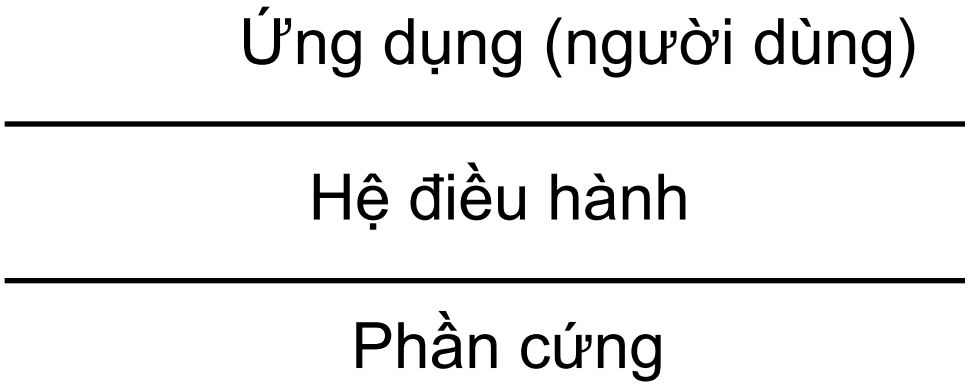
Hệ điều hành là gì?

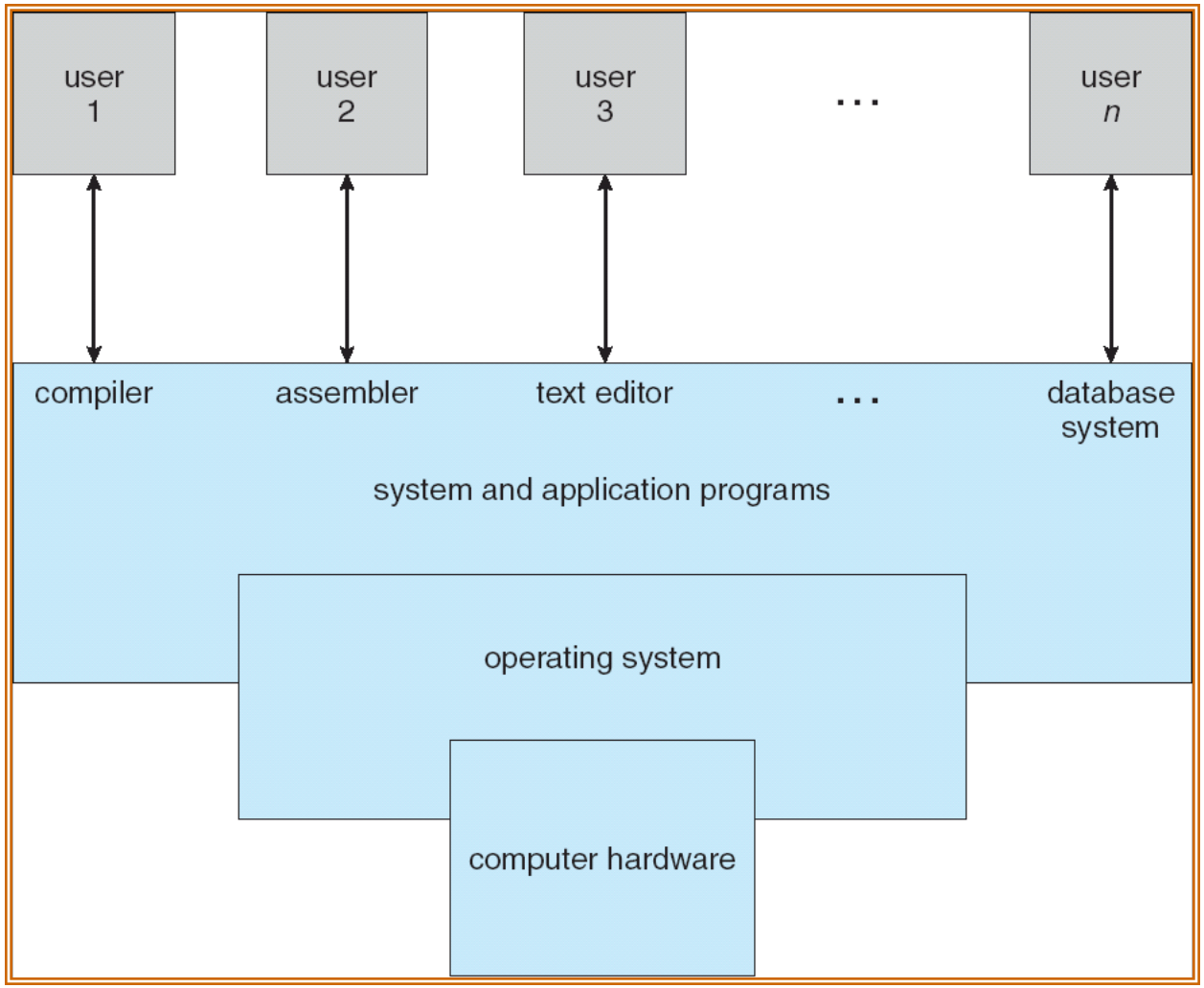


Một lớp phần mềm ở giữa phần cứng và các chương trình ứng dụng/người dùng, nó cung cấp một giao diện máy ảo (*virtual machine)* : dễ dàng và an toàn

Một bộ quản lý tài nguyên (*resource manager)* cho phép các chương trình/người dùng chia sẻ tài nguyên phần cứng: công bằng và hiệu quả

Một tập các tiện ích để đơn giản hóa việc phát triển ứng dụng

Tổng quát hóa các thành phần của hệ thống



Tại sao chúng ta cần hệ điều hành?

