CHƯƠNG 1

Giới thiệu về Hệ điều hành

Hệ điều hành (operating system) là một phần mềm quản lý phần cứng của máy tính. Nó cũng cung cấp nền tảng cơ sở cho các chương trình ứng dụng và đóng vai trò trung gian giữa người sử dụng máy tính và phần cứng máy tính. Một khía cạnh đáng kinh ngạc của hệ điều hành là cách chúng khác nhau trong việc hoàn thành các tác vụ phần mềm trong rất nhiều môi trường máy tính khác nhau. Hệ điều hành có ở khắp mọi nơi, từ ô tô và thiết bị gia dụng bao gồm thiết bị Internet vạn vật – "Internet of Things" – , đến điện thoại thông minh, máy tính cá nhân, máy tính doanh nghiệp và môi trường điện toán đám mây.

Để khám phá vai trò của hệ điều hành trong môi trường máy tính hiện đại, điều quan trọng đầu tiên là phải hiểu tổ chức và kiến trúc của phần cứng máy tính. Tổ chức này này bao gồm CPU, bộ nhớ chính và các thiết bị Nhập/Xuất cũng như bộ nhớ lưu trữ dài hạn. Trách nhiệm cơ bản của hệ điều hành là phân bổ các tài nguyên này cho các chương trình.

Bởi vì một hệ điều hành lớn và phức tạp, nó phải được tạo ra từ từng phần nhỏ. Mỗi phần này phải là một mảnh ghép được mô tả rõ ràng của hệ thống, với các đầu vào, đầu ra và chức năng được xác định cẩn thận. Trong chương này, chúng tôi cung cấp một cái nhìn tổng quát về các thành phần chính của một hệ thống máy tính đương đại cũng như các chức năng được cung cấp bởi hệ điều hành. Ngoài ra, chúng tôi đề cập đến một số chủ đề để giúp tạo tiền đề cho phần còn lại của quyển sách: cấu trúc dữ liệu được sử dụng trong hệ điều hành, môi trường máy tính và hệ điều hành mã nguồn mở và miễn phí.

MỤC TIÊU CHƯƠNG

- Mô tả tổ chức tổng quát của hệ thống máy tính và vai trò của các tín hiệu ngắt.
- Mô tả các thành phần trong một hệ thống máy tính đa xử lý hiện đại.
- Minh họa sự chuyển đổi từ chế độ người dùng sang chế độ nhân.
- Thảo luận về cách hệ điều hành được sử dụng trong các môi trường máy tính khác nhau.
- Cung cấp các ví dụ về hệ điều hành mã nguồn mở và miễn phí.

Mục lục

1.1 Hệ điều hành làm gì

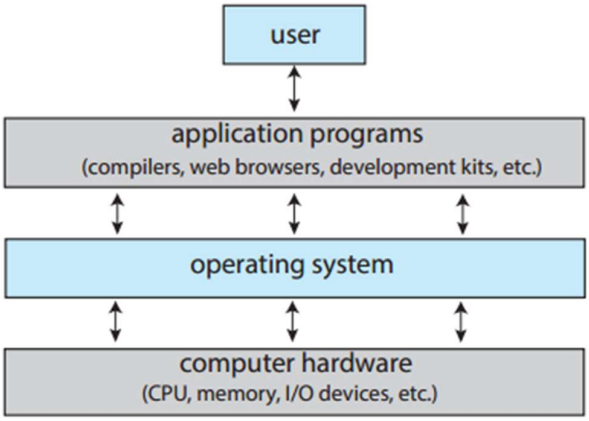
Chúng ta bắt đầu cuộc thảo luận của mình bằng cách xem xét vai trò của hệ điều hành trong hệ thống máy tính tổng thể. Một hệ thống máy tính có thể được chia thành bốn thành phần: ***phần cứng***, ***hệ điều hành***, ***chương trình ứng dụng*** và ***người dùng*** (Hình 1.1). **Phần cứng (hardware)** – đơn vị xử lý trung tâm (CPU), bộ nhớ và thiết bị Nhập/Xuất (Nhập/Xuất) – cung cấp các tài nguyên máy tính cơ bản cho hệ thống. Các chương trình ứng dụng (application programs) – chẳng hạn như trình xử lý văn bản, bảng tính, trình biên dịch và trình duyệt web – xác định cách thức sử dụng các tài nguyên này để giải quyết các vấn đề tính toán của người dùng. Hệ điều hành kiểm soát phần cứng và điều phối việc sử dụng nó giữa các chương trình ứng dụng khác nhau cho nhiều người dùng khác nhau.

Chúng ta cũng có thể xem một hệ thống máy tính bao gồm phần cứng, phần mềm và dữ liệu. Hệ điều hành cung cấp các phương tiện để sử dụng hợp lý các tài nguyên này trong hoạt động của hệ thống máy tính. Một hệ điều hành tương tự như một ban quản trị, tự họ không trực tiếp thực hiện chức năng hữu ích nào. Họ chỉ đơn giản là cung cấp một môi trường trong đó các chương trình khác có thể thực hiện công việc hữu ích.

Để hiểu đầy đủ hơn về vai trò của hệ điều hành, tiếp theo, chúng ta sẽ khám phá hệ điều hành từ hai góc nhìn: quan điểm của người dùng và quan điểm của hệ thống.

1.1.1 Góc nhìn người dùng

Góc nhìn máy tính của người dùng thay đổi tùy theo giao diện đang được sử dụng. Nhiều người dùng máy tính đang ngồi với máy tính xách tay hoặc ngồi trước PC bao gồm màn hình, bàn phím và chuột. Một hệ thống như vậy được thiết kế để một người dùng độc quyền tài nguyên của nó. Mục đích là tối đa hóa công việc (hoặc giải trí) mà người dùng đang thực hiện. Trong trường hợp này, hệ điều hành được thiết kế chủ yếu để **dễ sử dụng (easa of use)**, với một số chú ý đến hiệu suất và bảo mật và không chú ý đến việc **khai thác tài nguyên (resource utilization)** – cách mà các tài nguyên phần cứng và phần mềm khác nhau được chia sẻ với nhau.



Hình 1.1 Bản vẽ chi tiết về các thành phần của hệ thống máy tính.

Càng ngày, nhiều người dùng càng tương tác với các thiết bị di động như điện thoại thông minh và máy tính bảng – những thiết bị đang thay thế hệ thống máy tính để bàn và máy tính xách tay đối với một số người dùng. Các thiết bị này thường được kết nối mạng thông qua công nghệ di động hoặc công nghệ không dây khác. Giao diện người dùng cho máy tính di động thường có **màn hình cảm ứng (touch screen)**, nơi người dùng tương tác với hệ thống bằng cách nhấn và vuốt ngón tay trên màn hình chứ không phải sử dụng bàn phím và chuột vật lý. Nhiều thiết bị di động cũng cho phép người dùng tương tác thông qua giao diện **nhận dạng giọng nói (voice recognition)**, chẳng hạn như Apple's **Siri**.

Một số máy tính có rất ít hoặc không có giao diện người dùng. Ví dụ:các **máy tính nhúng (embedded computer)** trong thiết bị gia đình và ô tô có thể có bàn phím số và có thể bật hoặc tắt đèn báo để hiển thị trạng thái, nhưng chúng cũng như các hệ điều hành và ứng dụng của chúng được thiết kế chủ yếu để chạy mà không có sự can thiệp của người dùng.

1.1.2 Góc nhìn hệ thống

Từ góc nhìn của máy tính, hệ điều hành là chương trình liên quan mật thiết nhất đến phần cứng. Trong bối cảnh này, chúng ta có thể xem hệ điều hành như một **bộ phân bổ tài nguyên (resource allocator)**. Hệ thống máy tính có nhiều tài nguyên có thể được yêu cầu để giải quyết một vấn đề: thời gian dùng CPU, không gian bộ nhớ, không gian lưu trữ, thiết bị Nhập/Xuất , v.v. Hệ điều hành đóng vai trò như là người quản lý các tài nguyên này. Đối mặt với nhiều yêu cầu và có thể xung đột về tài nguyên, hệ điều hành phải quyết định cách phân bổ chúng cho các chương trình và người dùng cụ thể để có thể vận hành hệ thống máy tính một cách hiệu quả và công bằng.

Một quan điểm hơi khác về hệ điều hành nhấn mạnh sự cần thiết phải kiểm soát các thiết bị Nhập/Xuất và các chương trình khác nhau của người dùng. Hệ điều hành là một chương trình điều khiển. Một **chương trình điều khiển (control program)** quản lý việc thực thi các chương trình của người dùng để ngăn ngừa lỗi và việc sử dụng máy tính không đúng cách. Nó đặc biệt quan tâm đến hoạt động và điều khiển của các thiết bị Nhập/Xuất .

1.1.3 Định nghĩa hệ điều hành

Bây giờ, bạn có thể thấy rằng thuật ngữ ***hệ điều hành*** bao gồm nhiều vai trò và chức năng. Đó là do vô số thiết kế và cách sử dụng của máy tính. Máy tính có mặt trong lò nướng bánh mì, ô tô, tàu thủy, tàu vũ trụ, gia đình và doanh nghiệp. Chúng là cơ sở cho máy trò chơi, bộ chỉnh truyền hình cáp và hệ thống điều khiển công nghiệp.

Để giải thích sự đa dạng này, chúng ta có thể nhìn sang lịch sử của máy tính. Mặc dù máy tính có lịch sử tương đối ngắn, nhưng chúng đã phát triển nhanh chóng. Máy tính bắt đầu như một thử nghiệm để xác định những gì có thể được hiện thực và nhanh chóng chuyển sang các hệ thống có mục đích nhất định cho quân sự, chẳng hạn như phá mã và vẽ biểu đồ quỹ đạo, và các mục đích sử dụng của chính phủ, chẳng hạn như tính toán điều tra dân số. Những máy tính ban đầu đó đã phát triển thành các máy tính lớn đa chức năng, đa dụng, và đó là khi hệ điều hành ra đời. Vào những năm 1960, **Định luật Moore (Moore's Law)** đã dự đoán rằng số lượng bóng bán dẫn trên một mạch tích hợp sẽ tăng gấp đôi sau mỗi 18 tháng và dự đoán đó đã đúng. Máy tính được nâng cấp về chức năng và thu nhỏ về kích thước, dẫn đến việc sử dụng

rất nhiều và một số lượng lớn và đa dạng các hệ điều hành. (Xem **Phụ lục A** để biết thêm chi tiết về lịch sử của hệ điều hành.)

TẠI SAO NGHIÊN CỨU HỆ ĐIỀU HÀNH?

Mặc dù có nhiều người tham gia vào khoa học máy tính, nhưng chỉ một phần nhỏ trong số họ sẽ tham gia vào việc tạo ra hoặc sửa đổi hệ điều hành. Vậy tại sao lại nghiên cứu hệ điều hành và cách chúng hoạt động? Đơn giản vì hầu như tất cả các mã đều chạy trên hệ điều hành, kiến thức về cách hoạt động của hệ điều hành là rất quan trọng để lập trình phù hợp, đạt kết quả, hiệu suất cao và an toàn. cung cấp cho các ứng dụng không chỉ cần thiết cho những người lập trình chúng mà còn rất hữu ích cho những người viết chương trình trên chúng và sử dụng chúng.

Sau đó, chúng ta có thể định nghĩa hệ điều hành là gì? Nói chung, chúng ta không có định nghĩa hoàn toàn đầy đủ về hệ điều hành. Hệ điều hành tồn tại bởi vì chúng đưa ra một cách hợp lý để giải quyết vấn đề tạo ra hệ thống máy tính có thể sử dụng được. Mục tiêu cơ bản của hệ thống máy tính là thực thi các chương trình và giúp giải quyết các vấn đề của người dùng dễ dàng hơn. Phần cứng máy tính được xây dựng để hướng tới mục tiêu này. Vì việc sử dụng các phần cứng nguyên bản không dễ dàng gì, các chương trình ứng dụng được phát triển. Các chương trình này yêu cầu một số hoạt động phổ biến nhất định, chẳng hạn như các hoạt động điều khiển thiết bị Nhập/Xuất . Các chức năng chung của việc kiểm soát và phân bổ tài nguyên sau đó được tập hợp lại thành một phần mềm: hệ điều hành.

Ngoài ra, chúng ta không có định nghĩa được chấp nhận rộng rãi về những thành phần nào thuộc về hệ điều hành. Một quan điểm đơn giản là nó bao gồm mọi thứ mà nhà cung cấp gửi cho bạn khi đặt mua "hệ điều hành". Nhiều tính năng bao gồm, tuy nhiên chúng rất khác nhau giữa các hệ thống. Một số hệ thống chiếm ít hơn một megabyte dung lượng và thậm chí thiếu trình soạn thảo toàn màn hình, trong khi những hệ thống khác yêu cầu đến (vài) gigabyte dung lượng và hoàn toàn hoạt động trên hệ thống cửa sổ đồ họa. Một định nghĩa hài hước hơn, và định nghĩa mà chúng ta thường làm theo, đó là hệ điều hành là một chương trình chạy mọi lúc trên máy tính –thường được gọi là **nhân (kernel)**. Cùng với hạt nhân, có hai loại chương trình khác: **chương trình hệ thống (system programs)**, được liên kết với hệ điều hành nhưng không nhất thiết là một phần của nhân và chương trình ứng dụng, bao gồm tất cả các chương trình không liên quan đến hoạt động của hệ thống.

Vấn đề cấu thành hệ điều hành ngày càng trở nên quan trọng khi máy tính cá nhân ngày càng phổ biến và hệ điều hành ngày càng tinh vi. Năm 1998, Bộ Tư pháp Hoa Kỳ đã đệ đơn kiện Microsoft, về bản chất, tuyên bố rằng Microsoft đã đưa quá nhiều chức năng vào hệ điều hành của mình và do đó ngăn cản các nhà cung cấp ứng dụng cạnh tranh. (Ví dụ: trình duyệt web là một phần không thể thiếu trong hệ điều hành của Microsoft.) Do đó, Microsoft bị kết tội sử dụng độc quyền hệ điều hành của mình để hạn chế cạnh tranh.

Tuy nhiên, ngày nay nếu chúng ta nhìn vào các hệ điều hành dành cho thiết bị di động, chúng ta sẽ thấy rằng một lần nữa số lượng các tính năng cấu thành hệ điều hành ngày càng tăng lên. Hệ điều hành di động thường không chỉ bao gồm nhân nhân mà còn bao gồm **phần**

mềm trung gian (middleware) – một tập hợp các khung phần mềm cung cấp các dịch vụ bổ sung cho các nhà phát triển ứng dụng. Ví dụ: trong mỗi hệ điều hành di động nổi bật nhất – iOS của Apple và Android của Google – đều có nhân nòng cốt cùng với phần mềm trung gian hỗ trợ cơ sở dữ liệu, đa phương tiện và đồ họa.

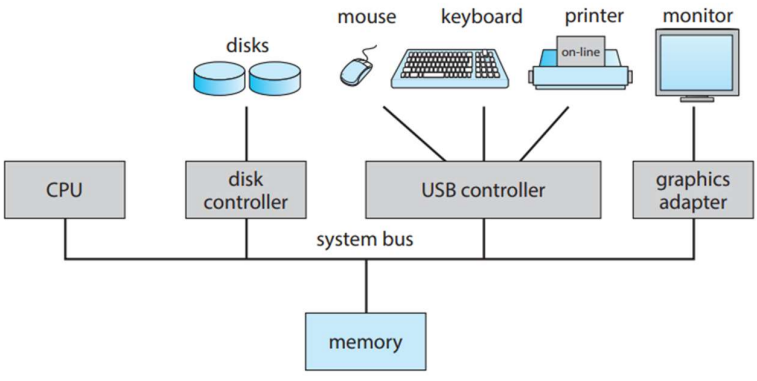
Tóm lại, theo mục đích của chúng ta, hệ điều hành bao gồm nhân luôn chạy, các khung phần mềm trung gian giúp dễ dàng phát triển ứng dụng và cung cấp các tính năng, và các chương trình hệ thống hỗ trợ quản lý hệ thống khi nó đang chạy. Phần lớn nội dung này liên quan đến nhân của các hệ điều hành đa năng, nhưng các thành phần khác sẽ được thảo luận khi cần thiết để giải thích đầy đủ về thiết kế và hoạt động của hệ điều hành.

1.2 Tổ chức hệ thống máy tính

Một hệ thống máy tính đa năng hiện đại bao gồm một hoặc nhiều CPU và một số bộ điều khiển thiết bị được kết nối thông qua một **liên kết chung (bus)** cung cấp quyền truy cập giữa các thành phần và bộ nhớ dùng chung (Hình 1.2). Mỗi bộ điều khiển thiết bị phụ trách một loại thiết bị cụ thể (ví dụ: ổ đĩa, thiết bị âm thanh hoặc màn hình đồ họa). Tùy thuộc vào bộ điều khiển, nhiều hơn một thiết bị có thể được gắn vào. Ví dụ, một cổng USB của hệ thống có thể kết nối với một bộ chia USB, một số thiết bị có thể kết nối với nhau. Bộ điều khiển thiết bị duy trì một số lưu trữ bộ đệm cục bộ và một tập hợp các thanh ghi có mục đích đặc biệt. Bộ điều khiển thiết bị chịu trách nhiệm di chuyển dữ liệu giữa các thiết bị ngoại vi mà nó điều khiển và bộ nhớ đệm cục bộ của nó.

