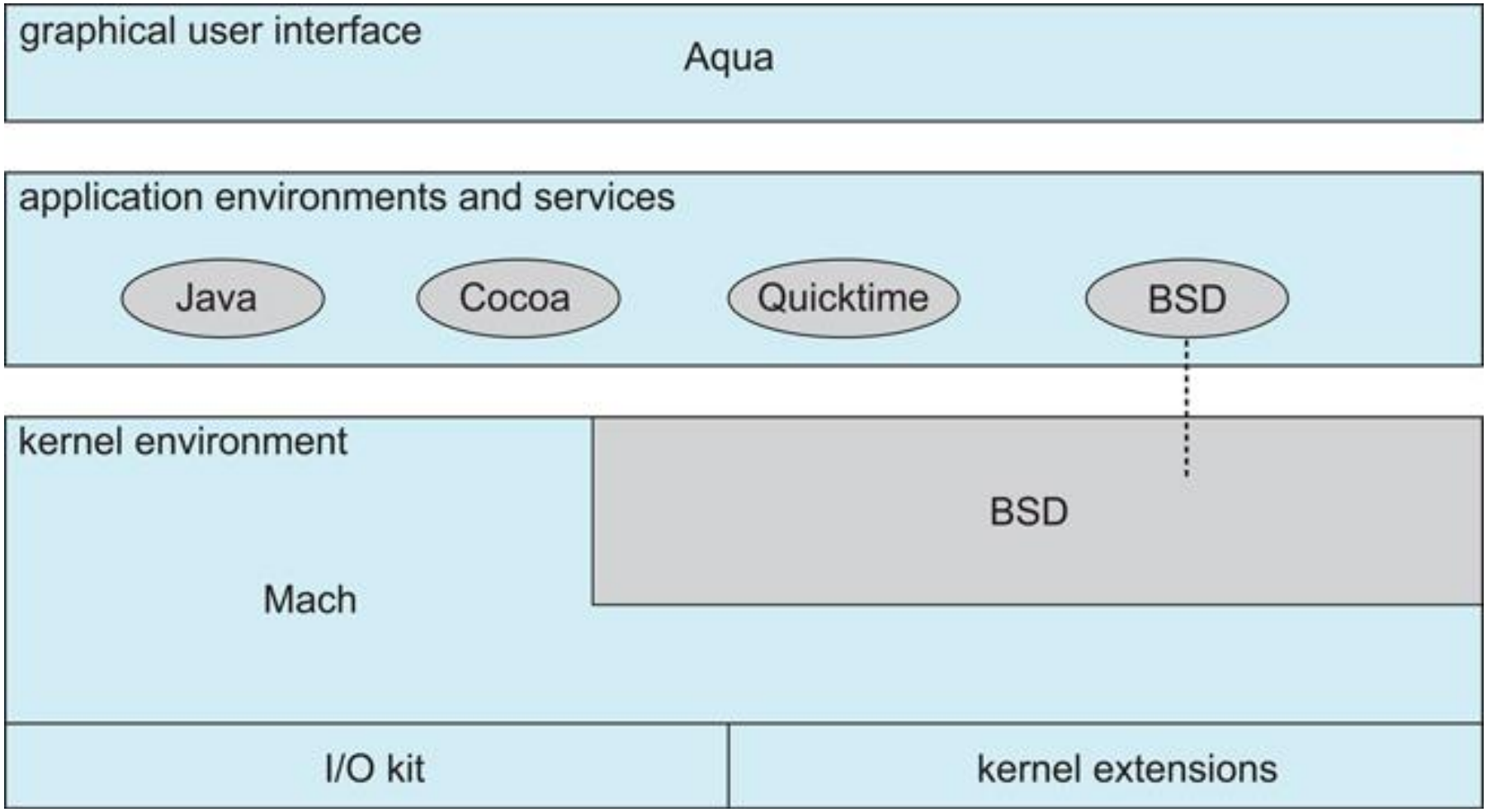
1.8. Kiến trúc hệ điều hành

- Kiến trúc hệ điều hành Windows



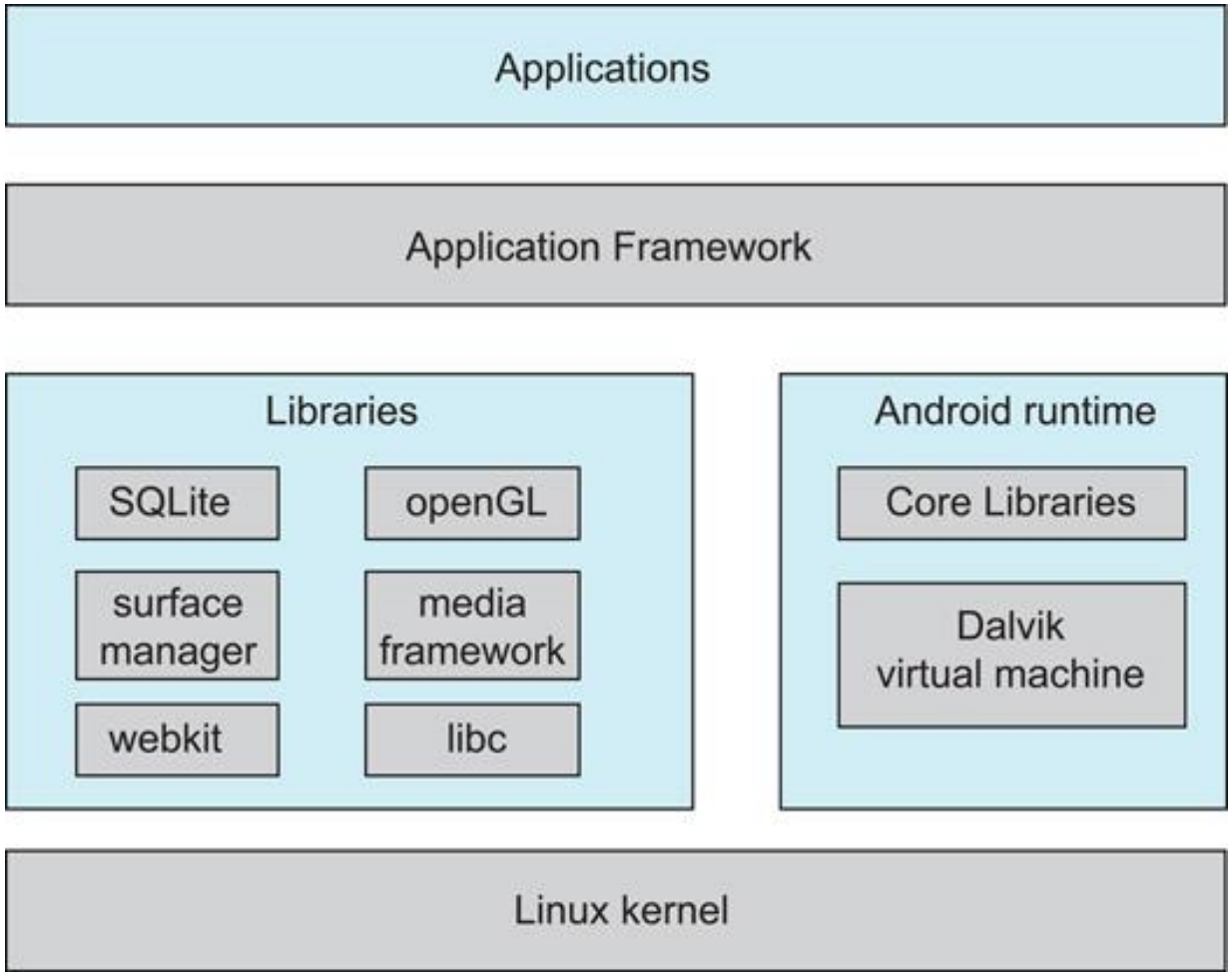
1.8. Kiến trúc hệ điều hành

- Kiến trúc hệ điều hành Mac OS X



1.8. Kiến trúc hệ điều hành

- Kiến trúc hệ điều hành Android



Câu hỏi thảo luận