Lợi ích cho người lập trình

- Dễ dàng hơn trong việc lập trình
  - Chỉ thấy mức trừu tượng cao, không cần phải biết chi tiết phần cứng
    - V.d. tập tin chứ không phải các blocks trên ổ cứng
- Tính tương thích

Lợi ích của người sử dụng máy tính

- Dễ dàng sử dụng máy tính
  - Bạn có thể hình dung việc sử dụng máy tính không cần hệ điều hành?
- An toàn
  - HĐH bảo về chương trình giữa các chương trình khác nhau
  - HĐH bảo về người dùng giữa các người dùng khác nhau

Cơ chế và chính sách

Ứng dụng (người dùng)

Hệ điều hành: *cơ chế* + *chính sách*

Phần cứng

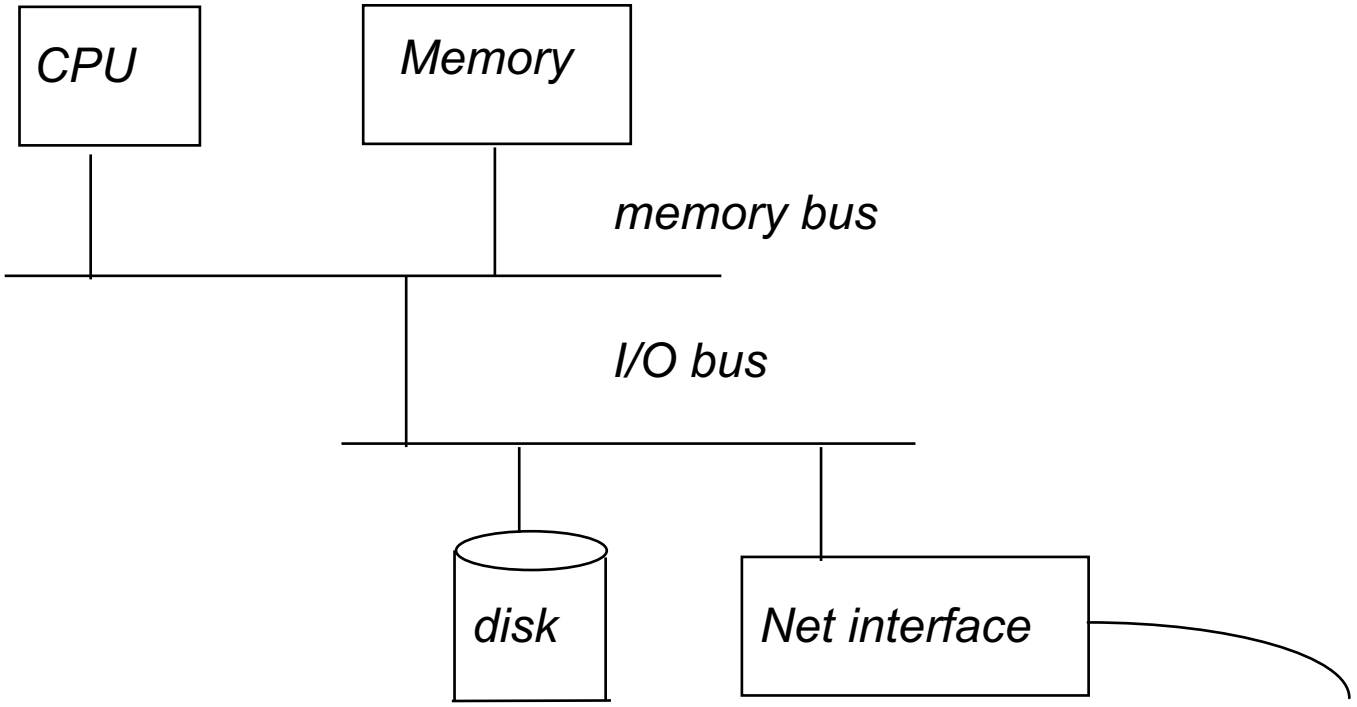
Cơ chế: cấu trúc dữ liệu và các thao tác trên một đối tượng nào đó (v.d. buffer cache)

Chính sách: các phương thức đưa ra quyết định khi phải chọn lựa (v.d. chính sách thay thế buffer cache)

Cần phân tách cơ chế và chính sách càng rõ ràng càng tốt

Các hệ điều hành khác nhau có thể cần các chính sách khác nhau

Cấu trúc máy tính cơ bản



Biểu trưng của hệ thống: Tiến trình

Một tiến trình là một khái niệm trừu tượng:
Tạo ảo giác như là công việc duy nhất trong một hệ thống.

Người dùng: *sử dụng ứng dụng*

HĐH: *tiến trình*

Phần cứng: *máy tính*

Tạo, hủy tiến trình,
giao tiếp giữa các tiến trình

Quản lý tài nguyên chung

Tiến trình: Cơ chế và chính sách

Cơ chế:

Tạo, hủy, dừng, context switch, báo hiệu, v,v.

Chính sách:

Các câu hỏi về chính sách cơ bản:

Ai có thể tạo/ hủy/ dừng tiến trình?

Mỗi người dùng có thể chạy bao nhiêu tiến trình đồng thời?

Chúng ta chủ yếu tập trung vào câu hỏi về các chính sách sau:

Làm sao chia sẻ tài nguyên giữa các tiến trình?

Thường chia nhỏ thành các chính sách riêng cho mỗi nguồn tài nguyên riêng như là CPU, memory, và disk.