Thông thường, các hệ điều hành có **trình điều khiển thiết bị (device driver)** cho mỗi bộ điều khiển thiết bị. Trình điều khiển thiết bị này hiểu bộ điều khiển thiết bị và cung cấp cho phần còn lại của hệ điều hành một giao diện thống nhất cho thiết bị. CPU và bộ điều khiển thiết bị có thể thực thi song song, cạnh tranh nhau về chu kỳ bộ nhớ. Để đảm bảo truy cập có trật tự vào bộ nhớ dùng chung, bộ điều khiển bộ nhớ sẽ đồng bộ hóa quyền truy cập vào bộ nhớ.

Trong tiểu mục con sau đây, chúng tôi mô tả một số điều cơ bản về cách thức hoạt động của một hệ thống như vậy, tập trung vào ba khía cạnh chính của hệ thống. Chúng tôi bắt đầu với các ngắt, cảnh báo CPU về các sự kiện cần chú ý. Sau đó chúng ta thảo luận về cấu trúc lưu trữ và cấu trúc Nhập/Xuất .



Hình 1.2 Một hệ thống máy tính điển hình.

1.2.1 Ngắt

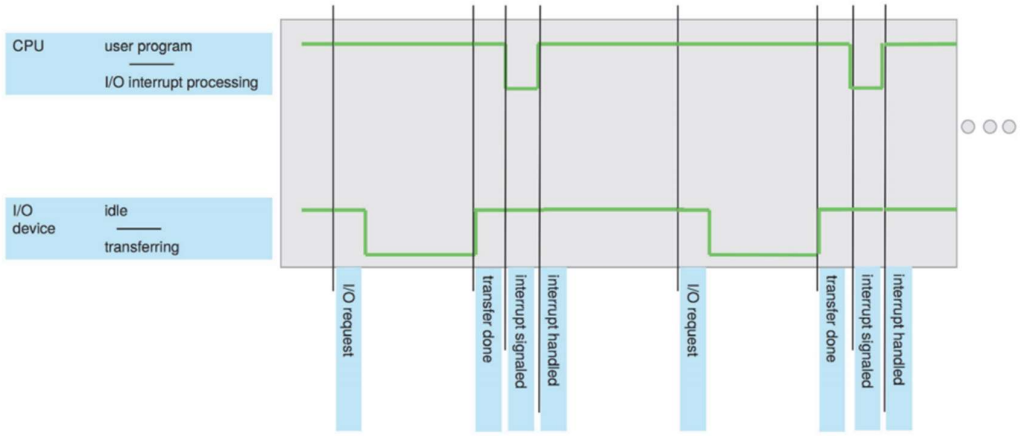
Hãy xem xét một hoạt động máy tính điển hình: một chương trình thực hiện Nhập/Xuất. Để bắt đầu hoạt động Nhập/Xuất, trình điều khiển thiết bị tải các giá trị từ các thanh ghi liên quan trong bộ điều khiển thiết bị. Đến lượt mình, bộ điều khiển thiết bị sẽ kiểm tra giá trị của các thanh ghi này để xác định hành động cần thực hiện (chẳng hạn như "đọc một ký tự từ bàn phím"). Bộ điều khiển bắt đầu chuyển dữ liệu từ thiết bị đến bộ đệm cục bộ của nó. Khi quá trình truyền dữ liệu hoàn tất, bộ điều khiển thiết bị sẽ thông báo cho trình điều khiển thiết bị rằng nó đã kết thúc hoạt động. Sau đó, trình điều khiển thiết bị cung cấp quyền kiểm soát cho các phần khác của hệ điều hành, có thể trả về dữ liệu hoặc một con trỏ tới dữ liệu nếu thao tác được đọc. Đối với các hoạt động khác, trình điều khiển thiết bị trả về thông tin trạng thái như "đã ghi thành công" hoặc "thiết bị bận". Nhưng làm thế nào để bộ điều khiển thông báo cho trình điều khiển thiết bị rằng nó đã kết thúc hoạt động? Điều này được thực hiện thông qua một **ngắt (interrupt)**.

1.2.1.1 Tổng quan

Phần cứng có thể kích hoạt ngắt bất kỳ lúc nào bằng cách gửi tín hiệu đến CPU, thường là qua đường bus hệ thống. (Có thể có nhiều bus trong một hệ thống máy tính, nhưng bus hệ thống là đường truyền thông tin chính giữa các thành phần chính.) Ngắt cũng được sử dụng cho nhiều mục đích khác và là một phần quan trọng trong cách hệ điều hành và phần cứng tương tác.

Khi CPU bị ngắt, nó sẽ dừng những gì nó đang thực thi và ngay lập tức chuyển việc thực thi đến một vị trí cố định, nơi thường chứa địa chỉ bắt đầu của tiến trình dịch vụ dành cho ngắt. Tiến trình dịch vụ ngắt sẽ thực thi; khi hoàn thành, CPU tiếp tục quá trình tính toán bị gián đoạn trước đó. Diễn biến của hoạt động này được hiển thị trong Hình 1.3.

Ngắt là một phần quan trọng của kiến trúc máy tính. Mỗi thiết kế máy tính có cơ chế ngắt riêng, nhưng một số chức năng là chung. Ngắt phải chuyển quyền điều khiển sang tiến trình phục vụ ngắt thích hợp. Phương pháp đơn giản để quản lý việc chuyển giao này là gọi một tiến trình chung để kiểm tra thông tin ngắt. Đến lượt mình, tiến trình sẽ gọi trình xử lý ngắt cụ thể. Tuy nhiên, các ngắt phải được xử lý nhanh chóng, vì chúng xảy ra rất thường xuyên. Thay vào đó, một bảng con trỏ để ngắt các tiến trình có thể được sử dụng để thao tác nhanh như cần thiết. Tiến trình ngắt được gọi gián tiếp thông qua bảng, không cần tiến trình trung gian. Nói chung, bảng con trỏ được lưu trữ trong bộ nhớ địa chỉ thấp (hàng trăm vị trí đầu tiên hoặc nhiều hơn). Các vị trí này giữ địa chỉ của các tiến trình phục vụ ngắt cho các thiết bị khác nhau. Mảng địa chỉ này, hoặc còn gọi là **vectơ ngắt (interrupt vector)**, sau đó được lập chỉ mục bằng một giá trị số duy nhất, được cung cấp cùng với yêu cầu ngắt, để cung cấp địa chỉ của tiến trình phục vụ ngắt cho thiết bị ngắt. Các hệ điều hành khác nhau như Windows và UNIX điều phối các ngắt theo cách này.



Hình 1.3 Diễn biến thời gian của ngắt cho một chương trình duy nhất đang thực hiện xuất ra.

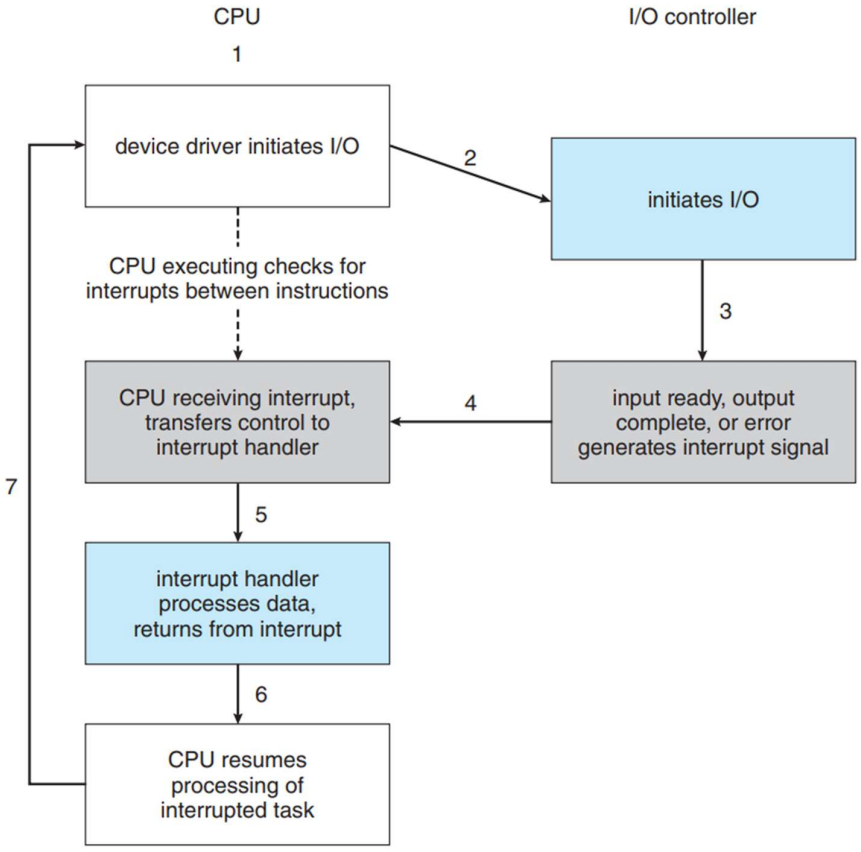
Kiến trúc ngắt cũng phải lưu thông tin trạng thái của bất kỳ thứ gì bị ngắt, để nó có thể khôi phục thông tin này sau khi hoàn thành phục vụ ngắt. Nếu tiến trình ngắt cần sửa đổi trạng thái của bộ xử lý – ví dụ, sửa đổi các giá trị thanh ghi – thì nó phải lưu trạng thái hiện tại một cách rõ ràng rồi khôi phục trạng thái đó trước khi quay trở lại. Sau khi ngắt được phục vụ, địa chỉ trả về đã lưu được tải vào bộ đếm chương trình (thanh ghi PC) và quá trình tính toán bị ngắt tiếp tục như thể ngắt chưa xảy ra.

1.2.1.2 Hiện thực

Cơ chế ngắt cơ bản hoạt động như sau. Phần cứng CPU có một dây gọi là **dây yêu cầu ngắt (interrupt–request line)** mà CPU cảm nhận được sau khi thực hiện mọi lệnh. Khi CPU phát hiện thấy bộ điều khiển đã xác nhận một tín hiệu trên dòng yêu cầu ngắt, nó sẽ đọc số ngắt và chuyển đến **tiến trình xử lý ngắt (interrupt–handler routine)** bằng cách sử dụng số ngắt đó làm chỉ số vào vectơ ngắt. Sau đó, nó bắt đầu thực thi tại địa chỉ được liên kết với chỉ mục đó. Trình xử lý ngắt lưu bất kỳ trạng thái nào mà nó sẽ thay đổi trong quá trình hoạt động, xác định nguyên nhân của ngắt, thực hiện xử lý cần thiết, thực hiện khôi phục trạng thái và thực hiện lệnh `return_from_interrupt` (quay về từ ngắt) để đưa CPU về trạng thái thực thi trước khi ngắt . Chúng ta nói rằng bộ điều khiển thiết bị tạo ra (raise) một ngắt bằng cách xác nhận một tín hiệu trên dây yêu cầu ngắt, CPU bắt (catches) ngắt và gửi (dispatches) nó đến trình xử lý ngắt và trình xử lý xóa (clear) ngắt sau khi thiết bị phục vụ. Hình 1.4 tóm tắt chu trình Nhập/Xuất bằng điều khiển ngắt. Cơ chế ngắt cơ bản vừa mô tả cho phép CPU phản hồi với sự kiện không đồng bộ, như khi bộ điều khiển thiết bị sẵn sàng hoạt động. Tuy nhiên, trong một hệ điều hành hiện đại, chúng ta cần các tính năng xử lý ngắt phức tạp hơn.

1. Chúng ta cần khả năng trì hoãn việc xử lý ngắt trong tiến trình xử lý quan trọng.
2. Chúng ta cần một cách hiệu quả để điều phối trình xử lý ngắt thích hợp cho một thiết bị.
3. Chúng ta cần các ngắt đa cấp để hệ điều hành có thể phân biệt giữa các ngắt có mức độ ưu tiên cao và thấp và có thể đáp ứng với mức độ khẩn cấp thích hợp.

Trong phần cứng máy tính hiện đại, ba tính năng này được cung cấp bởi CPU và **phần cứng bộ điều khiển ngắt (interupt–controller hardware)**.



Hình 1.4 Chu trình xử lý ngắt Nhập/Xuất.

Hầu hết các CPU đều có hai dây yêu cầu ngắt. Một là **ngắt không che được (nonmaskable interrupt)**, được dành riêng cho các sự kiện như lỗi bộ nhớ không thể khôi phục. Dây ngắt thứ hai **có thể che được (maskable)**: nó có thể được CPU tắt trước khi thực hiện các chuỗi lệnh quan trọng mà ngắt không được phép xảy ra. Ngắt có thể che được được sử dụng bởi bộ điều khiển thiết bị để yêu cầu dịch vụ.

Nhắc lại rằng mục đích của cơ chế ngắt bằng vectơ là giảm nhu cầu về một trình xử lý ngắt đơn lẻ để tìm kiếm trong tất cả các nguồn ngắt có thể có cái ngắt nào cần phục vụ. Tuy nhiên, trong thực tế, máy tính có nhiều thiết bị (và do đó, cũng có nhiều trình xử lý ngắt) hơn là số lượng phần tử lưu địa chỉ trong vectơ ngắt. Một cách phổ biến để giải quyết vấn đề này là sử dụng **chuỗi ngắt (interrupt chaining)**, trong đó mỗi phần tử trong vectơ ngắt trỏ đến phần đầu của danh sách các trình xử lý ngắt. Khi một ngắt được đưa ra, các trình xử lý trong danh sách tương ứng được gọi lần lượt cho đến khi tìm thấy một ngắt có thể phục vụ yêu cầu. Cấu trúc này là sự thỏa hiệp giữa chi phí của một bảng ngắt khổng lồ và sự kém hiệu quả của việc điều phối đến một trình xử lý ngắt duy nhất.

Hình 1.5 minh họa thiết kế của vector ngắt cho bộ xử lý Intel. Các sự kiện từ 0 đến 31, không thể che được, được sử dụng để báo hiệu các điều kiện lỗi khác nhau. Các sự kiện từ 32 đến 255, có thể che được, được sử dụng cho các mục đích như ngắt do thiết bị tạo ra.

Cơ chế ngắt cũng thực hiện một hệ thống **các mức ưu tiên ngắt (interrupt priority levels)**. Các mức này cho phép CPU trì hoãn việc xử lý các ngắt có mức ưu tiên thấp mà không che tất cả các ngắt và làm cho ngắt có mức ưu tiên cao có thể thực hiện trước ngắt có mức ưu tiên thấp.

Giá trị vec–tơ	Diễn giải
0	chia lỗi (ví dụ chia cho 0)
1	ngoại lệ gỡ lỗi
2	ngắt rỗng
3	điểm ngắt
4	INTO–phát hiện tràn
5	ngoại lệ phạm vi giới hạn
6	opcode không hợp lệ
7	thiết bị không có sẵn
8	lỗi nặng
9	overrun phân đoạn bộ đồng xử lý (dành riêng)
10	phân đoạn trạng thái nhiệm vụ không hợp lệ
11	phân đoạn không có mặt
12	lỗi ngăn xếp
13	bảo vệ chung
14	Lỗi trang
15	(Intel dành riêng, không sử dụng)
16	lỗi dấu phẩy động
17	kiểm tra căn chỉnh
18	kiểm tra máy móc
19–31	(Intel dành riêng, không sử dụng)
32–255	Các ngắt có thể che được

Hình 1.5 Intel processor event–vector table.

Tóm lại, ngắt được sử dụng trong khắp các hệ điều hành hiện đại để xử lý các sự kiện không đồng bộ (và cho các mục đích khác, chúng ta sẽ thảo luận trong toàn bộ văn bản). Bộ điều khiển thiết bị và lỗi phần cứng gây ra ngắt. Để cho phép thực hiện công việc khẩn cấp nhất trước tiên, các máy tính hiện đại sử dụng một hệ thống ngắt nhiều mức ưu tiên. Bởi vì ngắt được sử dụng rất nhiều cho quá trình xử lý "nhạy cảm về thời gian", cần phải xử lý ngắt hiệu quả để hệ thống hoạt động tốt.

1.2.2 Cấu trúc lưu trữ

CPU chỉ có thể tải các lệnh từ bộ nhớ, vì vậy bất kỳ chương trình nào trước tiên phải được tải vào bộ nhớ để chạy. Máy tính đa năng chạy hầu hết các chương trình của chúng từ bộ nhớ ghi lại, được gọi là **bộ nhớ chính** (còn được gọi là **bộ nhớ truy cập ngẫu nhiên**, hoặc **RAM**). Bộ nhớ chính thường được thực hiện trong một công nghệ bán dẫn được gọi là **bộ nhớ truy cập ngẫu nhiên động** (**DRAM** *vt.* **Dynamic RAM**).

Máy tính cũng sử dụng các dạng bộ nhớ khác. Ví dụ: chương trình đầu tiên chạy trên máy tính bật nguồn là **chương trình khởi động mồi (bootstrap program)**, chương trình này sau đó sẽ tải hệ điều hành. Vì RAM dễ **biến động (volatile)** – mất nội dung lưu trữ khi tắt nguồn hoặc bị mất – chúng ta không thể tin tưởng để nó lưu chương trình khởi động mồi. Thay vào đó, cho mục đích này và một số mục đích khác, máy tính sử dụng bộ nhớ chỉ đọc lập trình được có thể xóa bằng điện (EEPROM vt. electrically erasable programmable read–only memory) và các dạng **phần sụn (firmware)** khác – bộ nhớ không thường xuyên được ghi vào và không biến động. EEPROM có thể được thay đổi nhưng không thể thay đổi thường xuyên. Ngoài ra, nó có tốc độ thấp nên chủ yếu chứa các chương trình tĩnh và dữ liệu không được sử dụng thường xuyên. Ví dụ, iPhone sử dụng EEPROM để lưu trữ số sê–ri và thông tin phần cứng về thiết bị.

ĐỊNH NGHĨA VÀ LƯU Ý VỀ LƯU TRỮ

Đơn vị cơ bản của lưu trữ máy tính là **bit**. Một bit có thể chứa một trong hai giá trị, 0 và 1. Tất cả các bộ lưu trữ khác trong máy tính đều dựa trên tập hợp các bit. Được cung cấp đủ số bit, thật đáng kinh ngạc làm thế nào mà máy tính có thể biểu diễn được nhiều thứ: số, chữ cái, hình ảnh, phim, âm thanh, tài liệu và chương trình, .v.v. . Một **byte** là 8 bit và trên hầu hết các máy tính, đó là đoạn nhỏ tiện lợi nhất của lưu trữ. Ví dụ: hầu hết các máy tính không có lệnh để di chuyển một bit nhưng có một lệnh để di chuyển một byte. Một thuật ngữ ít phổ biến hơn là **từ (word)**, là đơn vị dữ liệu gốc của kiến trúc máy tính nhất định. Một từ được tạo thành từ một hoặc nhiều byte. Ví dụ, một máy tính có thanh ghi 64-bit và địa chỉ bộ nhớ 64-bit thường có các từ 64-bit (8-byte). Một máy tính thực hiện nhiều thao tác với kích thước từ gốc của nó thay vì một byte tại một thời điểm.

Lưu trữ máy tính, cùng với hầu hết thông lượng máy tính, thường được đo lường và xử lý theo byte và tập hợp các byte. Một **kilobyte**, hoặc KB, là 1,024 byte; một **megabyte**, hoặc MB, là $1{,}024^2$ byte; một **gigabyte**, hoặc GB, là $1{,}024^3$ byte; một **terabyte**, hoặc TB, là $1{,}024^4$ byte; và một **petabyte**, hoặc PB, là $1{,}024^5$ byte. Các nhà sản xuất máy tính thường làm tròn những con số này và nói rằng một megabyte là 1 triệu byte và gigabyte là 1 tỷ byte. Các phép đo mạng là một ngoại lệ đối với quy tắc chung này; chúng được cung cấp theo từng bit (vì các mạng di chuyển dữ liệu từng bit một).

Tất cả các dạng bộ nhớ đều cung cấp một mảng các byte. Mỗi byte có địa chỉ riêng của nó. Tương tác đạt được thông qua một chuỗi các lệnh `load` (tải) hoặc `store` (lưu trữ) đến các địa chỉ bộ nhớ cụ thể. Lệnh `load` di chuyển một byte hoặc một từ (word) từ bộ nhớ chính sang một thanh ghi bên trong CPU, trong khi lệnh `store` di chuyển nội dung của một thanh ghi vào bộ nhớ chính. Ngoài lệnh `load` và `store` rõ ràng, CPU tự động tải các lệnh từ bộ nhớ chính để thực thi từ vị trí được lưu trữ trong bộ đếm chương trình.

Một chu trình thực thi lệnh điển hình, khi được thực thi trên hệ thống có **kiến trúc Von**

Neumann, trước tiên tìm nạp một lệnh từ bộ nhớ và lưu lệnh đó trong **thanh ghi lệnh (instruction register)**. Sau đó, lệnh được giải mã và có thể tiếp diễn bằng việc nạp giá trị các toán hạng từ bộ nhớ và được lưu trữ trong một số thanh ghi bên trong. Sau khi lệnh trên các toán hạng đã được thực thi, kết quả có thể được lưu lại trong bộ nhớ. Lưu ý rằng đơn vị bộ nhớ chỉ thấy một dòng địa chỉ bộ nhớ. Nó không biết chúng được tạo ra như thế nào (bởi bộ đếm lệnh, lập chỉ mục, chuyển hướng, địa chỉ nguyên văn hoặc một số cách khác) hoặc chúng dùng để làm gì (lệnh hay dữ liệu). Do đó, chúng ta có thể bỏ qua cách một chương trình tạo ra địa chỉ bộ nhớ. Chúng ta chỉ quan tâm đến chuỗi địa chỉ bộ nhớ được tạo ra bởi chương trình đang chạy.

Lý tưởng nhất là chúng ta muốn các chương trình và dữ liệu thường trú trong bộ nhớ chính. Sự sắp xếp này thường không thể thực hiện được trên hầu hết các hệ thống vì hai lý do:

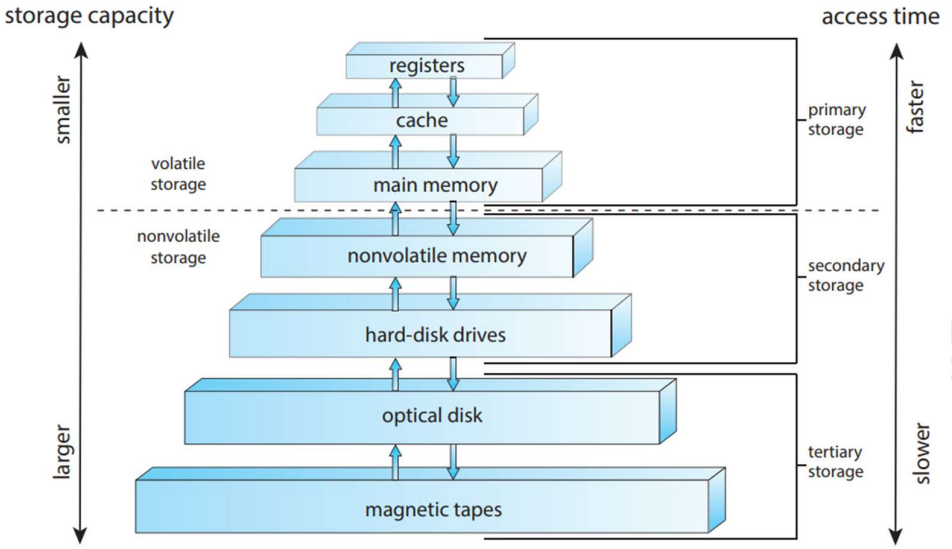
1. Bộ nhớ chính thường quá nhỏ để lưu trữ vĩnh viễn tất cả các chương trình và dữ liệu cần thiết.
2. Bộ nhớ chính, như đã đề cập, có tính biến động – nó mất nội dung khi tắt nguồn hoặc sự cố mất điện.

Vì vậy, hầu hết các hệ thống máy tính cung cấp **bộ nhớ thứ cấp (secondary storage)** (đôi khi còn gọi là lưu trữ mức thứ hai) như một phần mở rộng của bộ nhớ chính. Yêu cầu chính đối với bộ nhớ thứ cấp là nó có thể lưu trữ số lượng lớn dữ liệu vĩnh viễn.

Các thiết bị lưu trữ thứ cấp phổ biến nhất là **ổ đĩa cứng (HDD *vt.* Hard disk drive)** và thiết bị **bộ nhớ không biến động (NVM *vt.* Nonvolatile memory)**, cung cấp khả năng lưu trữ cho cả chương trình và dữ liệu. Hầu hết các chương trình (hệ thống và ứng dụng) được lưu trữ trong bộ nhớ thứ cấp cho đến khi chúng được tải vào bộ nhớ. Nhiều chương trình sau đó sử dụng bộ nhớ thứ cấp vừa là nguồn vừa là đích xử lý của chúng. Bộ nhớ thứ cấp cũng chậm hơn nhiều so với bộ nhớ chính. Do đó, việc quản lý thích hợp bộ nhớ thứ cấp có tầm quan trọng trung tâm đối với hệ thống máy tính, như chúng ta sẽ thảo luận trong Chương 11.

Tuy nhiên, theo một nghĩa lớn hơn, cấu trúc lưu trữ mà chúng ta đã mô tả – bao gồm các thanh ghi, bộ nhớ chính và bộ nhớ thứ cấp – chỉ là một trong nhiều thiết kế hệ thống lưu trữ khả thi. Các thành phần có thể có khác bao gồm bộ nhớ đệm, CD–ROM hoặc blu–ray, băng từ, .v.v. . Những thiết bị đủ chậm và đủ lớn để chúng chỉ được sử dụng cho các mục đích đặc biệt – để lưu trữ các bản sao dự phòng của tài liệu được lưu trữ trên các thiết bị khác – được gọi là **lưu trữ mức thứ ba (tertiary storage)**. Mỗi hệ thống lưu trữ cung cấp các chức năng cơ bản để lưu trữ một mức dữ liệu và giữ mức dữ liệu đó cho đến khi nó được truy xuất sau đó. Sự khác biệt chính giữa các hệ thống lưu trữ khác nhau nằm ở tốc độ, kích thước và tính biến động.

Nhiều hệ thống lưu trữ có thể được tổ chức theo thứ bậc (Hình 1.6) tùy theo dung lượng lưu trữ và thời gian truy cập. Theo nguyên tắc chung, có sự cân bằng giữa kích thước và tốc độ, với bộ nhớ nhỏ hơn và nhanh hơn gần CPU hơn. Như thể hiện trong hình, ngoài việc khác nhau về tốc độ và dung lượng, các hệ thống lưu trữ khác nhau có thể biến động hoặc không. Bộ lưu trữ biến động, như đã đề cập trước đó, mất nội dung của nó khi nguồn điện cho thiết bị bị ngắt, vì vậy dữ liệu phải được ghi vào bộ lưu trữ không biến động để bảo quản an toàn.



Hình 1.6 Thứ bậc thiết bị lưu trữ.

Bốn cấp độ bộ nhớ cao nhất trong hình được xây dựng bằng **bộ nhớ bán dẫn (semi–conductor memory)**, bao gồm các mạch điện tử dựa trên chất bán dẫn. Các thiết bị NVM, ở cấp độ thứ tư, có một số biến thể nhưng nói chung là nhanh hơn so với đĩa cứng. Hình thức phổ biến nhất của thiết bị NVM là bộ nhớ flash, phổ biến trong các thiết bị di động như điện thoại thông minh và máy tính bảng. Bộ nhớ flash ngày càng được sử dụng để lưu trữ lâu dài trên máy tính xách tay, máy tính để bàn và máy chủ.

Vì lưu trữ đóng một vai trò quan trọng trong cấu trúc hệ điều hành, chúng tôi sẽ đề cập đến nó thường xuyên trong sách. Nói chung, chúng ta sẽ sử dụng thuật ngữ sau:

- Bộ nhớ biến động sẽ được gọi đơn giản là **bộ nhớ (memory)**. Nếu chúng ta cần nhấn mạnh một loại thiết bị lưu trữ cụ thể (ví dụ, một thanh ghi), chúng ta sẽ làm như vậy một cách rõ ràng.
- Bộ lưu trữ không biến động giữ lại nội dung của nó khi mất điện. Nó sẽ được gọi là **NVS**. Phần lớn thời gian chúng ta dành cho NVS sẽ dành cho bộ nhớ thứ cấp. Loại lưu trữ này có thể được phân thành hai loại riêng biệt:
  - **Cơ khí (Mechanical)**. Một vài ví dụ về các hệ thống lưu trữ như vậy là ổ cứng, đĩa quang, lưu trữ ảnh ba chiều và băng từ. Nếu cần nhấn mạnh một loại thiết bị lưu trữ cơ học cụ thể (ví dụ, băng từ), chúng tôi sẽ làm như vậy một cách rõ ràng.
  - **Điện (Electrical)**. Một vài ví dụ về các hệ thống lưu trữ như vậy là bộ nhớ flash, FRAM, NRAM và SSD. Lưu trữ điện sẽ được gọi là **NVM**. Nếu chúng ta cần nhấn mạnh một loại thiết bị lưu trữ điện cụ thể (ví dụ: SSD), chúng ta sẽ làm như vậy một cách rõ ràng.

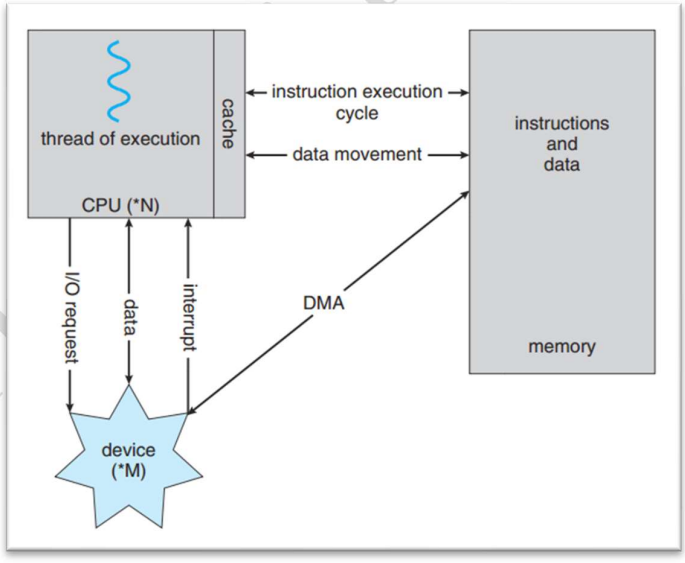
Lưu trữ cơ học thường lớn hơn và ít tốn kém hơn trên mỗi byte so với lưu trữ điện. Ngược lại, lưu trữ điện thường tốn kém, nhỏ hơn và nhanh hơn so với lưu trữ cơ học.

Thiết kế của một hệ thống lưu trữ hoàn chỉnh phải cân bằng tất cả các yếu tố vừa thảo luận: nó phải chỉ sử dụng bộ nhớ đắt tiền nhất cần thiết trong khi cung cấp càng nhiều lưu trữ rẻ tiền, không dễ bay hơi càng tốt. Bộ nhớ đệm có thể được cài đặt để cải thiện hiệu suất khi có sự chênh lệch lớn về thời gian truy cập hoặc tốc độ truyền giữa hai thành phần.

1.2.3 Cấu trúc Nhập/Xuất

Một phần lớn mã hệ điều hành được dành riêng để quản lý Nhập/Xuất, cả vì tầm quan trọng của nó đối với độ tin cậy và hiệu suất của hệ thống cũng như vì tính chất khác nhau của các thiết bị.

Nhớ lại từ đầu phần này rằng một hệ thống máy tính đa năng bao gồm nhiều thiết bị, tất cả đều trao đổi dữ liệu thông qua một bus chung. Hình thức Nhập/Xuất điều khiển gián đoạn được mô tả trong Phần 1.2.1 phù hợp để di chuyển một lượng nhỏ dữ liệu nhưng có thể tạo ra chi phí cao khi được sử dụng để di chuyển dữ liệu hàng loạt như các thiết bị NVS Nhập/Xuất. Để giải quyết vấn đề này, **truy cập bộ nhớ trực tiếp (DMA *vt.* direct memory access)** được sử dụng. Sau khi thiết lập bộ đệm, con trỏ và bộ đếm cho thiết bị Nhập/Xuất, bộ điều khiển thiết bị truyền toàn bộ khối dữ liệu trực tiếp đến hoặc từ thiết bị và bộ nhớ chính mà không có sự can thiệp của CPU. Chỉ một ngắt được tạo cho mỗi khối, để cho trình điều khiển thiết bị biết rằng hoạt động đã hoàn thành, thay vì một ngắt trên mỗi byte được tạo cho các thiết bị tốc độ thấp. Trong khi bộ điều khiển thiết bị đang thực hiện các hoạt động này, CPU vẫn khả dụng để hoàn thành công việc khác.



Hình 1.7 Cách thức máy tính hiện đại hoạt động.

Một số hệ thống cao cấp sử dụng chuyển mạch thay vì kiến trúc bus. Trên các hệ thống này, nhiều thành phần có thể trao đổi dữ liệu với các thành phần khác một cách đồng thời, thay vì cạnh tranh cho các chu kỳ trên một liên kết bus được chia sẻ. Trong trường hợp này, DMA thậm chí còn hiệu quả hơn. Hình 1.7 cho thấy tác động qua lại của tất cả các thành phần của hệ thống máy tính.

1.3 Kiến trúc hệ thống máy tính

Trong phần 1.2, chúng tôi đã giới thiệu cấu trúc chung của một hệ thống máy tính điển hình. Một hệ thống máy tính có thể được tổ chức theo một số cách khác nhau, chúng ta có thể phân loại một cách đại khái theo số lượng bộ xử lý đa năng được sử dụng.

1.3.1 Hệ thống một bộ xử lý

Nhiều năm trước, hầu hết các hệ thống máy tính đều sử dụng một bộ xử lý duy nhất chứa một CPU với một nhân xử lý duy nhất. Phần **nhân (core)** là thành phần thực thi các lệnh và các thanh ghi dùng để lưu trữ dữ liệu cục bộ. Một CPU chính với nhân của nó có khả năng thực hiện một tập lệnh tổng quát, bao gồm các lệnh từ các tiến trình. Các hệ thống này cũng có các bộ xử lý có mục đích đặc biệt khác. Chúng có thể ở dạng bộ xử lý dành riêng cho thiết bị, chẳng hạn như đĩa, bàn phím và bộ điều khiển đồ họa.