Biểu trưng của bộ xử lý: Tiểu trình

Tiểu trình đặc trưng cho một bộ xử lý:
tạo ảo giác như là có riêng 1 bộ xử lý cho một thực thi nhất định

Ứng dụng: *một thực thi*

Tạo, hủy, đồng bộ.

HĐH: *tiểu trình*

context switch

Phần cứng: *bộ xử lý*

Tiểu trình: cơ chế và chính sách

Cơ chế:

Tạo, hủy , dừng, context switch, báo hiệu, đồng bộ, v.v.

Chính sách:

Làm sao chia sẻ CPU giữa các tiểu trình thuộc các tiến trình khác nhau?

Làm sao chia sẻ CPU giữa các tiểu trình thuộc cùng một tiến trình?

Làm sao để đồng bộ các tiểu trình với nhau?

Làm sao điều khiển sự liên lạc giữa các tiểu trình?

Liệu một tiểu trình nào đó có thể xâm hại các tiểu trình khác?

Biểu trưng của bộ nhớ: Bộ nhớ logic (*Virtual memory*)

Bộ nhớ logic đặc trưng cho bộ nhớ:
Tạo ảo giác vùng nhớ lớn liên tục, thường là nhiều hơn bộ nhớ vật lý thật sự

Ứng dụng: *không gian địa chỉ*

Địa chỉ logic

HĐH: *bộ nhớ logic*

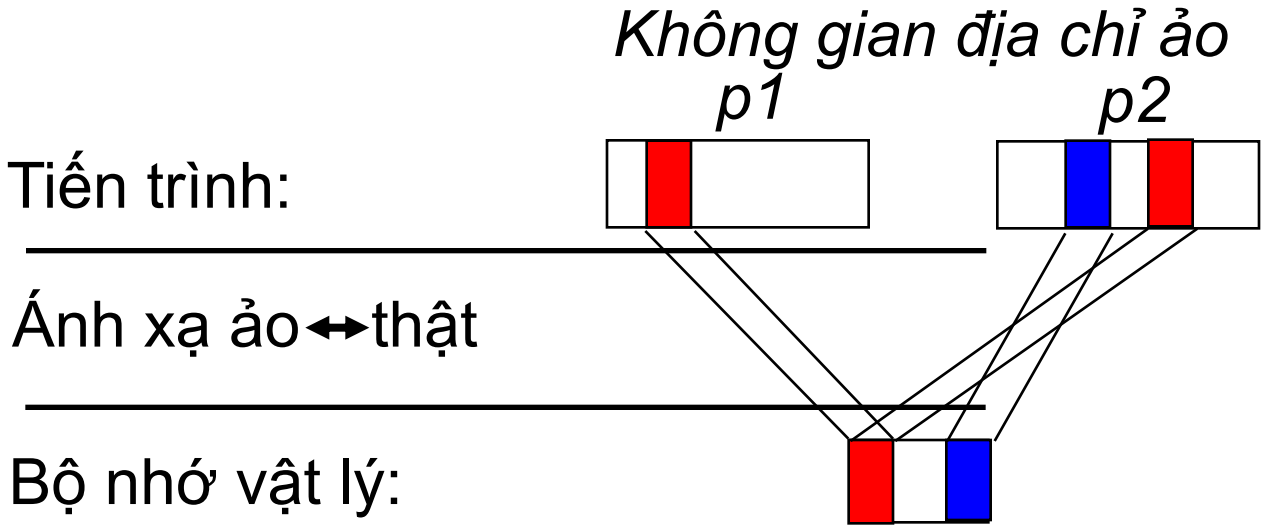
Địa chỉ vật lý

Phần cứng: *bộ nhớ vật lý*

Bộ nhớ ảo: cơ chế

Cơ chế:

Bộ nhớ lôgic ↔ bộ nhớ vật lý, lỗi trang (page-fault), v.v.



Bộ nhớ ảo: chính sách

Chính sách:

Làm sao để biến bộ nhớ ảo trông lớn hơn bộ nhớ vật lý thật dựa trên tài nguyên có sẵn?

Làm sao cấp phát bộ nhớ vật lý cho các tiến trình đang tranh giành nhau?

Làm sao điều phối việc chia sẻ bộ nhớ vật lý giữa các tiến trình?

Biểu trưng cho sự lưu trữ: Hệ thống tập tin

Hệ thống tập tin biểu trưng cho sự lưu trữ: tạo ảo giác một không gian lưu trữ có cấu trúc

ứng dụng/người dùng: *copy file1 file2*

HĐH: *tập tin, thư mục*

Phần cứng: *ổ đĩa*

Đặt tên, bảo vệ, các thao tác với file

các thao tác với blocks của ổ đĩa...

Hệ thống tập tin

Cơ chế:

Tạo tập tin, xóa, đọc, ghi, ánh xạ file-block-to-disk-block, cache vùng đệm của tập tin, ...

Chính sách:

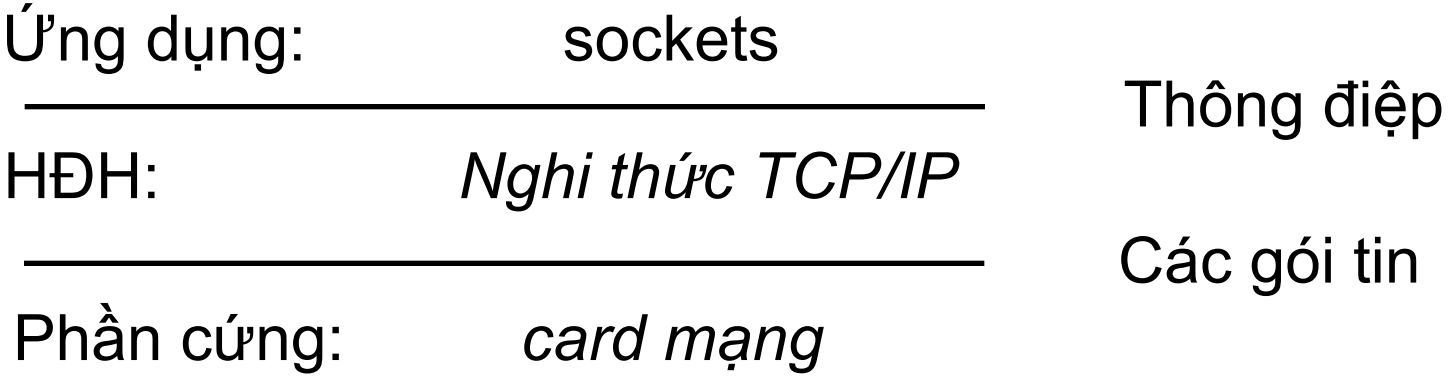
Chia sẻ - so với - bảo vệ?

Cấp phát những block nào?

Quản lý cache vùng đệm của tập tin?

Biểu trưng của truyền thông: Thông điệp (*Messaging*)

Truyền thông điệp biểu trưng cho việc truyền thông:
tạo ảo giác sự truyền thông tin cậy



Truyền thông điệp

Cơ chế:

Gửi, nhận, bộ đệm, truyền lại, v.v.

Chính sách:

Điều khiển tắt nghẽn và dẫn đường

Đảm bảo đa nối kết trên cùng một card mạng

Thiết bị

Giao diện của thiết bị: tạo ảo giác các thiết bị hỗ trợ cùng API – truy cập thông tin, vùng nhớ

Ứng dụng:	*đọc dữ liệu từ thiết bị*	Đọc, ghi, bảo vệ
HĐH:	*API giao tiếp với dữ liệu*	Xác định PIO, xử lý ngắt, truy cập vùng nhớ trực tiếp
Phần cứng	*bàn phím, chuột, v.v.*	

PIO: Programmed Input/Output giao diện giao tiếp giữa CPU và thiết bị ATA

Thiết bị

Cơ chế

Mở, đóng, đọc, ghi, v.v..

Bộ đệm

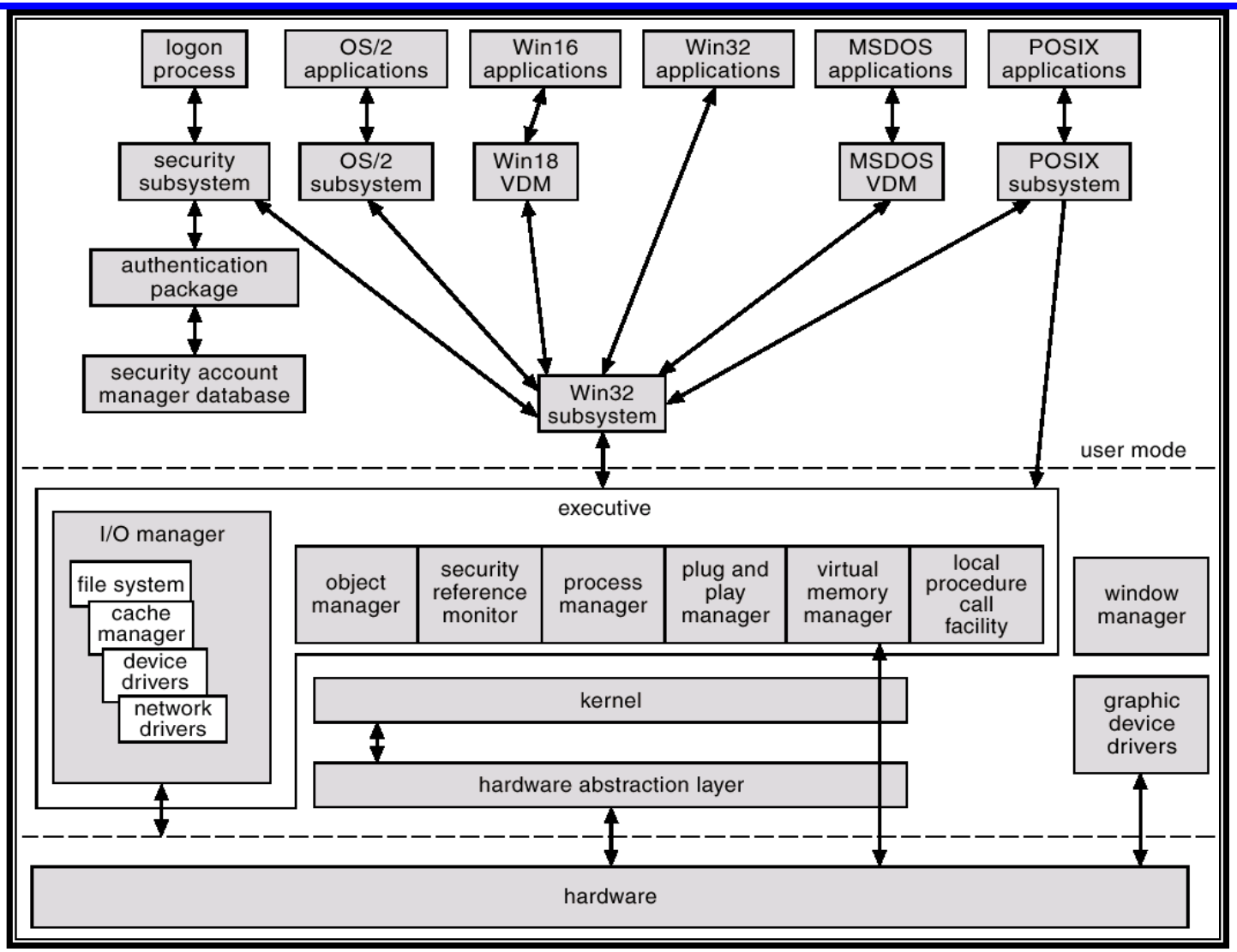
Chính sách

Bảo vệ

Chia sẻ?