Tất cả các bộ xử lý có mục đích đặc biệt này đều chạy một tập lệnh giới hạn và không thực thi các tiến trình. Đôi khi, chúng được quản lý bởi hệ điều hành, trong đó hệ điều hành gửi cho chúng thông tin về tác vụ tiếp theo và giám sát trạng thái của chúng. Ví dụ, một bộ vi xử lý điều khiển đĩa nhận một chuỗi các yêu cầu từ nhân CPU chính và thực hiện thuật toán lập lịch và xếp hàng đĩa của riêng nó. Sự sắp xếp này giải phóng CPU chính của chi phí lập lịch đĩa. PC có chứa một bộ vi xử lý trong bàn phím để chuyển các lần gõ phím thành mã để gửi đến CPU. Trong các hệ thống hoặc trường hợp khác, bộ xử lý có mục đích đặc biệt là các thành phần cấp thấp được tích hợp trong phần cứng. Hệ điều hành không thể giao tiếp với các bộ xử lý này; họ làm công việc của họ một cách tự chủ. Việc sử dụng các bộ vi xử lý có mục đích đặc biệt là phổ biến và không biến hệ thống đơn xử lý thành đa xử lý. Nếu chỉ có một CPU đa năng với một nhân xử lý duy nhất, thì hệ thống là hệ thống một bộ xử lý. Tuy nhiên, theo định nghĩa này, rất ít hệ thống máy tính đương đại là hệ thống một bộ xử lý.

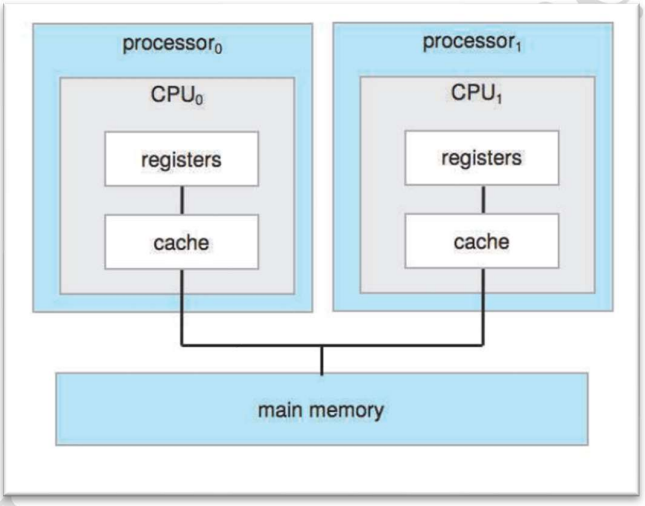
1.3.2 Hệ thống đa xử lý

Trên các máy tính hiện đại, từ thiết bị di động đến máy chủ, các **hệ thống đa xử lý (multiprocessor system)** hiện đang chiếm ưu thế trong bối cảnh điện toán. Theo truyền thống, các hệ thống như vậy có hai (hoặc nhiều) bộ xử lý, mỗi bộ có một CPU đơn nhân. Các bộ xử lý chia sẻ bus máy tính và đôi khi là xung nhịp, bộ nhớ và các thiết bị ngoại vi. Ưu điểm chính của hệ thống đa xử lý là tăng thông lượng. Tức là, bằng cách tăng số lượng bộ xử lý, chúng ta hy vọng sẽ hoàn thành nhiều công việc hơn trong thời gian ngắn hơn. Tuy nhiên, tỷ lệ tăng tốc độ với N bộ xử lý không phải là N; tỷ lệ đó nhỏ hơn N. Khi nhiều bộ xử lý hợp tác trong một nhiệm vụ, một lượng chi phí nhất định sẽ phát sinh để giữ cho tất cả các bộ phận hoạt động chính xác. Chi phí này, cộng với sự tranh giành tài nguyên dùng chung, làm giảm lợi nhuận mong đợi từ các bộ xử lý bổ sung.

Các hệ thống đa xử lý phổ biến nhất sử dụng **đa xử lý đối xứng (SMP *vt.* symmetric multiprocessing)**, trong đó mỗi bộ xử lý CPU ngang hàng thực hiện tất cả các tác vụ, bao gồm các chức năng của hệ điều hành và các tiến trình của người dùng. Hình 1.8 minh họa một kiến trúc SMP điển hình với hai bộ xử lý, mỗi bộ có một CPU riêng. Lưu ý rằng mỗi bộ xử lý CPU có một bộ thanh ghi riêng, cũng như một bộ nhớ đệm riêng – hoặc cục bộ –. Tuy nhiên, tất cả các bộ xử lý đều chia sẻ bộ nhớ vật lý qua bus hệ thống.

Lợi ích của mô hình này là nhiều tiến trình có thể chạy đồng thời – N tiến trình có thể chạy nếu có N CPU – mà không làm giảm hiệu suất đáng kể. Tuy nhiên, vì các CPU riêng biệt nên một CPU có thể không hoạt động trong khi một CPU khác bị quá tải, dẫn đến hoạt động kém hiệu quả. Những sự kém hiệu quả này có thể tránh được nếu các bộ xử lý chia sẻ các cấu trúc dữ liệu nhất định. Hệ thống đa xử lý dạng này sẽ cho phép các tiến trình và tài nguyên – chẳng hạn như bộ nhớ – được chia sẻ động giữa các bộ xử lý khác nhau và có thể giảm bớt sự chênh lệch khối lượng công việc giữa các bộ xử lý. Hệ thống như vậy phải được lập trình cẩn thận, như chúng ta sẽ thấy trong Chương 5 và Chương 6.

Định nghĩa về ***đa xử lý*** đã phát triển theo thời gian và giờ đây bao gồm các hệ thống **đa nhân (multicore)**, trong đó nhiều nhân điện toán nằm trên một chip duy nhất. Hệ thống đa nhân có thể hiệu quả hơn nhiều chip với nhân đơn vì giao tiếp trên chip nhanh hơn giao tiếp giữa các chip. Ngoài ra, một chip với nhiều nhân sử dụng ít năng lượng hơn đáng kể so với nhiều chip nhân đơn, một vấn đề quan trọng đối với các thiết bị di động cũng như máy tính xách tay.

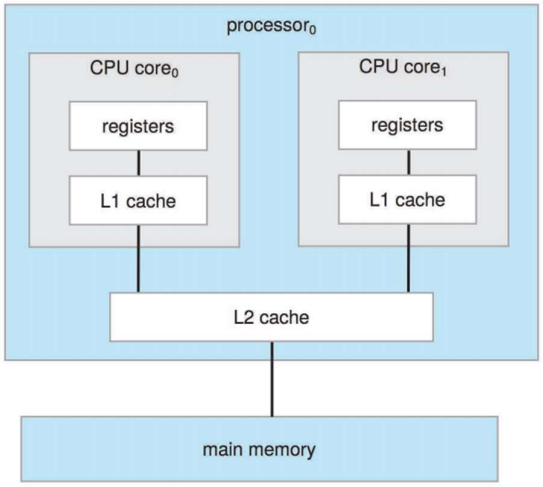


Hình 1.8 Kiến trúc đa xử lý đối xứng.

Trong hình 1.9, chúng tôi cho thấy một thiết kế lõi kép với hai nhân trên cùng một chip xử lý. Trong thiết kế này, mỗi nhân có bộ thanh ghi riêng, cũng như bộ đệm cục bộ của riêng nó, thường được gọi là bộ đệm cấp 1, hoặc L1 (Level 1). Cũng lưu ý rằng bộ nhớ cache cấp 2 (L2) là cục bộ của chip nhưng được chia sẻ bởi hai nhân xử lý. Hầu hết các kiến trúc áp dụng cách tiếp cận này, kết hợp bộ nhớ đệm cục bộ và bộ nhớ đệm dùng chung, trong đó bộ nhớ đệm cục bộ, bộ nhớ đệm cấp thấp hơn thường nhỏ hơn và nhanh hơn bộ nhớ đệm dùng chung cấp cao hơn. Bên cạnh những cân nhắc về kiến trúc, chẳng hạn như bộ nhớ đệm, bộ nhớ và tranh chấp bus, một bộ xử lý đa nhân với N nhân xuất hiện trong hệ điều hành dưới dạng N CPU tiêu chuẩn. Đặc điểm này gây áp lực lên các nhà thiết kế hệ điều hành – và các nhà lập trình ứng dụng – để sử dụng hiệu quả các nhân xử lý này, một vấn đề mà chúng tôi theo đuổi trong Chương 4. Hầu như tất cả các hệ điều hành hiện đại – bao gồm Windows, macOS và Linux, cũng như Android và Hệ thống di động iOS – đều hỗ trợ hệ thống đa nhân đối xứng.

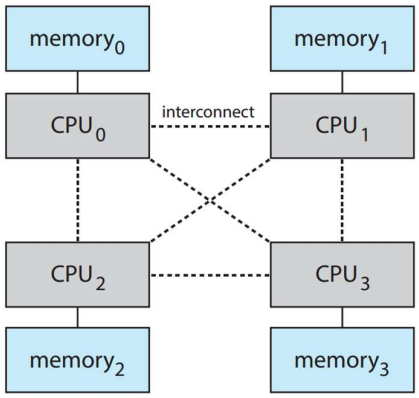
Thêm CPU bổ sung vào hệ thống đa xử lý sẽ tăng sức mạnh tính toán; tuy nhiên, như đã đề xuất trước đó, khái niệm này không mở rộng tốt lắm, và một khi chúng ta thêm quá nhiều

CPU, sự tranh giành bus hệ thống sẽ trở thành một nút thắt cổ chai và hiệu suất bắt đầu suy giảm. Một cách tiếp cận khác là cung cấp cho mỗi CPU (hoặc nhóm CPU) bộ nhớ cục bộ của riêng nó được truy cập thông qua một bus cục bộ nhỏ, nhanh. Các CPU được kết nối bằng một **kết nối hệ thống dùng chung (shared system interconnect)**, do đó tất cả các CPU đều chia sẻ một không gian địa chỉ vật lý. Cách tiếp cận này – được gọi là **truy cập bộ nhớ không đồng nhất**, hoặc **NUMA (Non–uniform memory access)** – được minh họa trong Hình 1.10. Ưu điểm là, khi một CPU truy cập bộ nhớ cục bộ của nó, nó không chỉ nhanh mà còn không có tranh chấp về kết nối hệ thống. Do đó, các hệ thống NUMA có thể mở rộng quy mô hiệu quả hơn khi nhiều bộ xử lý được thêm vào.



Hình 1.9 Một thiết kế lõi kép với hai lõi trên cùng một con chip.

Một nhược điểm tiềm ẩn với hệ thống NUMA là độ trễ tăng lên khi CPU phải truy cập bộ nhớ từ xa qua kết nối hệ thống, tạo ra một hình phạt hiệu suất có thể xảy ra. Nói cách khác, ví dụ, CPU_0 không thể truy cập bộ nhớ cục bộ của CPU_3 nhanh như nó có thể truy cập bộ nhớ cục bộ của chính nó, làm chậm hiệu suất. Hệ điều hành có thể giảm thiểu hình phạt NUMA này thông qua việc lập lịch CPU và quản lý bộ nhớ cẩn thận, như đã thảo luận trong Phần 5.5.2 và Phần 10.5.4. Bởi vì các hệ thống NUMA có thể mở rộng quy mô để chứa một số lượng lớn bộ xử lý, chúng ngày càng trở nên phổ biến trên các máy chủ cũng như các hệ thống tính toán hiệu suất cao.



Hình 1.10 Kiến trúc vi xử lý NUMA.

ĐỊNH NGHĨA CÁC THÀNH PHẦN CỦA HỆ THỐNG MÁY TÍNH

- **CPU** – Phần cứng thực thi các lệnh.
- **Bộ xử lý (Processor)** – Một chip vật lý có chứa một hoặc nhiều CPU.
- **Nhân (Core)** – Đơn vị tính toán cơ bản của CPU.
- **Đa nhân (Multicore)** – Gồm nhiều nhân tính toán trên cùng một CPU.
- Multiprocessor-Bao gồm nhiều bộ vi xử lý.

Mặc dù hầu như tất cả các hệ thống hiện nay đều là đa nhân, chúng tôi sử dụng thuật ngữ chung CPU khi đề cập đến một đơn vị tính toán duy nhất của hệ thống máy tính và nhân cũng như đa nhân khi đề cập cụ thể đến một hoặc nhiều nhân trên CPU.

Cuối cùng, các **máy chủ phiến (blade server)** là hệ thống trong đó nhiều bo mạch xử lý, bo mạch Nhập/Xuất và bo mạch mạng được đặt trong cùng một khung. Sự khác biệt giữa các hệ thống này và các hệ thống đa xử lý truyền thống là mỗi bo mạch bộ xử lý phiến khởi động độc lập và chạy hệ điều hành riêng. Một số bo mạch máy chủ phiến cũng là bộ xử lý đa xử lý, điều này làm mờ ranh giới giữa các loại máy tính. Về bản chất, các máy chủ này bao gồm nhiều hệ thống đa xử lý độc lập.

1.3.3 Hệ thống phân cụm

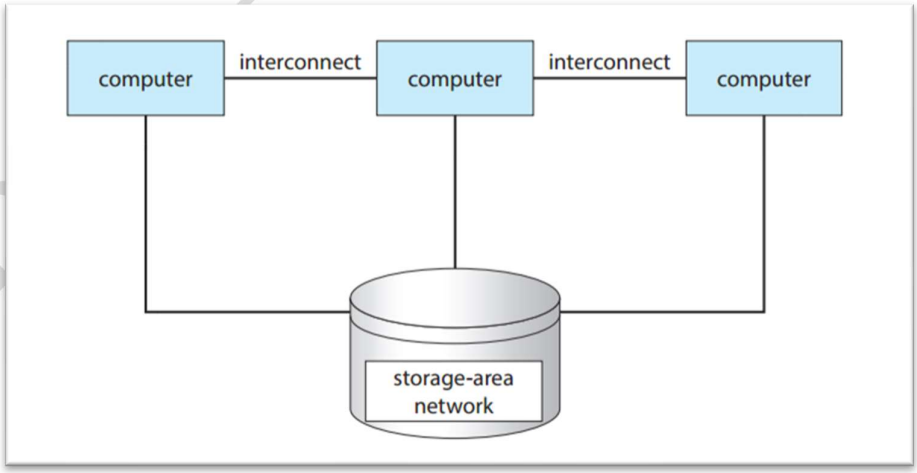
Một loại hệ thống đa xử lý khác là **hệ thống phân cụm (clustered system)**, tập hợp nhiều CPU lại với nhau. Hệ thống phân cụm khác với hệ thống đa xử lý được mô tả trong Phần 1.3.2 ở chỗ chúng bao gồm hai hoặc nhiều hệ thống riêng lẻ – hoặc các nút tính toán – được kết hợp với nhau; mỗi nút thường là một hệ thống đa nhân. Các hệ thống như vậy được coi là **liên kết lỏng lẻo (loosely coupled)**. Chúng ta cần lưu ý rằng định nghĩa của cụm không phải là cụ thể; nhiều gói mã nguồn mở và thương mại vật lộn để xác định hệ thống ***phân cụm*** là gì và tại sao một dạng này tốt hơn dạng khác. Định nghĩa được chấp nhận chung là các máy tính phân cụm chia sẻ dung lượng lưu trữ và được liên kết chặt chẽ qua mạng LAN cục bộ (như được mô tả trong Chương 19) hoặc kết nối liên thông nhanh hơn, chẳng hạn như InfiniBand.

Phân cụm thường được sử dụng để cung cấp dịch vụ có **tính khả dụng cao (high–availability service)** – tức là dịch vụ sẽ tiếp tục ngay cả khi một hoặc nhiều hệ thống trong cụm bị lỗi. Nói chung, chúng ta sẽ có được tính khả dụng cao bằng cách thêm một mức độ dự phòng trong hệ thống. Một lớp phần mềm cụm chạy trên các nút cụm. Mỗi nút có thể giám sát một hoặc nhiều nút khác (qua mạng). Nếu máy được giám sát bị lỗi, máy theo dõi có thể chiếm quyền sở hữu bộ nhớ của nó và khởi động lại các ứng dụng đang chạy trên máy bị lỗi. Người dùng và khách hàng của các ứng dụng chỉ thấy một thời gian ngắn dịch vụ bị gián đoạn.

Tính khả dụng cao giúp tăng độ tin cậy, điều này rất quan trọng trong nhiều ứng dụng. Khả năng tiếp tục cung cấp dịch vụ tỷ lệ thuận với mức độ phần cứng còn tồn tại được gọi là **suy giảm chất lượng (graceful degradation)**[1]. Một số hệ thống vượt quá mức xuống cấp đáng kể và được gọi là **khả năng chịu lỗi (fault tolerant)**, bởi vì chúng có thể bị lỗi ở bất kỳ thành phần đơn lẻ nào và vẫn tiếp tục hoạt động. Khả năng chịu lỗi đòi hỏi một cơ chế cho phép phát hiện, chẩn đoán và sửa lỗi nếu có thể.