Điều phối?

Biểu đồ khối Windows 2000



Các chủ đề chính trong thiết kế hệ điều hành

Giao diện lập trình: máy ảo nên như thế nào?

Quản lý tài nguyên: tài nguyên phần cứng được điều phối giữa nhiều người dùng như thế nào?

Chia sẻ: Tài nguyên được chia sẻ như thế nào?

An toàn: làm sao bảo vệ người dùng? Làm sao bảo vệ chương trình giữa các chương trình khác? Làm sao tạo lớp bảo vệ an toàn cho thiết bị dùng chung từ các ứng dụng và người dùng?

Truyền thông: các ứng dụng trao đổi dữ liệu như thế nào?

Cấu trúc: tổ chức một hệ điều hành như thế nào?

Đồng thời: làm sao để làm các tiến trình thực thi đồng thời?

Các chủ đề chính trong thiết kế hệ điều hành

Thực thi: làm sao để tất cả mọi thứ đều chạy nhanh?

Tin cậy: làm sao để tránh việc "treo", "sụp" hệ thống?

Bền vững: làm sao để đảm bảo dữ liệu nhất quán giữa các chương trình đang thực thi?

Sổ sách: làm sao để lưu vết tài nguyên đang được sử dụng?

Phân tán: làm sao để nhiều máy tính hợp tác cùng làm việc dễ dàng hơn?

Tính phát triển(scaling): làm sao để đảm bảo HĐH vẫn hiệu quả và tin cậy khi khối lượng công việc cũng như phần cứng tăng lên?

Vắn tắt lịch sử HĐH

Trong thời kì đầu, chúng ta thật ra không có HĐH

- Các chương trình nhị phân được nạp sử dụng bộ chuyển
- Giao diện là những đèn nhấp nháy (xịn!)

Tiếp theo là hệ thống theo lô

- HĐH được phát triển và nó tự động làm các công việc theo tuần tự
- HĐH luôn "định cư" trong bộ nhớ
  - Quản lý thường trú
- Người điều khiển đưa cho máy một chuỗi các chương trình và các phân cách
  - Thông thường, nhập vào là một card reader tiếp đó là các phân cách được xem là control cards

Cuộn (Spooling)

CPUs nhanh hơn rất nhiều so với card readers và printers

Ổ cứng ra đời – ổ cứng nhanh hơn nhiều so với card reader

Vậy, chúng ta sẽ làm gì?

Đọc cv 1 từ card vào đĩa. Thực thi cv 1, trong khi đó đọc cv 2 từ cards vào đĩa; lưu kết quả cv 1 vào đĩa. In kết quả công việc 1, trong khi đó thực thi cv 2 và đọc cv 3 từ card vào đĩa. Và tiếp tục như vậy …

Như trên gọi là spooling: Simultaneous Peripheral Operation On-Line

Sử dụng các chữ viết tắt nhưng vẫn tạo ra ngữ nghĩa phù hợp!

Có thể sử dụng nhiều card readers và printers để theo kịp tốc độ CPU

Cải thiện tốc độ xử lý cũng như thời gian phản hồi

Đa chương (Multiprogramming)

CPUs vẫn sẽ nhàn rỗi mỗi khi chương trình thực thi cần giao tiếp với thiết bị ngoại vi

- Đọc dữ liệu từ đĩa

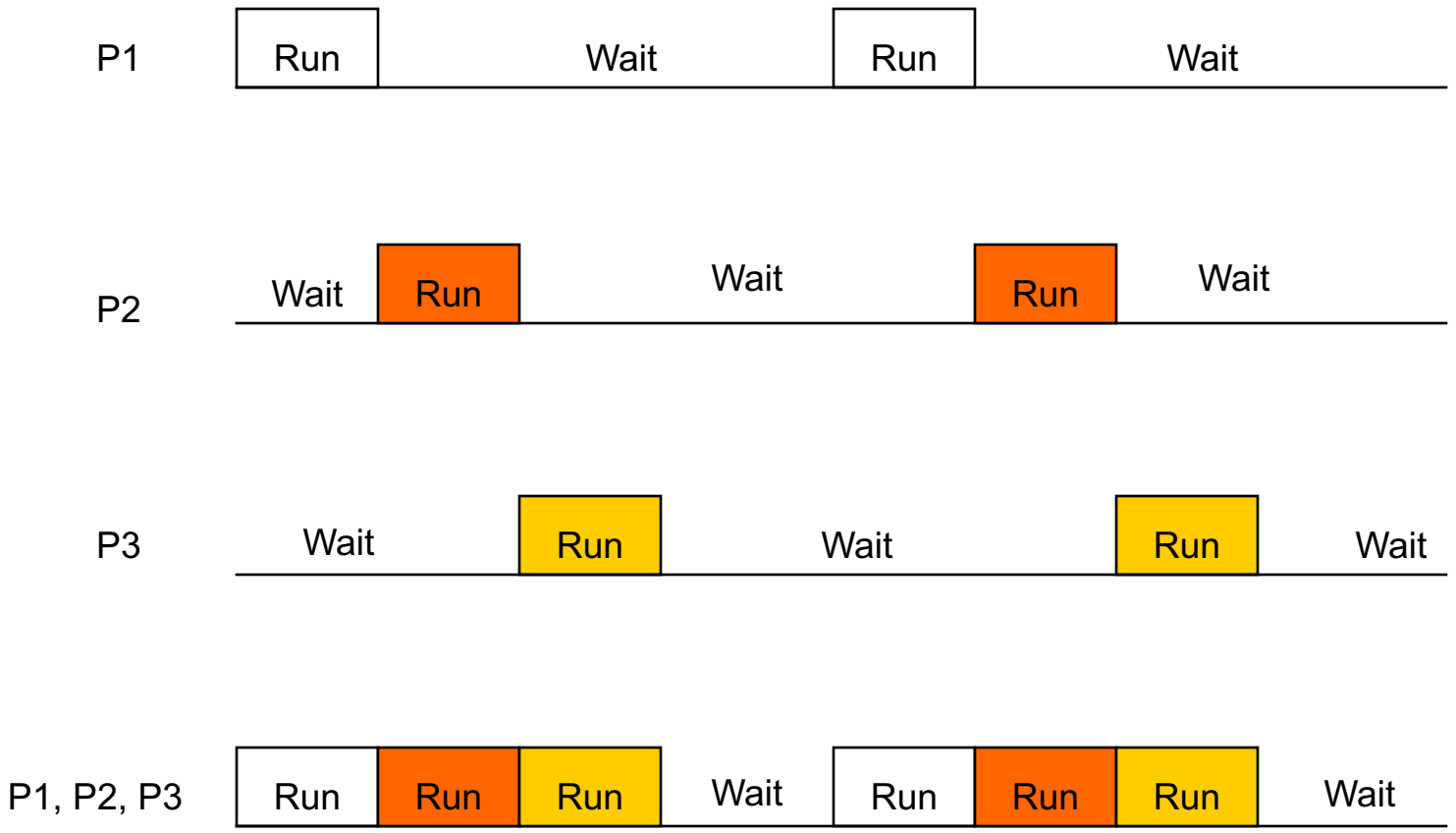
Hệ thống đa chương theo lô (Multiprogrammed batch systems) ra đời

- Nạp nhiều chương trình vào đĩa cùng một thời gian (sau này là vào bộ nhớ)
- Chuyển sang công việc kế tiếp nếu công việc hiện thời đang thực hiện lệnh I/O
  - Thiết bị ngoại vi thường chậm hơn trên đĩa (hay bộ nhớ)
- Đồng thời thực hiện I/O của chương trình này và tính toán cho chương trình khác
- Thiết bị ngoại vi phải là bất đồng bộ
- Phải biết khi nào công việc I/O xong: ngắt vs. polling

Tăng khả năng phục vụ của hệ thống, có thể tốn nhiều thời gian hơn để phản hồi

- Khi nào thì tốt cho thời gian phản hồi? Khi nào thì xấu cho thời gian phản hồi?

Multiprogramming



Chia sẻ thời gian (Time-Sharing)

Các bạn có thể tưởng tượng, theo lô có những hạn chế lớn

Bạn nhập 1 công việc, đợi một lúc, nhận kết quả, thấy lỗi, cố gắng tìm ra chỗ sai, nhập lại công việc, v.v..

Công nghệ mới hơn: có thể kết thúc, có giao diện tương tác

Làm sao chia sẻ cùng 1 máy (nhớ là các máy lúc đó rất đắt) giữa nhiều người dùng và vẫn duy trì giao diện giao tiếp với người dùng?

Chia sẻ thời gian

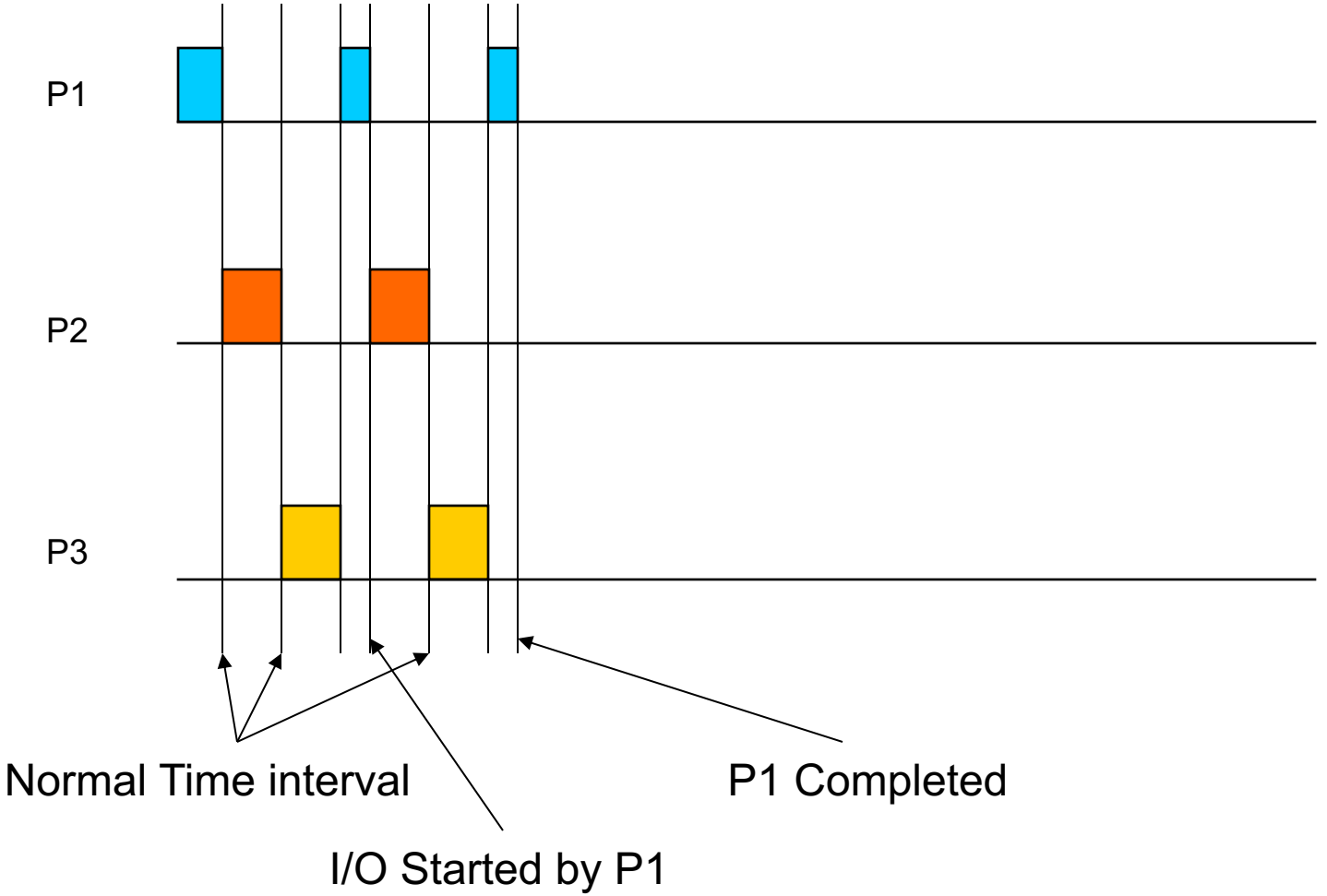
Nối nhiều điều khiển đầu cuối đến một máy