Phân cụm có thể được cấu trúc không đối xứng hoặc đối xứng. Trong **phân cụm không đối xứng (asymmetric clustering)**, một máy ở **chế độ chờ nóng (hot–standby mode)** trong khi máy kia đang chạy các ứng dụng. Máy chủ ở chế độ chờ nóng không làm gì khác ngoài việc giám sát máy chủ đang hoạt động. Nếu máy chủ đó bị lỗi, máy chủ dự phòng nóng sẽ trở thành máy chủ hoạt động. Trong **phân cụm đối xứng (symmetric clustering)**, hai hoặc nhiều máy chủ đang chạy các ứng dụng và đang giám sát lẫn nhau. Cấu trúc này rõ ràng là hiệu quả hơn, vì nó sử dụng tất cả các phần cứng có sẵn. Tuy nhiên, nó yêu cầu phải có nhiều hơn một ứng dụng để chạy.

Vì một cụm bao gồm một số hệ thống máy tính được kết nối qua mạng, các cụm cũng có thể được sử dụng để cung cấp môi trường **tính toán hiệu suất cao (high–performance computing)**. Các hệ thống như vậy có thể cung cấp sức mạnh tính toán lớn hơn đáng kể so với các hệ thống xử lý đơn hoặc thậm chí là SMP vì chúng có thể chạy một ứng dụng đồng thời trên tất cả các máy tính trong cụm. Tuy nhiên, ứng dụng phải được viết riêng để tận dụng lợi thế của cụm. Điều này liên quan đến một kỹ thuật được gọi là **song song hóa (parallelization)**, chia một chương trình thành các thành phần riêng biệt chạy song song trên các nhân riêng lẻ trong một máy tính hoặc các máy tính trong một cụm. Thông thường, các ứng dụng này được thiết kế để một khi mỗi nút tính toán trong cụm đã giải quyết được phần của vấn đề, kết quả từ tất cả các nút sẽ được kết hợp thành một giải pháp cuối cùng.



Hình 1.11 Cấu trúc chung của một hệ thống phân cụm.

Các dạng cụm khác bao gồm cụm song song và cụm qua mạng diện rộng (WAN) (như được mô tả trong Chương 19). Các cụm song song cho phép nhiều máy chủ truy cập vào cùng

[1] Graceful degradation: là khả năng của máy tính, máy móc, hệ thống điện tử hoặc mạng để duy trì chức năng hạn chế ngay cả khi một phần lớn của nó đã bị phá hủy hoặc không hoạt động. ... Khi bị xuống cấp, hiệu quả hoạt động hoặc tốc độ giảm dần do ngày càng có nhiều bộ phận hỏng hóc.

một dữ liệu trên bộ nhớ dùng chung. Bởi vì hầu hết các hệ điều hành thiếu hỗ trợ truy cập dữ liệu đồng thời bởi nhiều máy chủ, các cụm song song thường yêu cầu sử dụng các phiên bản phần mềm đặc biệt và các bản phát hành ứng dụng đặc biệt. Ví dụ, Oracle Real Application Cluster là một phiên bản cơ sở dữ liệu của Oracle đã được thiết kế để chạy trên một cụm song song. Mỗi máy chạy Oracle và một lớp phần mềm theo dõi quyền truy cập vào đĩa dùng chung. Mỗi máy có toàn quyền truy cập vào tất cả dữ liệu trong cơ sở dữ liệu. Để cung cấp quyền truy cập được chia sẻ này, hệ thống cũng phải cung cấp kiểm soát truy cập và khóa để đảm bảo rằng không có hoạt động xung đột nào xảy ra. Chức năng này, thường được gọi là **trình quản lý khóa phân tán (DLM vt. distributed lock manager)**, được bao gồm trong một số công nghệ cụm.

HADOOP

Hadoop là một khung phần mềm mã nguồn mở được sử dụng để xử lý phân tán các tập dữ liệu lớn (được gọi là **dữ liệu lớn (big data)**) trong một hệ thống phân cụm chứa các thành phần phần cứng đơn giản, chi phí thấp. Hadoop được thiết kế để mở rộng quy mô từ một hệ thống đơn lẻ thành một cụm chứa hàng nghìn nút máy tính. Các tác vụ được giao cho một nút trong cụm và Hadoop sắp xếp giao tiếp giữa các nút để quản lý các phép tính song song để xử lý và kết hợp các kết quả. Hadoop cũng phát hiện và quản lý các lỗi trong các nút, cung cấp dịch vụ tính toán phân tán hiệu quả và có độ tin cậy cao.

Hadoop được tổ chức xung quanh ba thành phần sau:

1. Hệ thống tệp phân tán quản lý dữ liệu và tệp qua các nút máy tính phân tán.
2. Khung YARN (“Yet Another Resource Negotiator”), quản lý các tài nguyên trong cụm cũng như lập lịch các tác vụ trên các nút trong cụm.
3. Hệ thống MapReduce, cho phép xử lý song song dữ liệu giữa các nút trong cụm.

Hadoop được thiết kế để chạy trên các hệ thống Linux và các ứng dụng Hadoop có thể được viết bằng một số ngôn ngữ lập trình, bao gồm các ngôn ngữ lập trình như PHP, Perl và Python. Java là một lựa chọn phổ biến để phát triển các ứng dụng Hadoop, vì Hadoop có một số thư viện Java hỗ trợ MapReduce. Bạn có thể tìm thêm thông tin về MapReduce và Hadoop tại https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html và https://hadoop.apache.org .

Công nghệ cụm đang thay đổi nhanh chóng. Một số sản phẩm cụm hỗ trợ hàng ngàn hệ thống trong một cụm, cũng như các nút được phân cụm cách xa nhau nhiều kilo–mét. Nhiều cải tiến trong số này có thể thực hiện được nhờ **mạng vùng lưu trữ (SAN *vt.* storage–area network)**, như được mô tả trong Phần 11.7.4, cho phép nhiều hệ thống gắn vào một nhóm lưu

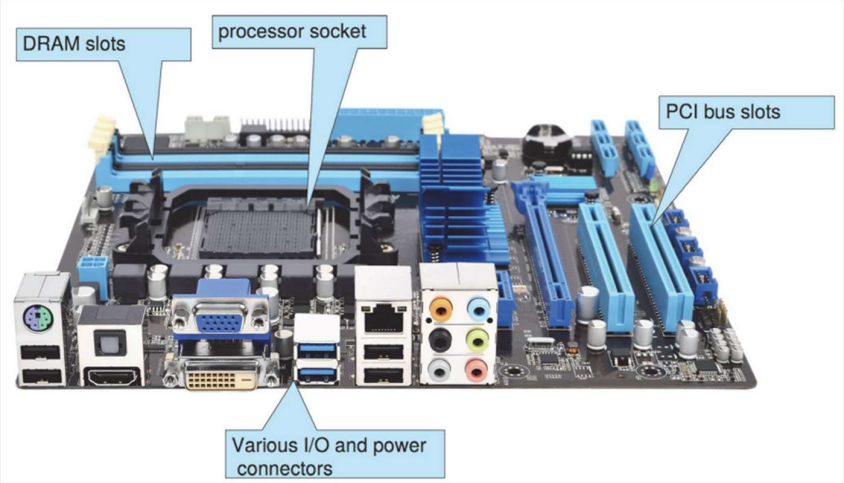
trữ. Nếu các ứng dụng và dữ liệu của chúng được lưu trữ trên SAN, thì phần mềm cụm có thể chỉ định ứng dụng chạy trên bất kỳ máy chủ nào được gắn vào SAN. Nếu máy chủ bị lỗi, thì bất kỳ máy chủ nào khác có thể tiếp quản. Trong một cụm cơ sở dữ liệu, hàng chục máy chủ có thể chia sẻ cùng một cơ sở dữ liệu, làm tăng đáng kể hiệu suất và độ tin cậy. Hình 1.11 mô tả cấu trúc chung của một hệ thống phân cụm.

1.4 Các thao tác của Hệ điều hành

Bây giờ chúng ta đã thảo luận về thông tin cơ bản về tổ chức và kiến trúc hệ thống máy tính, chúng ta đã sẵn sàng để nói về hệ điều hành. Hệ điều hành cung cấp môi trường trong đó các chương trình được thực thi. Về mặt nội bộ, các hệ điều hành khác nhau rất nhiều, vì chúng được tổ chức theo nhiều dòng khác nhau. Tuy nhiên, có nhiều điểm chung mà chúng tôi xem xét trong phần này.

BO MẠCH CHỦ MÁY TÍNH

Hãy xem xét bo mạch chủ PC để bàn có đế cắm bộ xử lý được hiển thị bên dưới:



Bo mạch này là một máy tính hoạt động đầy đủ, khi các khe cắm của nó được gắn linh kiện vào. Nó bao gồm một đế cắm bộ xử lý chứa CPU, khe cắm DRAM, khe cắm bus PCIe và các đầu nối thiết bị Nhập/Xuất khác nhau. Ngay cả CPU đa năng giá rẻ nhất cũng chứa nhiều lõi. Một số bo mạch chủ chứa nhiều ổ cắm bộ xử lý. Các máy tính tiên tiến hơn cho phép nhiều hơn một bo mạch hệ thống, tạo ra các hệ thống NUMA.

Để một máy tính bắt đầu chạy – ví dụ, khi nó được khởi động hoặc khởi động lại – nó cần phải có một chương trình ban đầu để chạy. Như đã lưu ý trước đó, chương trình ban đầu này, hoặc chương trình khởi động mồi, có xu hướng đơn giản. Thông thường, nó được lưu trữ trong phần cứng máy tính trong phần sụn. Nó khởi tạo tất cả các khía cạnh của hệ thống, từ thanh ghi CPU đến bộ điều khiển thiết bị đến nội dung bộ nhớ. Chương trình khởi động mồi

phải biết cách tải hệ điều hành và cách bắt đầu thực thi hệ thống đó. Để thực hiện mục tiêu này, chương trình khởi động mồi phải định vị hạt nhân của hệ điều hành và tải nó vào bộ nhớ.

Khi nhân được tải và thực thi, nó có thể bắt đầu cung cấp các dịch vụ cho hệ thống và người dùng của nó. Một số dịch vụ được cung cấp bên ngoài hạt nhân bởi các chương trình hệ thống được tải vào bộ nhớ tại thời điểm khởi động để trở thành trình nền hệ thống, chạy trong toàn bộ thời gian hạt nhân đang chạy. Trên Linux, chương trình hệ thống đầu tiên là "`systemd`" và nó khởi động nhiều **daemon**[2] khác. Khi giai đoạn này hoàn tất, hệ thống được khởi động hoàn toàn và hệ thống sẽ đợi một số sự kiện xảy ra.

Nếu không có tiến trình nào để thực thi, không có thiết bị Nhập/Xuất nào để phục vụ và không có người dùng nào để phản hồi, hệ điều hành sẽ ngồi yên lặng, chờ đợi điều gì đó xảy ra. Các sự kiện hầu như luôn được báo hiệu bằng sự xuất hiện của một ngắt. Trong Phần 1.2.1, chúng ta đã mô tả các ngắt phần cứng. Một dạng khác của ngắt là một **bẫy (trap)** (hoặc một **ngoại lệ (exception)**), là một ngắt do phần mềm tạo ra do lỗi (ví dụ: chia cho 0 hoặc truy cập bộ nhớ không hợp lệ) hoặc bởi một yêu cầu cụ thể từ một chương trình người dùng mà một hoạt động – dịch vụ hệ thống được thực hiện bằng cách thực hiện một thao tác đặc biệt gọi là lệnh gọi hệ thống.

1.4.1 Đa chương trình và đa tác vụ

Một trong những khía cạnh quan trọng nhất của hệ điều hành là khả năng chạy nhiều chương trình, vì một chương trình đơn lẻ nói chung không thể giữ cho CPU hoặc các thiết bị Nhập/Xuất luôn bận rộn. Hơn nữa, người dùng thường muốn chạy nhiều chương trình cùng một lúc. **Đa chương trình (Multiprogramming)** làm tăng khả năng sử dụng CPU, cũng như giữ cho người dùng hài lòng, bằng cách tổ chức các chương trình sao cho CPU luôn có một chương trình để thực thi. Trong một hệ thống đa chương trình, một chương trình đang được thực thi được gọi là một **tiến trình (process)**.

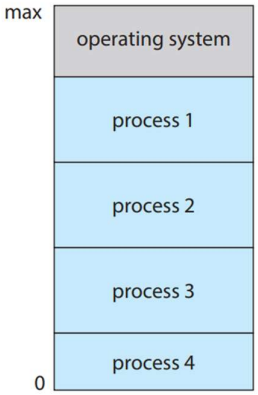
Ý tưởng như sau: Hệ điều hành giữ một số tiến trình trong bộ nhớ đồng thời (Hình 1.12). Hệ điều hành chọn và bắt đầu thực hiện một trong các quá trình này. Cuối cùng, tiến trình có thể phải đợi một số tác vụ, chẳng hạn như hoạt động Nhập/Xuất, để hoàn thành. Trong một hệ thống không đa chương trình, CPU sẽ không hoạt động. Trong một hệ thống đa chương trình, hệ điều hành chỉ cần chuyển sang và thực thi một tiến trình khác. Khi tiến trình đó cần chờ, CPU sẽ chuyển sang một tiến trình khác, v.v. Cuối cùng, tiến trình đầu tiên kết thúc và chờ CPU trở lại. Miễn là ít nhất một tiến trình cần thực thi, CPU sẽ không bao giờ rảnh.

Ý tưởng này là phổ biến trong các tình huống cuộc sống khác. Ví dụ, một luật sư không chỉ làm việc cho một khách hàng tại một thời điểm. Trong khi một vụ án đang chờ xét xử hoặc có giấy tờ được đánh máy, luật sư có thể giải quyết một vụ án khác. Nếu cô ấy có đủ khách hàng, luật sư sẽ không bao giờ nhàn rỗi vì thiếu việc.

Đa nhiệm (multitasking) là một phần mở rộng hợp lý của đa chương trình. Trong các

[2] Daemon là một loại chương trình trên các hệ điều hành Like-Unix hoạt động ẩn trong background không cần sự kiểm soát bởi user. Daemon sẽ được kích hoạt bởi một sự kiện hoặc điều kiện nào đó xảy ra cụ thể và sẽ phục vụ cho việc trả lời các yêu cầu cho các dịch vụ. Trong Unix, tên của daemon thường kết thúc bằng "d". Một số ví dụ bao gồm inetd, httpd, nfsd, sshd, có tên và lpd.

hệ thống đa nhiệm, CPU thực hiện nhiều tiến trình bằng cách chuyển đổi giữa chúng, nhưng việc chuyển đổi xảy ra thường xuyên, cung cấp cho người dùng sự nhanh chóng **thời gian đáp ứng (respone time)**. Hãy xem xét rằng khi một tiến trình thực thi, nó thường chỉ thực thi trong một thời gian ngắn trước khi kết thúc hoặc cần thực hiện Nhập/Xuất. Nhập/Xuất có thể tương tác; nghĩa là, đầu ra được chuyển đến màn hình cho người dùng và đầu vào đến từ bàn phím, chuột hoặc màn hình cảm ứng của người dùng. Vì Nhập/Xuất tương tác thường chạy ở "tốc độ con người", nên có thể mất nhiều thời gian để hoàn thành. Ví dụ: đầu vào có thể bị giới hạn bởi tốc độ nhập của người dùng; bảy ký tự mỗi giây là nhanh đối với con người nhưng lại cực kỳ chậm đối với máy tính. Thay vì để CPU không hoạt động khi quá trình nhập tương tác này diễn ra, hệ điều hành sẽ nhanh chóng chuyển CPU sang một tiến trình khác.



Hình 1.12 Bố trí bộ nhớ cho hệ thống đa chương trình.

Việc có một số tiến trình trong bộ nhớ cùng một lúc đòi hỏi một số hình thức quản lý bộ nhớ, mà chúng tôi đề cập trong Chương 9 và Chương 10. Ngoài ra, nếu một số tiến trình sẵn sàng chạy cùng một lúc, hệ thống phải chọn tiến trình nào sẽ chạy tiếp theo. Việc đưa ra quyết định này là **lập lịch CPU (CPU scheduling)**, được thảo luận trong Chương 5. Cuối cùng, việc chạy đồng thời nhiều tiến trình đòi hỏi khả năng ảnh hưởng đến nhau của chúng bị hạn chế trong tất cả các giai đoạn của hệ điều hành, bao gồm lập lịch tiến trình, lưu trữ đĩa và quản lý bộ nhớ. Chúng tôi thảo luận về những cân nhắc này trong suốt văn bản.

Trong hệ thống đa nhiệm, hệ điều hành phải đảm bảo thời gian đáp ứng hợp lý. Một phương pháp phổ biến để làm như vậy là **bộ nhớ ảo (virtual memory)**, một kỹ thuật cho phép thực hiện một tiến trình không hoàn toàn trong bộ nhớ (Chương 10). Ưu điểm chính của lược đồ này là nó cho phép người dùng chạy các chương trình lớn hơn **bộ nhớ vật lý (physical memory)** thực tế. Hơn nữa, nó trừu tượng hóa bộ nhớ chính thành một mảng lưu trữ lớn, đồng nhất, tách **bộ nhớ luận lý (logical memory)** được người dùng xem khỏi bộ nhớ vật lý. Sự sắp xếp này giải phóng các lập trình viên khỏi mối quan tâm về giới hạn bộ nhớ–lưu trữ.