Điều phối 1 máy cho nhiều người dùng

Máy phải đủ nhanh để tạo cảm giác mỗi người dùng đang dùng máy riêng của mình

Multics là hệ thống time-sharing lớn đầu tiên – giữa thập niên-1960's

Time-sharing



Hệ điều hành song song (Parallel OS)

Vài ứng dụng có các công việc có thể thực hiện đồng thời

Dự báo thời tiết, mô phỏng, tính toán lại các bảng tính

Có thể tăng tốc độ bằng cách chạy các công việc trên các bộ xử lý khác nhau song song đồng thời

Cần HĐH và ngôn ngữ lập trình hỗ trợ chia nhỏ công việc thành các hành động song song

Cần HĐH hỗ trợ đồng bộ và truyền thông nhanh

Nhiều kiến trúc song song khác nhau

Khả năng thực thi và tính mở rộng

HĐH thời gian thực (Real-Time OS)

Thực thi các ứng dụng có thời hạn cho trước

Hard real-time system

Hệ thống điều khiển bay, các hệ thống điều khiển công nghiệp, v.v..

Gây thảm họa nếu ta trễ hạn

Thách thức là làm sao không trễ hạn mà không phung phí nhiều tài nguyên

Soft real-time system

Ứng dụng multimedia

Có thể gây khó chịu nhưng không đến nỗi thảm họa nếu bị quá hạn đôi chút!

Thách thức là làm sao không trễ hạn mà không phung phí nhiều tài nguyên

Thử thách ở chỗ là khi hệ thống quá tải

HĐH phân tán (Distributed OS)

Clustering

Dùng nhiều máy nhỏ để phục vụ các công việc lớn

- Rẻ hơn là dùng một máy tính lớn
- Độ tin cậy cao hơn, tăng khả năng mở rộng

Hệ thống phân tán trong khu vực rộng

Cho phép sử dụng tài nguyên phân tán

- V.d. sử dụng PC để truy cập Web
- Không cần mang nhiều thông tin cần thiết theo mình

Cần HĐH hỗ trợ truyền thông và chia sẻ tài nguyên phân tán

V.d., hệ lưu trữ tập tin trên mạng

Quan tâm tính thực thi (mặc dù tăng tốc không phải là mục tiêu của HĐH này), độ tin cậy cao, sử dụng nguồn tài nguyên đa dạng

HĐH nhúng (Embedded OS)

Phát triển rộng khắp

Hiện thời, ĐTDĐ và PDAs

Tương lai, các thiết bị tính toán ở mọi nơi

Đặc tính

Tài nguyên hạn hẹp: CPU chậm, bộ nhớ nhỏ, không ổ đĩa, v.v.

Chúng ta cần chạy những ứng dụng mạnh hơn!..