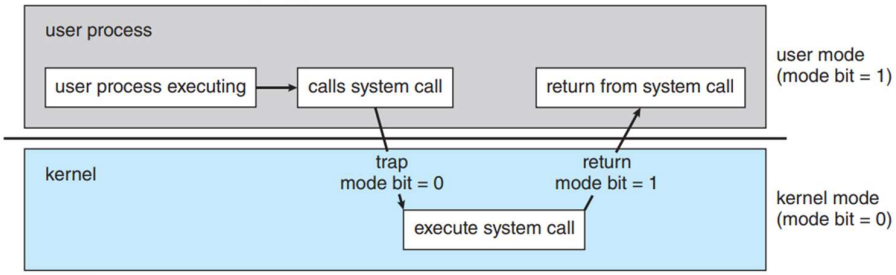
Các hệ thống đa chương trình và đa nhiệm cũng phải cung cấp một hệ thống tập tin (Chương 13, Chương 14 và Chương 15). Hệ thống tập tin nằm trên bộ lưu trữ thứ cấp; do đó, quản lý lưu trữ phải được cung cấp (Chương 11). Ngoài ra, một hệ thống phải bảo vệ tài nguyên khỏi việc sử dụng không phù hợp (Chương 17). Để đảm bảo thực thi có trật tự, hệ thống cũng phải cung cấp các cơ chế để đồng bộ hóa tiến trình và giao tiếp (Chương 6 và Chương 7), và

nó có thể đảm bảo rằng các tiến trình không bị mắc kẹt trong tắc nghẽn, mãi mãi chờ đợi nhau (Chương 8).

1.4.2 Hoạt động ở chế độ kép và đa chế độ

Vì hệ điều hành và người dùng chia sẻ tài nguyên phần cứng và phần mềm của hệ thống máy tính, một hệ điều hành được thiết kế đúng cách phải đảm bảo rằng một chương trình không chính xác (hoặc độc hại) không thể khiến các chương trình khác – hoặc chính hệ điều hành – thực thi không chính xác. Để đảm bảo hệ thống thực thi đúng cách, chúng ta phải có khả năng phân biệt giữa việc thực thi mã hệ điều hành và mã do người dùng xác định. Cách tiếp cận được thực hiện bởi hầu hết các hệ thống máy tính là cung cấp hỗ trợ phần cứng cho phép phân biệt giữa các phương thức thực thi khác nhau.

Ít nhất, chúng ta cần hai chế độ hoạt động riêng biệt: **chế độ người dùng (user mode)** và **chế độ nhân [3](kernel mode)** (còn được gọi là **chế độ giám sát (supervisor mode)**, **chế độ hệ thống (system mode)** hoặc **chế độ đặc quyền (previleged mode)**). Một bit, được gọi là bit chế độ, được thêm vào phần cứng của máy tính để chỉ ra chế độ hiện tại: nhân (0) hoặc người dùng (1). Với bit chế độ, chúng ta có thể phân biệt giữa tác vụ được thực thi thay mặt cho hệ điều hành và tác vụ được thực thi thay mặt cho người dùng. Khi hệ thống máy tính đang thực thi nhân danh ứng dụng người dùng, hệ thống đang ở chế độ người dùng. Tuy nhiên, khi ứng dụng người dùng yêu cầu một dịch vụ từ hệ điều hành (thông qua lệnh gọi hệ thống), hệ thống phải chuyển từ chế độ người dùng sang chế độ hạt nhân để thực hiện yêu cầu. Điều này được thể hiện trong Hình 1.13. Như chúng ta sẽ thấy, cải tiến kiến trúc này cũng hữu ích cho nhiều khía cạnh khác của hoạt động hệ thống.



Hình 1. 13 Chuyển dịch từ chế độ người dùng sang chế độ nhân.

Tại thời điểm khởi động hệ thống, phần cứng khởi động ở chế độ nhân. Hệ điều hành sau đó được tải và khởi động các ứng dụng người dùng ở chế độ người dùng. Bất cứ khi nào xảy ra bẫy hoặc ngắt, phần cứng sẽ chuyển từ chế độ người dùng sang chế độ nhân (nghĩa là thay đổi trạng thái của bit chế độ thành 0). Do đó, bất cứ khi nào hệ điều hành giành được quyền kiểm soát máy tính, nó sẽ ở chế độ nhân. Hệ thống luôn chuyển sang chế độ người dùng (bằng cách đặt bit chế độ thành 1) trước khi chuyển quyền điều khiển cho chương trình người dùng.

Chế độ hoạt động kép cung cấp cho chúng ta các phương tiện để bảo vệ hệ điều hành

[3] Lưu ý rằng "Nhân" khi được nhắc đến trong ngữ cảnh hệ điều hành được hiểu là những chức năng cốt lõi. Còn "Nhân" của CPU là một bản mạch điện tử phần cứng.

khỏi những người dùng sai lầm – và những người dùng sai lầm lẫn nhau. Chúng ta thực hiện biện pháp bảo vệ này bằng cách chỉ định một số lệnh máy có thể gây hại là **lệnh đặc quyền (privileged instruction)**. Phần cứng chỉ cho phép các lệnh đặc quyền được thực thi trong chế độ hạt nhân. Nếu cố gắng thực hiện một lệnh đặc quyền trong chế độ người dùng, phần cứng sẽ không thực hiện lệnh đó mà coi nó là bất hợp pháp và bẫy nó vào hệ điều hành.

Lệnh chuyển sang chế độ hạt nhân là một ví dụ về lệnh đặc quyền. Một số ví dụ khác bao gồm điều khiển Nhập/Xuất, quản lý bộ hẹn giờ và quản lý ngắt. Nhiều lệnh đặc quyền bổ sung được thảo luận trong suốt sách.

Khái niệm về chế độ có thể được mở rộng ra ngoài hai chế độ. Ví dụ, bộ xử lý Intel có bốn **vòng bảo vệ (protection ring)** riêng biệt, trong đó vòng 0 là chế độ nhân và vòng 3 là chế độ người dùng. (Mặc dù các vòng 1 và 2 có thể được sử dụng cho các dịch vụ hệ điều hành khác nhau, nhưng trên thực tế, chúng hiếm khi được sử dụng.) Hệ thống ARMv8 có bảy chế độ. Các CPU hỗ trợ ảo hóa (Phần 18.1) thường có một chế độ riêng để cho biết khi nào **trình quản lý máy ảo (VMM *vt.* virtual machine manager)** kiểm soát hệ thống. Trong chế độ này, VMM có nhiều đặc quyền hơn các tiến trình của người dùng nhưng ít hơn hạt nhân. Nó cần mức đặc quyền đó để nó có thể tạo và quản lý các máy ảo, thay đổi trạng thái CPU để làm như vậy.

Bây giờ chúng ta có thể hiểu rõ hơn về vòng đời của việc thực thi lệnh trong hệ thống máy tính. Kiểm soát ban đầu nằm trong hệ điều hành, nơi các lệnh được thực thi trong chế độ hạt nhân. Khi quyền điều khiển được trao cho một ứng dụng người dùng, chế độ được đặt thành chế độ người dùng. Cuối cùng, quyền điều khiển được chuyển trở lại hệ điều hành thông qua ngắt, bẫy hoặc lệnh gọi hệ thống. Hầu hết các hệ điều hành hiện đại – chẳng hạn như Microsoft Windows, Unix và Linux – tận dụng tính năng chế độ kép này và cung cấp khả năng bảo vệ tốt hơn cho hệ điều hành.

Lệnh gọi hệ thống cung cấp cách thức để chương trình người dùng yêu cầu hệ điều hành thay mặt chương trình người dùng thực hiện các tác vụ dành riêng cho hệ điều hành. Một lệnh gọi hệ thống được gọi theo nhiều cách khác nhau, tùy thuộc vào chức năng được cung cấp bởi bộ xử lý bên dưới. Trong tất cả các hình thức, nó là phương thức được sử dụng bởi một tiến trình để yêu cầu hệ điều hành hành động. Một lệnh gọi hệ thống thường có dạng một cái bẫy đến một vị trí cụ thể trong vector ngắt. Bẫy này có thể được thực hiện bằng lệnh bẫy chung, mặc dù một số hệ thống có lệnh gọi tổng hợp cụ thể để gọi lệnh gọi hệ thống.

Khi một lệnh gọi hệ thống được thực thi, nó thường được phần cứng coi là ngắt phần mềm. Điều khiển truyền qua vectơ ngắt tới một tiến trình dịch vụ trong hệ điều hành và bit chế độ được đặt thành chế độ nhân. Tiến trình gọi dịch vụ hệ thống là một phần của hệ điều hành. Nhân kiểm tra lệnh ngắt để xác định lệnh gọi hệ thống nào đã xảy ra; một tham số cho biết loại dịch vụ mà chương trình người dùng đang yêu cầu. Thông tin bổ sung cần thiết cho yêu cầu có thể được chuyển trong các thanh ghi, trên ngăn xếp, hoặc trong bộ nhớ (với các con trỏ đến các vị trí bộ nhớ được truyền trong các thanh ghi). Nhân xác minh rằng các tham số là chính xác và hợp pháp, thực hiện yêu cầu và trả lại quyền điều khiển cho lệnh sau lệnh gọi hệ thống. Chúng tôi mô tả các lệnh gọi hệ thống đầy đủ hơn trong Phần 2.3.

Sau khi bảo vệ phần cứng được thực hiện, nó sẽ phát hiện các lỗi vi phạm các chế độ.

Những lỗi này thường được xử lý bởi hệ điều hành. Nếu chương trình người dùng bị lỗi theo một cách nào đó – chẳng hạn như cố gắng thực hiện một lệnh bất hợp pháp hoặc truy cập bộ nhớ không có trong không gian địa chỉ của người dùng – thì phần cứng sẽ mắc vào hệ điều hành. Bẫy chuyển điều khiển thông qua vectơ ngắt đến hệ điều hành, giống như một ngắt thực hiện. Khi xảy ra lỗi chương trình, hệ điều hành phải kết thúc chương trình một cách bất thường. Tình huống này được xử lý bằng mã tương tự như khi chấm dứt bất thường do người dùng yêu cầu. Một thông báo lỗi thích hợp được đưa ra và bộ nhớ của chương trình có thể bị kết xuất. Kết xuất bộ nhớ thường được ghi vào một tập tin để người dùng hoặc lập trình viên có thể kiểm tra nó và có thể sửa nó và khởi động lại chương trình.

1.4.3 Bộ định thời

BỘ HẸN GIỜ LINUX

Trên các hệ thống Linux, tham số cấu hình hạt nhân HZ chỉ định tần suất ngắt bộ hẹn giờ. Giá trị HZ là 250 có nghĩa là bộ hẹn giờ tạo ra 250 lần ngắt mỗi giây hoặc một lần ngắt sau mỗi 4 mili-giây. Giá trị của HZ phụ thuộc vào cách cấu hình nhân, cũng như kiểu máy và kiến trúc mà nó đang chạy. Một biến trong nhân có liên quan là `jiffies`, đại diện cho số lần ngắt bộ định thời đã xảy ra kể từ khi hệ thống được khởi động. Một dự án lập trình trong Chương 2 khám phá thêm về thời gian trong nhân Linux.

Chúng ta phải đảm bảo rằng hệ điều hành duy trì quyền kiểm soát đối với CPU. Chúng ta không thể cho phép một chương trình người dùng bị mắc kẹt trong một vòng lặp vô hạn hoặc không gọi được các dịch vụ hệ thống và không bao giờ trả lại quyền kiểm soát cho hệ điều hành. Để thực hiện mục tiêu này, chúng ta có thể sử dụng **bộ hẹn giờ (timer)**. Bộ hẹn giờ có thể được đặt để ngắt máy tính sau một khoảng thời gian nhất định. Khoảng thời gian có thể cố định (ví dụ: 1/60 giây) hoặc biến động (ví dụ: từ 1 mili-giây đến 1 giây). Một **bộ hẹn giờ thay đổi (varible timer)** thường được thực hiện bởi một đồng hồ tốc độ cố định và một bộ đếm. Hệ điều hành đặt bộ đếm. Mỗi khi đồng hồ tích tắc, bộ đếm được giảm dần. Khi bộ đếm về 0, ngắt xảy ra. Ví dụ: bộ đếm 10 bit với đồng hồ 1 mili-giây cho phép ngắt ở các khoảng thời gian từ 1 mili-giây đến 1,024 mili-giây, với mỗi bước là 1 mili-giây.

Trước khi chuyển quyền điều khiển cho người dùng, hệ điều hành đảm bảo rằng bộ hẹn giờ được đặt để ngắt. Nếu bộ hẹn giờ ngắt, điều khiển sẽ tự động chuyển đến hệ điều hành, điều này có thể coi việc ngắt là một lỗi nghiêm trọng hoặc có thể cho chương trình thêm thời gian. Rõ ràng, các lệnh sửa đổi nội dung của bộ hẹn giờ là lệnh đặc quyền.

1.5 Quản lý tài nguyên

Như chúng ta đã thấy, một hệ điều hành là một **trình quản lý tài nguyên (resource manager)**. CPU, không gian bộ nhớ, không gian lưu trữ tập tin và thiết bị Nhập/Xuất của hệ thống nằm trong số các tài nguyên mà hệ điều hành phải quản lý.

1.5.1 Quản lý tiến trình

Một chương trình không thể làm gì trừ khi các lệnh của nó được CPU thực thi. Một chương trình đang thực thi, như đã đề cập, là một tiến trình. Một chương trình như trình biên dịch là một tiến trình và một chương trình xử lý văn bản đang được chạy bởi một người dùng cá nhân trên PC là một tiến trình. Tương tự, một ứng dụng mạng xã hội trên thiết bị di động là một tiến trình. Hiện tại, bạn có thể coi một tiến trình là một thể hiện của một chương trình đang được thực thi, nhưng sau này bạn sẽ thấy rằng khái niệm này tổng quát hơn. Như được mô tả trong Chương 3, có thể cung cấp các lệnh gọi hệ thống cho phép các tiến trình tạo ra các tiến trình con để thực thi đồng thời.

Một tiến trình cần một số tài nguyên nhất định – bao gồm thời gian dùng CPU, bộ nhớ, tập tin và thiết bị Nhập/Xuất – để hoàn thành nhiệm vụ của nó. Các tài nguyên này thường được cấp phát cho tiến trình khi nó đang chạy. Ngoài các tài nguyên vật lý và luận lý khác nhau mà một tiến trình có được khi nó được tạo, nhiều dữ liệu khởi tạo (đầu vào) khác nhau có thể được chuyển cùng. Ví dụ: hãy xem xét một tiến trình chạy trình duyệt web có chức năng là hiển thị nội dung của trang web trên màn hình. Quá trình sẽ được cung cấp URL làm đầu vào và sẽ thực hiện các lệnh thích hợp và lệnh gọi hệ thống để lấy và hiển thị thông tin mong muốn trên màn hình. Khi quá trình kết thúc, hệ điều hành sẽ lấy lại mọi tài nguyên có thể tái sử dụng.

Chúng tôi nhấn mạnh rằng bản thân một chương trình không phải là một tiến trình. Một chương trình là một thực thể thụ động, giống như nội dung của một tập tin được lưu trữ trên đĩa, trong khi một tiến trình là một thực thể hoạt động. Một tiến trình đơn luồng có một **bộ đếm chương trình (program counter)** chỉ định lệnh tiếp theo để thực thi. (Các luồng được đề cập trong Chương 4.) Việc thực hiện một tiến trình như vậy phải theo trình tự. CPU thực hiện hết lệnh của tiến trình này đến lệnh khác cho đến khi tiến trình hoàn tất. Hơn nữa, tại bất kỳ thời điểm nào, chỉ có nhiều nhất một lệnh được thực hiện thay mặt cho tiến trình. Do đó, mặc dù hai tiến trình có thể được liên kết với cùng một chương trình, nhưng chúng vẫn được coi là hai trình tự thực thi riêng biệt. Một tiến trình đa luồng có nhiều bộ đếm chương trình, mỗi bộ đếm chỉ đến lệnh tiếp theo để thực thi cho một luồng nhất định.

Tiến trình là đơn vị công việc trong một hệ thống. Một hệ thống bao gồm một tập hợp các tiến trình, một số trong số đó là các tiến trình của hệ điều hành (những tiến trình thực thi mã hệ thống) và phần còn lại là các tiến trình của người dùng (những tiến trình thực thi mã người dùng). Tất cả các tiến trình này có khả năng thực thi đồng thời – bằng cách ghép kênh trên một lõi CPU – hoặc song song trên nhiều lõi CPU.

Hệ điều hành chịu trách nhiệm về các hoạt động sau liên quan đến quản lý tiến trình:

- Tạo và xóa cả tiến trình của người dùng và hệ thống
- Lập lịch các tiến trình và luồng trên CPU
- Tạm dừng và tiếp tục tiến trình
- Cung cấp cơ chế đồng bộ hóa tiến trình
- Cung cấp cơ chế giao tiếp tiến trình

Chúng ta thảo luận về các kỹ thuật quản lý tiến trình từ Chương 3 đến Chương 7.

1.5.2 Quản lý bộ nhớ

Như đã thảo luận trong Phần 1.2.2, bộ nhớ chính là trung tâm của hoạt động của một hệ thống máy tính hiện đại. Bộ nhớ chính là một mảng lớn các byte, có kích thước từ hàng trăm nghìn đến hàng tỷ. Mỗi byte có địa chỉ riêng của nó. Bộ nhớ chính là một kho lưu trữ dữ liệu có thể truy cập nhanh chóng được chia sẻ bởi CPU và các thiết bị Nhập/Xuất. CPU đọc các lệnh từ bộ nhớ chính trong chu kỳ tìm nạp lệnh và cả đọc và ghi dữ liệu từ bộ nhớ chính trong chu kỳ tìm nạp dữ liệu (trên kiến trúc von Neumann). Như đã lưu ý trước đó, bộ nhớ chính thường là thiết bị lưu trữ lớn duy nhất mà CPU có thể xử lý và truy cập trực tiếp. Ví dụ, để CPU xử lý dữ liệu từ đĩa, trước tiên những dữ liệu đó phải được chuyển vào bộ nhớ chính bằng các lệnh gọi Nhập/Xuất do CPU tạo ra. Theo cách tương tự, các lệnh phải nằm trong bộ nhớ để CPU thực thi chúng.

Để một chương trình được thực thi, nó phải được ánh xạ tới các địa chỉ tuyệt đối và được tải vào bộ nhớ. Khi chương trình thực thi, nó truy cập các hướng dẫn chương trình và dữ liệu từ bộ nhớ bằng cách tạo ra các địa chỉ tuyệt đối này. Cuối cùng, chương trình kết thúc, không gian bộ nhớ của nó được khai báo là có sẵn, và chương trình tiếp theo có thể được tải và thực thi.

Để cải thiện cả việc sử dụng CPU và tốc độ phản hồi của máy tính với người dùng, các máy tính đa năng phải lưu giữ một số chương trình trong bộ nhớ, tạo ra nhu cầu quản lý bộ nhớ. Nhiều lược đồ quản lý bộ nhớ khác nhau được sử dụng. Các lược đồ này phản ánh nhiều cách tiếp cận khác nhau và hiệu quả của bất kỳ thuật toán nhất định nào phụ thuộc vào tình huống. Khi lựa chọn một sơ đồ quản lý bộ nhớ cho một hệ thống cụ thể, chúng ta phải tính đến nhiều yếu tố – đặc biệt là ***thiết kế phần cứng*** của hệ thống. Mỗi thuật toán yêu cầu hỗ trợ phần cứng riêng.

Hệ điều hành chịu trách nhiệm về các hoạt động sau liên quan đến quản lý bộ nhớ:

- Theo dõi phần nào của bộ nhớ hiện đang được sử dụng và tiến trình nào đang sử dụng chúng
- Phân bổ và giải quyết không gian bộ nhớ khi cần thiết
- Quyết định tiến trình nào (hoặc các phần của tiến trình) và dữ liệu để di chuyển vào và ra khỏi bộ nhớ

Các kỹ thuật quản lý bộ nhớ được thảo luận trong Chương 9 và Chương 10

1.5.3 Quản lý hệ thống tập tin

Để làm cho hệ thống máy tính thuận tiện cho người sử dụng, hệ điều hành cung cấp một cái nhìn thống nhất và logic về lưu trữ thông tin. Hệ điều hành tóm tắt từ các thuộc tính vật lý của các thiết bị lưu trữ của nó để xác định một đơn vị lưu trữ logic, gọi là **tập tin (file)**. Hệ điều hành ánh xạ các tập tin lên phương tiện vật lý và truy cập các tập tin này qua thiết bị lưu trữ.

Quản lý tập tin là một trong những thành phần dễ thấy nhất của hệ điều hành. Máy tính có thể lưu trữ thông tin trên một số loại phương tiện vật lý khác nhau. Lưu trữ thứ cấp là phổ biến nhất, nhưng lưu trữ cấp ba cũng có thể. Mỗi phương tiện này có đặc điểm và tổ chức vật lý riêng. Hầu hết được điều khiển bởi một thiết bị, chẳng hạn như ổ đĩa, cũng có những đặc điểm riêng biệt. Các thuộc tính này bao gồm tốc độ truy cập, dung lượng, tốc độ truyền dữ liệu và phương thức truy cập (tuần tự hoặc ngẫu nhiên).

Tập tin là một tập hợp các thông tin liên quan do người tạo ra nó xác định. Thông thường, tập tin đại diện cho chương trình (cả dạng nguồn và dạng đối tượng) và dữ liệu. Tập tin dữ liệu có thể là số, chữ cái, chữ và số hoặc nhị phân. Các tập tin có thể ở dạng tự do (ví dụ: tập tin văn bản) hoặc chúng có thể được định dạng cứng nhắc (ví dụ: các trường cố định như tập tin nhạc mp3). Rõ ràng, khái niệm tập tin là một khái niệm cực kỳ chung chung.

Hệ điều hành thực hiện khái niệm trừu tượng của một tập tin bằng cách quản lý các phương tiện lưu trữ đại chúng và các thiết bị điều khiển chúng. Ngoài ra, các tập tin thường được tổ chức thành các thư mục để dễ sử dụng hơn. Cuối cùng, khi nhiều người dùng có quyền truy cập vào tập tin, có thể mong muốn kiểm soát người dùng nào có thể truy cập tập tin và cách người dùng đó có thể truy cập tập tin đó (ví dụ: đọc, ghi, nối thêm).

Hệ điều hành chịu trách nhiệm về các hoạt động sau liên quan đến quản lý tập tin:

- Tạo và xóa tập tin
- Tạo và xóa thư mục để sắp xếp tập tin
- Hỗ trợ các nguyên tắc cơ bản để thao tác tập tin và thư mục
- Ánh xạ tập tin vào bộ nhớ chung
- Sao lưu tập tin trên ổn định (không thay đổi) phương tiện lưu trữ

Các kỹ thuật quản lý tập tin được thảo luận trong Chương 13, Chương 14 và Chương 15.

1.5.4 Quản lý lưu trữ lớn

Như chúng ta đã thấy, hệ thống máy tính phải cung cấp bộ nhớ phụ để sao lưu bộ nhớ chính. Hầu hết các hệ thống máy tính hiện đại sử dụng ổ cứng HDD và thiết bị NVM làm phương tiện lưu trữ trực tuyến chính cho cả chương trình và dữ liệu. Hầu hết các chương trình – bao gồm trình biên dịch, trình duyệt web, trình xử lý văn bản và trò chơi – được lưu trữ trên các thiết bị này cho đến khi được tải vào bộ nhớ. Sau đó, các chương trình sử dụng các thiết bị vừa là nguồn vừa là đích xử lý của chúng. Do đó, việc quản lý thích hợp lưu trữ thứ cấp có tầm quan trọng trung tâm đối với một hệ thống máy tính.

Hệ điều hành chịu trách nhiệm về các hoạt động sau liên quan đến quản lý bộ nhớ thứ cấp:

- Gắn vào (mounting) và tách ra.
- Quản lý dung lượng trống

- Phân bổ bộ nhớ
- Lập lịch đĩa
- Phân vùng
- Bảo vệ

Vì bộ nhớ thứ cấp được sử dụng thường xuyên và rộng rãi nên nó phải được sử dụng một cách hiệu quả. Toàn bộ tốc độ hoạt động của máy tính có thể phụ thuộc vào tốc độ của hệ thống con lưu trữ thứ cấp và các thuật toán điều khiển hệ thống con đó.

Đồng thời, có nhiều cách sử dụng để lưu trữ chậm hơn và chi phí thấp hơn (và đôi khi là dung lượng cao hơn) so với lưu trữ thứ cấp. Sao lưu dữ liệu đĩa, lưu trữ dữ liệu hiếm khi được sử dụng và lưu trữ lâu dài là một số ví dụ. Ổ đĩa từ và băng từ của chúng và ổ đĩa CD DVD và Blu–ray và đĩa cứng là những thiết bị lưu trữ cấp ba điển hình.

Lưu trữ cấp ba không quan trọng đối với hiệu suất của hệ thống, nhưng nó vẫn phải được quản lý. Một số hệ điều hành đảm nhận nhiệm vụ này, trong khi những hệ điều hành khác để lại quyền quản lý lưu trữ cấp ba cho các chương trình ứng dụng. Một số chức năng mà hệ điều hành có thể cung cấp bao gồm gắn và ngắt kết nối phương tiện trong thiết bị, cấp phát và giải phóng thiết bị để các tiến trình sử dụng riêng và di chuyển dữ liệu từ bộ nhớ thứ cấp sang bộ nhớ thứ ba. Các kỹ thuật quản lý lưu trữ thứ cấp và lưu trữ cấp ba được thảo luận trong Chương 11

1.5.5 Cache Management

Bộ nhớ đệm (caching[4]) là một nguyên tắc quan trọng của hệ thống máy tính. Đây là cách nó hoạt động. Thông tin thường được lưu giữ trong một số hệ thống lưu trữ (chẳng hạn như bộ nhớ chính). Khi nó được sử dụng, nó được sao chép vào một hệ thống lưu trữ nhanh hơn – bộ nhớ đệm – trên nguyên lý tạm thời. Khi chúng ta cần một thông tin cụ thể, trước tiên chúng ta kiểm tra xem nó có nằm trong bộ nhớ đệm hay không. Nếu có, chúng ta sử dụng thông tin trực tiếp từ bộ nhớ đệm. Nếu không, chúng tôi sử dụng thông tin từ nguồn, đưa một bản sao vào bộ nhớ đệm với giả định rằng chúng tôi sẽ sớm cần lại.

[4] Caching là kỹ thuật vùng nhớ đệm làm trung gian khi giao tiếp giữa hai thiết bị phần cứng cách biệt về tốc độ xử lý hoặc là không đồng bộ với nhau.

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

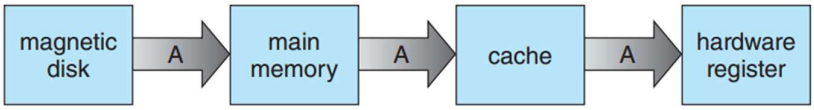
Hình 1.14 Đặc trưng của một số loại thiết bị lưu trữ.

Ngoài ra, các thanh ghi có thể lập trình bên trong cung cấp một bộ đệm tốc độ cao cho bộ nhớ chính. Lập trình viên (hoặc trình biên dịch) thực hiện các thuật toán phân bổ thanh ghi và thay thế thanh ghi để quyết định thông tin nào cần giữ trong các thanh ghi và thông tin nào sẽ giữ trong bộ nhớ chính.

Các bộ nhớ đệm khác được thực hiện hoàn toàn trong phần cứng. Ví dụ, hầu hết các hệ thống đều có bộ nhớ đệm chứa lệnh để lưu giữ các lệnh dự kiến sẽ được thực thi tiếp theo. Nếu không có bộ nhớ đệm này, CPU sẽ phải đợi vài chu kỳ trong khi một lệnh được tìm nạp từ bộ nhớ chính. Vì những lý do tương tự, hầu hết các hệ thống có một hoặc nhiều bộ nhớ đệm dữ liệu tốc độ cao trong hệ thống phân cấp bộ nhớ. Chúng tôi không quan tâm đến những bộ đệm chỉ dành cho phần cứng trong văn bản này, vì chúng nằm ngoài sự kiểm soát của hệ điều hành.

Bởi vì bộ nhớ đệm có kích thước hạn chế, **quản lý bộ nhớ đệm (cache management)** là một vấn đề thiết kế quan trọng. Việc lựa chọn cẩn thận kích thước bộ nhớ cache và chính sách thay thế có thể dẫn đến hiệu suất tăng lên đáng kể, như bạn có thể thấy bằng cách xem Hình 1.14. Các thuật toán thay thế cho bộ nhớ đệm do phần mềm điều khiển được thảo luận trong Chương 10.

Sự di chuyển thông tin giữa các cấp của hệ thống phân cấp lưu trữ có thể rõ ràng hoặc ẩn, tùy thuộc vào thiết kế phần cứng và phần mềm hệ điều hành điều khiển. Ví dụ, truyền dữ liệu từ bộ nhớ cache đến CPU và thanh ghi thường là một chức năng phần cứng, không có sự can thiệp của hệ điều hành. Ngược lại, việc chuyển dữ liệu từ đĩa sang bộ nhớ thường do hệ điều hành kiểm soát. Trong cấu trúc lưu trữ phân cấp, dữ liệu giống nhau có thể xuất hiện ở các cấp khác nhau của hệ thống lưu trữ. Ví dụ, giả sử rằng một số nguyên A được tăng thêm 1 nằm trong tập tin B và tập tin B nằm trên đĩa cứng. Hoạt động gia tăng tiến hành bằng cách đưa ra một thao tác Nhập/Xuất trước tiên để sao chép khối đĩa mà A nằm trên đó vào bộ nhớ chính. Thao tác này được thực hiện sau bằng cách sao chép A vào bộ nhớ cache và vào một thanh ghi bên trong. Do đó, bản sao của A xuất hiện ở một số nơi: trên đĩa cứng, trong bộ nhớ chính, trong bộ đệm và trong một thanh ghi bên trong (xem Hình 1.15). Khi quá trình tăng diễn ra trong thanh ghi bên trong, giá trị của A sẽ khác trong các hệ thống lưu trữ khác nhau. Giá trị của A chỉ trở nên giống nhau sau khi giá trị mới của A được ghi từ thanh ghi bên trong trở lại đĩa cứng.



Hình 1.15 Sự di chuyển của số nguyên A từ đĩa vào thanh ghi.

Trong môi trường máy tính chỉ có một tiến trình thực thi tại một thời điểm, cách sắp xếp này không gây khó khăn gì, vì quyền truy cập vào số nguyên A sẽ luôn là bản sao ở cấp cao nhất của hệ thống phân cấp. Tuy nhiên, trong môi trường đa nhiệm, nơi CPU được chuyển đổi qua lại giữa các tiến trình khác nhau, cần hết sức thận trọng để đảm bảo rằng, nếu một số tiến trình muốn truy cập A, thì mỗi tiến trình này sẽ nhận được giá trị được cập nhật gần đây nhất của A Tình hình trở nên phức tạp hơn trong môi trường đa xử lý, ngoài việc duy trì các thanh ghi bên trong, mỗi CPU còn chứa một bộ nhớ đệm cục bộ (xem lại Hình 1.8). Trong một môi trường như vậy, một bản sao của A có thể tồn tại đồng thời trong một số bộ nhớ đệm. Vì tất cả các CPU khác nhau đều có thể thực thi song song, chúng ta phải đảm bảo rằng cập nhật giá trị của A trong một bộ nhớ cache được phản ánh ngay lập tức trong tất cả các bộ nhớ cache khác nơi A cư trú. Tình huống này được gọi là **bộ nhớ đệm kết hợp (cache coherency)** và nó thường là sự cố phần cứng (được xử lý dưới cấp hệ điều hành).

Trong một môi trường phân tán, tình hình thậm chí còn trở nên phức tạp hơn. Trong môi trường này, một số bản sao (hoặc bản sao) của cùng một tập tin có thể được lưu giữ trên các máy tính khác nhau. Vì các bản sao khác nhau có thể được truy cập và cập nhật đồng thời, một số hệ thống phân tán đảm bảo rằng, khi một bản sao được cập nhật ở một nơi, tất cả các bản sao khác sẽ được cập nhật càng sớm càng tốt. Có nhiều cách khác nhau để đạt được sự đảm bảo này, như chúng ta đã thảo luận trong Chương 19.

1.5.6 Quản lý hệ thống Nhập/Xuất

Một trong những mục đích của hệ điều hành là che giấu những đặc thù của các thiết bị phần cứng cụ thể khỏi người dùng. Ví dụ, trong UNIX, các đặc thù của thiết bị Nhập/Xuất được **hệ thống con Nhập/Xuất (I/O subsystem)** ẩn khỏi phần lớn hệ điều hành. Hệ thống con Nhập/Xuất bao gồm một số thành phần:

- Thành phần quản lý bộ nhớ bao gồm bộ đệm (buffering), bộ nhớ đệm (caching) và cuộn dữ liệu (spooling[5])
- Giao diện thiết bị – trình điều khiển chung
- Trình điều khiển cho các thiết bị phần cứng cụ thể

Chỉ trình điều khiển thiết bị mới biết các đặc điểm của thiết bị cụ thể mà nó được chỉ định.

Chúng ta đã thảo luận trước đó trong chương này về cách sử dụng trình xử lý ngắt và trình điều khiển thiết bị trong việc xây dựng các hệ thống con Nhập/Xuất hiệu quả. Trong

[5] Là động tác lưu dữ liệu tài liệu trong một hàng đợi, ở đó dữ liệu chờ tới lượt được truy cập.

Chương 12, chúng ta thảo luận về cách hệ thống con Nhập/Xuất giao tiếp với các thành phần hệ thống khác, quản lý thiết bị, truyền dữ liệu và phát hiện quá trình hoàn thành Nhập/Xuất.

1.6 Bảo mật và bảo vệ

Nếu một hệ thống máy tính có nhiều người dùng và cho phép thực hiện đồng thời nhiều quá trình, thì quyền truy cập vào dữ liệu phải được quy định. Với mục đích đó, các cơ chế đảm bảo rằng các tập tin, phân đoạn bộ nhớ, CPU và các tài nguyên khác chỉ có thể được vận hành bởi những tiến trình đã được hệ điều hành ủy quyền thích hợp. Ví dụ, phần cứng định địa chỉ bộ nhớ đảm bảo rằng một tiến trình chỉ có thể thực thi trong không gian địa chỉ của chính nó. Bộ đếm thời gian đảm bảo rằng không có quá trình nào có thể giành được quyền kiểm soát CPU mà không từ bỏ quyền kiểm soát cuối cùng. Người dùng không thể truy cập thanh ghi điều khiển thiết bị, vì vậy tính toàn vẹn của các thiết bị ngoại vi khác nhau được bảo vệ.