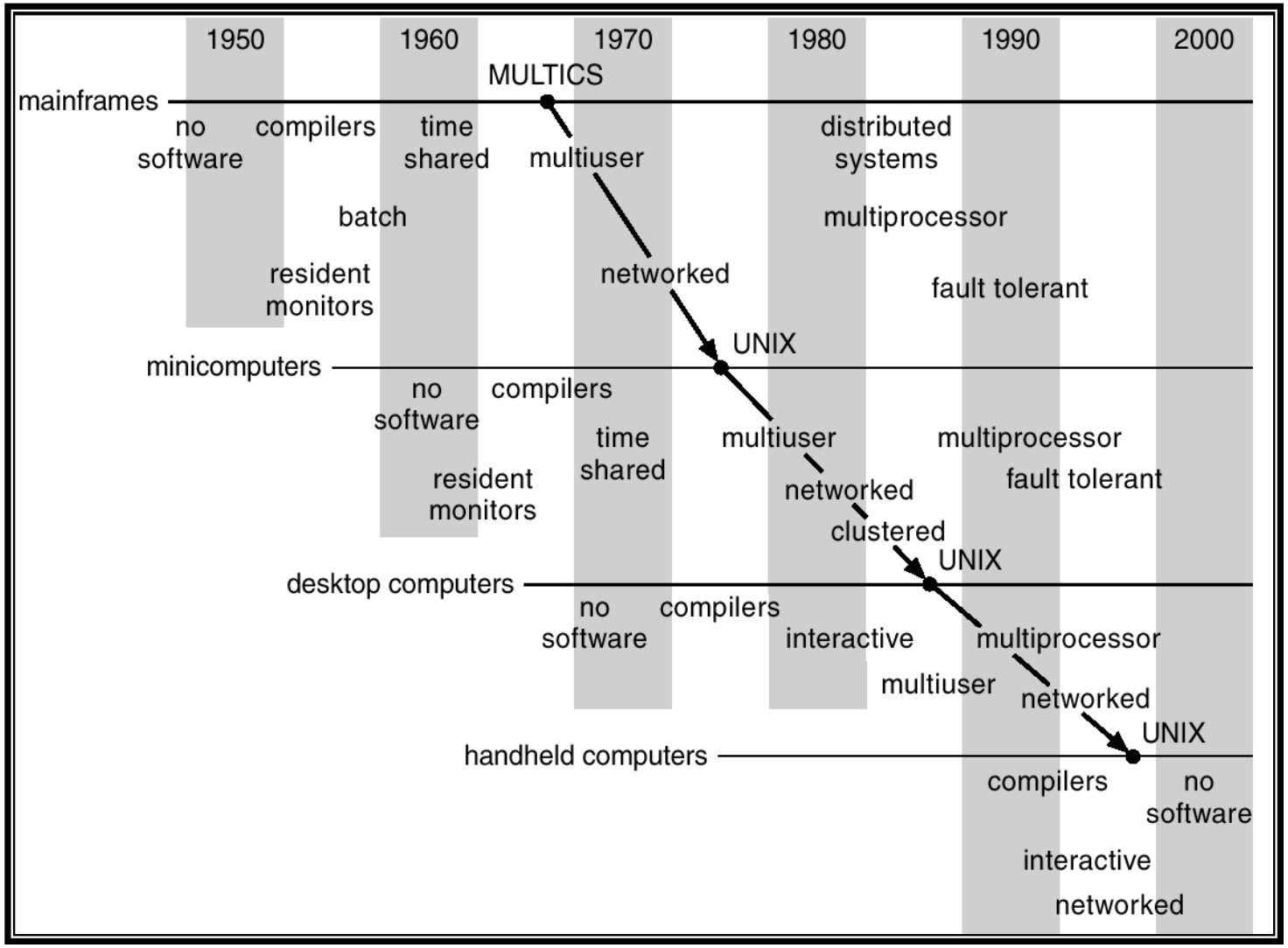
Làm sao ta có thể chạy các chương trình mạnh hơn khi mà phần cứng vẫn như trước đây?

Sử dụng nhiều thiết bị…

Tăng thêm các tiện ích trên thiết bị

HĐH giúp quản lý năng lượng, tính di động, tìm kiếm tài nguyên, v.vv.

Quá trình phát triển các khái niệm và tính năng HĐH



Bài tập

- Hãy so sánh HĐH và các phần mềm khác dựa vào các tiêu chí sau:
  - Khả năng tự hoạt động ngay sau khi bật máy
  - Tác động đến máy tính khi chương trình kết thúc
  - Mức độ cần thiết cho sự hoạt động tối thiểu của máy tính
  - Khả năng điều khiển phần cứng
  - Độ phức tạp
  - Thứ tự cài đặt
  - Mức độ sử dụng
  - Hình thức quản lý
  - Số lượng cài đặt trên mỗi máy tính
- Hãy so sánh ĐTDĐ có sử dụng HĐH và không sử dụng HĐH

44

Gợi ý

- Khả năng tự hoạt động ngay sau khi bật máy: HĐH thường có khả năng này còn các phần mềm khác thì không.
- Tác động đến máy tính khi chương trình kết thúc: khi kết thúc HĐH thì máy tính không sử dụng được nữa (và HĐH thường tự thực hiện luôn thao tác tắt máy), các phần mềm khác không như vậy.
- Mức độ cần thiết cho sự hoạt động tối thiểu của máy tính: HĐH là phần mềm bắt buộc phải có, các phần mềm khác thì không tới mức bắt buộc.
- Khả năng điều khiển phần cứng: Các phần mềm khác không điều hành trực tiếp các thiết bị phần cứng (trong 1 số trường hợp hiếm hoi thì cũng có – nhưng khi đó chỉ điều hành 1 vài thiết bị), còn HĐH điều hành tất cả các thiết bị phần cứng.
- Độ phức tạp: HĐH thường được thiết kế công phu, phức tạp hơn các phần mềm khác.

45

Gợi ý (tt)

- Thứ tự cài đặt: HĐH phải được cài đặt vào máy tính trước các phần mềm khác.
- Mức độ sử dụng: người dùng máy tính nào cũng phải sử dụng HĐH, còn những phần mềm khác thì người có người không
- Hình thức quản lý: Các phần mềm khác không quản lý, điều hành HĐH mà là ngược lại: HĐH quản lý, điều hành các phần mềm khác.
- Số lượng cài đặt trên mỗi máy tính: mỗi máy tính cao cấp thường chỉ có 1 hoặc vài HĐH, nhưng thường có rất nhiều phần mềm
- Số lượng hoạt động trên mỗi máy tính: mỗi thời điểm chỉ có 1 HĐH hoạt động nhưng thường có nhiều phần mềm đang chạy cùng lúc.

46