Vì vậy, **bảo vệ (protection)** là bất kỳ cơ chế nào để kiểm soát quyền truy cập của các tiến trình hoặc người dùng vào các tài nguyên được xác định bởi hệ thống máy tính. Cơ chế này phải cung cấp các phương tiện để chỉ rõ các biện pháp kiểm soát sẽ được áp đặt và để thực thi các biện pháp kiểm soát.

Bảo vệ có thể cải thiện độ tin cậy bằng cách phát hiện các lỗi tiềm ẩn tại các giao diện giữa các hệ thống con thành phần. Việc phát hiện sớm các lỗi giao diện thường có thể ngăn chặn sự lây nhiễm của một hệ thống con khỏe mạnh bởi một hệ thống con khác đang bị trục trặc. Hơn nữa, tài nguyên không được bảo vệ không thể chống lại việc sử dụng (hoặc lạm dụng) bởi người dùng trái phép hoặc không đủ năng lực. Hệ thống hướng đến bảo vệ cung cấp một phương tiện để phân biệt giữa việc sử dụng được phép và không được phép, như chúng ta đã thảo luận trong Chương 17.

Một hệ thống có thể được bảo vệ đầy đủ nhưng vẫn dễ bị lỗi và cho phép truy cập không phù hợp. Hãy xem xét một người dùng có thông tin xác thực (phương tiện nhận dạng của cô ấy đối với hệ thống) bị đánh cắp. Dữ liệu của cô ấy có thể bị sao chép hoặc xóa, mặc dù tính năng bảo vệ tập tin và bộ nhớ đang hoạt động. Nhiệm vụ của **bảo mật (security)** là bảo vệ hệ thống khỏi các cuộc tấn công từ bên ngoài và bên trong. Các cuộc tấn công như vậy lan rộng trên một phạm vi rộng lớn và bao gồm vi rút và sâu máy tính, các cuộc tấn công từ chối dịch vụ (sử dụng tất cả tài nguyên của hệ thống và do đó giữ người dùng hợp pháp không tham gia vào hệ thống), đánh cắp danh tính và đánh cắp dịch vụ (sử dụng trái phép hệ thống). Phòng chống một số các cuộc tấn công này được coi là một chức năng của hệ điều hành trên một số hệ thống, trong khi các hệ thống khác để nó cho chính sách hoặc phần mềm bổ sung. Do sự gia tăng đáng báo động về các sự cố bảo mật, các tính năng bảo mật của hệ điều hành là một lĩnh vực nghiên cứu và triển khai đang phát triển nhanh chóng. Chúng ta thảo luận về bảo mật trong Chương 16.

Bảo vệ và bảo mật yêu cầu hệ thống phải có khả năng phân biệt giữa tất cả người dùng của nó. Hầu hết các hệ điều hành duy trì một danh sách tên người dùng và **mã định danh người dùng** được liên kết **(user ID *vt.* user indentifier)**. Theo cách nói của Windows, đây là một **ID bảo mật (SID *vt.* security ID)**. Các ID số này là duy nhất, một ID cho mỗi người dùng. Khi người dùng đăng nhập vào hệ thống, giai đoạn xác thực sẽ xác định ID người dùng phù hợp cho người dùng. ID người dùng đó được liên kết với tất cả các tiến trình và chuỗi của

người dùng. Khi một ID cần người dùng có thể đọc được, nó sẽ được dịch ngược lại thành tên người dùng thông qua danh sách tên người dùng.

Trong một số trường hợp, chúng tôi muốn phân biệt giữa các nhóm người dùng hơn là người dùng cá nhân. Ví dụ: chủ sở hữu của một tập tin trên hệ thống UNIX có thể được phép thực hiện tất cả các thao tác trên tập tin đó, trong khi một nhóm người dùng đã chọn có thể chỉ được phép đọc tập tin. Để thực hiện điều này, chúng ta cần xác định tên nhóm và tập hợp người dùng thuộc nhóm đó. Chức năng nhóm có thể được triển khai dưới dạng danh sách toàn hệ thống gồm các tên nhóm và **mã định danh nhóm (group indentifier)**. Người dùng có thể thuộc một hoặc nhiều nhóm, tùy thuộc vào quyết định thiết kế hệ điều hành. ID nhóm của người dùng cũng được bao gồm trong mọi tiến trình và luồng liên quan.

Trong quá trình sử dụng hệ thống bình thường, ID người dùng và ID nhóm cho người dùng là đủ. Tuy nhiên, người dùng đôi khi cần **nâng cấp đặc quyền (escalate privileges)** để có thêm quyền cho một hoạt động. Ví dụ, người dùng có thể cần quyền truy cập vào một thiết bị bị hạn chế. Hệ điều hành cung cấp các phương pháp khác nhau để cho phép leo thang đặc quyền. Ví dụ: trên UNIX, thuộc tính *setuid* trên một chương trình khiến chương trình đó chạy với ID người dùng của chủ sở hữu tập tin, thay vì ID của người dùng hiện tại. Tiến trình chạy với **UID hiệu quả (effective UID)** này cho đến khi nó tắt hoặc chấm dứt các đặc quyền bổ sung.

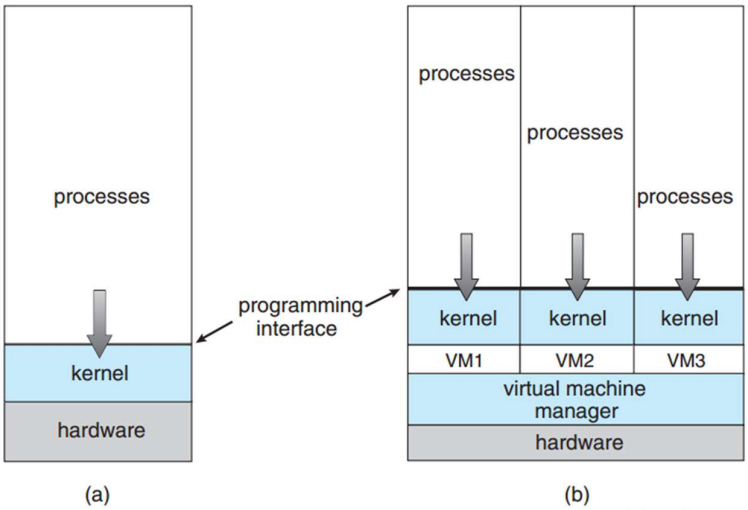
1.7 Ảo hoá

Ảo hóa (virtualization) là một công nghệ cho phép chúng ta trừu tượng hóa phần cứng của một máy tính (CPU, bộ nhớ, ổ đĩa, thẻ giao diện mạng, v.v.) thành một số môi trường thực thi khác nhau, do đó tạo ra ảo tưởng rằng mỗi môi trường riêng biệt đang chạy trên nó riêng máy tính riêng. Các môi trường này có thể được xem như các hệ điều hành riêng lẻ khác nhau (ví dụ: Windows và UNIX) có thể đang chạy cùng lúc và có thể tương tác với nhau. Người dùng **máy ảo (virtual machine)** có thể chuyển đổi giữa các hệ điều hành khác nhau giống như cách người dùng có thể chuyển đổi giữa các quá trình khác nhau chạy đồng thời trong một hệ điều hành.

Ảo hóa cho phép các hệ điều hành chạy như các ứng dụng trong các hệ điều hành khác. Lúc đầu đỏ mặt, dường như có rất ít lý do cho chức năng như vậy. Nhưng ngành công nghiệp ảo hóa rất rộng lớn và ngày càng phát triển, đó là minh chứng cho tiện ích và tầm quan trọng của nó.

Nói một cách rộng rãi, phần mềm ảo hóa là một thành viên của một lớp cũng bao gồm mô phỏng. **Mô phỏng (emulation)**, liên quan đến việc mô phỏng phần cứng máy tính trong phần mềm, thường được sử dụng khi loại CPU nguồn khác với loại CPU đích. Ví dụ: khi Apple chuyển từ CPU IBM Power sang CPU Intel x86 cho máy tính để bàn và máy tính xách tay của mình, nó bao gồm một cơ sở mô phỏng có tên là “Rosetta”, cho phép các ứng dụng được biên dịch cho CPU IBM chạy trên CPU Intel. Khái niệm tương tự đó có thể được mở rộng để cho phép toàn bộ hệ điều hành được viết cho một nền tảng này có thể chạy trên một nền tảng khác. Tuy nhiên, giả lập đi kèm với một cái giá đắt. Mọi lệnh cấp máy chạy nguyên bản trên hệ thống nguồn phải được dịch sang chức năng tương đương trên hệ thống đích, thường dẫn đến một số lệnh đích. Nếu CPU nguồn và CPU đích có mức hiệu suất tương tự, mã giả lập có thể chạy

chậm hơn nhiều so với mã gốc.



Hình 1.16 Một máy tính chạy (a) hệ điều hành duy nhất và (b) ba máy ảo.

Ngược lại, với ảo hóa, một hệ điều hành được biên dịch nguyên bản cho một kiến trúc CPU cụ thể sẽ chạy trong một hệ điều hành khác cũng có nguồn gốc từ CPU đó. Ảo hóa lần đầu tiên xuất hiện trên các máy tính lớn của IBM như một phương pháp để nhiều người dùng có thể chạy các tác vụ đồng thời. Cho phép chạy nhiều máy ảo (và vẫn cho phép) nhiều người dùng chạy các tác vụ trên hệ thống được thiết kế cho một người dùng. Sau đó, để giải quyết các vấn đề khi chạy nhiều ứng dụng Microsoft Windows trên CPU Intel x86, VMware đã tạo ra một công nghệ ảo hóa mới dưới dạng một ứng dụng chạy trên Windows. Ứng dụng đó đã chạy một hoặc nhiều bản sao **khách (guest)** của Windows hoặc các hệ điều hành x86 gốc khác, mỗi bản chạy các ứng dụng riêng của nó. (Xem Hình 1.16.) Windows là hệ điều hành **chủ (host)** và ứng dụng VMware là **trình quản lý máy ảo (VMM** vt. **virtual machine manager)**. VMM chạy hệ điều hành khách, quản lý việc sử dụng tài nguyên của họ và bảo vệ từng khách khỏi những hệ điều hành khác.

Mặc dù các hệ điều hành hiện đại hoàn toàn có khả năng chạy nhiều ứng dụng một cách đáng tin cậy, việc sử dụng ảo hóa vẫn tiếp tục phát triển. Trên máy tính xách tay và máy tính để bàn, VMM cho phép người dùng cài đặt nhiều hệ điều hành để khám phá hoặc chạy các ứng dụng được viết cho hệ điều hành khác với máy chủ gốc. Ví dụ: máy tính xách tay của Apple chạy macOS trên CPU x86 có thể chạy khách Windows 10 để cho phép thực thi các ứng dụng Windows. Các công ty viết phần mềm cho nhiều hệ điều hành có thể sử dụng ảo hóa để chạy tất cả các hệ điều hành đó trên một máy chủ vật lý duy nhất để phát triển, thử nghiệm và gỡ lỗi. Trong các trung tâm dữ liệu, ảo hóa đã trở thành một phương pháp phổ biến để thực thi và quản lý các môi trường máy tính. Các VMM như VMware ESX và Citrix XenServer không còn chạy trên hệ điều hành chủ nữa mà là hệ điều hành chủ, cung cấp dịch vụ và quản lý tài nguyên cho các tiến trình máy ảo.

Với quyển sách này, chúng tôi cung cấp một máy ảo Linux cho phép bạn chạy Linux – cũng như các công cụ phát triển mà chúng tôi cung cấp – trên hệ thống cá nhân của bạn bất kể hệ điều hành máy chủ của bạn là gì. Chi tiết đầy đủ về các tính năng và cách thực hiện ảo hóa có thể được tìm thấy trong Chương 18.

1.8 Hệ thống phân tán

Hệ thống phân tán là một tập hợp các hệ thống máy tính riêng biệt, có thể không đồng nhất về mặt vật lý được nối mạng để cung cấp cho người dùng quyền truy cập vào các tài nguyên khác nhau mà hệ thống duy trì. Quyền truy cập vào tài nguyên được chia sẻ làm tăng tốc độ tính toán, chức năng, tính khả dụng của dữ liệu và độ tin cậy. Một số hệ điều hành khái quát quyền truy cập mạng như một hình thức truy cập tập tin, với các chi tiết về mạng có trong trình điều khiển thiết bị của giao diện mạng Một số hệ điều hành khác yêu cầu người dùng gọi cụ thể các chức năng mạng. Nói chung, các hệ thống có sự kết hợp của hai chế độ – ví dụ FTP và NFS. Các giao thức tạo ra một hệ thống phân tán có thể ảnh hưởng lớn đến tiện ích và tính phổ biến của hệ thống đó.

Nói một cách đơn giản nhất, **mạng (network)** là một đường truyền thông tin giữa hai hoặc nhiều hệ thống. Hệ thống phân tán phụ thuộc vào mạng cho chức năng của chúng. Các mạng khác nhau tùy theo giao thức được sử dụng, khoảng cách giữa các nút và phương tiện truyền tải. **TCP/IP** là giao thức mạng phổ biến nhất và nó cung cấp kiến trúc cơ bản của Internet. Hầu hết các hệ điều hành hỗ trợ TCP / IP, bao gồm tất cả các hệ điều hành có mục đích chung. Một số hệ thống hỗ trợ các giao thức độc quyền để phù hợp với nhu cầu của họ. Đối với một hệ điều hành, chỉ cần giao thức mạng có thiết bị giao diện – bộ điều hợp mạng, ví dụ – với trình điều khiển thiết bị để quản lý nó, cũng như phần mềm để xử lý dữ liệu. Những khái niệm này được thảo luận trong suốt cuốn sách này.

Các mạng được đặc trưng dựa trên khoảng cách giữa các nút của chúng. **Mạng cục bộ (LAN** *vt.* **Local-area network)** kết nối các máy tính trong một căn phòng, một tòa nhà hoặc một khuôn viên trường. **Mạng diện rộng (WAN** *vt.* **Wide-area network)** thường liên kết các tòa nhà, thành phố hoặc quốc gia. Ví dụ, một công ty toàn cầu có thể có một mạng WAN để kết nối các văn phòng của mình trên toàn thế giới. Các mạng này có thể chạy một giao thức hoặc một số giao thức. Sự ra đời liên tục của các công nghệ mới làm xuất hiện các dạng mạng mới. Ví dụ: **mạng khu vực đô thị (MAN** *vt.* **Metropolitan-area network)** có thể liên kết các tòa nhà trong một thành phố. Các thiết bị BlueTooth và 802.11 sử dụng công nghệ không dây để giao tiếp trong khoảng cách vài feet, về bản chất là tạo ra **mạng cá nhân (PAN** *vt.* **Personal-area network)** giữa điện thoại và tai nghe hoặc điện thoại thông minh và máy tính để bàn.

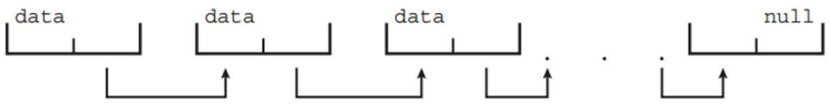
Các phương tiện để thực hiện các mạng cũng đa dạng như nhau. Chúng bao gồm dây đồng, sợi cáp quang và đường truyền không dây giữa các vệ tinh, đĩa vi sóng và radio. Khi các thiết bị máy tính được kết nối với điện thoại di động, chúng sẽ tạo ra một mạng. Ngay cả giao tiếp hồng ngoại tầm ngắn cũng có thể được sử dụng để kết nối mạng. Ở mức độ thô sơ, bất cứ khi nào các máy tính giao tiếp, chúng đều sử dụng hoặc tạo ra một mạng. Các mạng này cũng khác nhau về hiệu suất và độ tin cậy của chúng.

Một số hệ điều hành đã đưa khái niệm mạng và hệ thống phân tán đi xa hơn khái niệm cung cấp kết nối mạng. **Hệ điều hành mạng (network operating system)** là hệ điều hành cung cấp các tính năng như chia sẻ tập tin qua mạng, cùng với một sơ đồ truyền thông cho phép các tiến trình khác nhau trên các máy tính khác nhau trao đổi thông điệp. Một máy tính chạy hệ điều hành mạng hoạt động độc lập với tất cả các máy tính khác trong mạng, mặc dù nó nhận biết được mạng và có thể giao tiếp với các máy tính nối mạng khác. Hệ điều hành phân tán

cung cấp một môi trường ít tự trị hơn. Các máy tính khác nhau giao tiếp đủ chặt chẽ để tạo ra ảo tưởng rằng chỉ có một hệ điều hành duy nhất điều khiển mạng. Chúng tôi đề cập đến mạng máy tính và hệ thống phân tán trong Chương 19.

1.9 Cấu trúc dữ liệu Nhân

Chúng tôi chuyển sang một chủ đề trọng tâm trong việc triển khai hệ điều hành: cách dữ liệu được cấu trúc trong hệ thống. Trong phần này, chúng tôi mô tả ngắn gọn một số cấu trúc dữ liệu cơ bản được sử dụng rộng rãi trong các hệ điều hành. Những độc giả muốn biết thêm chi tiết về những cấu trúc này, cũng như những cấu trúc khác, nên tham khảo phần tài liệu tham khảo ở cuối chương.



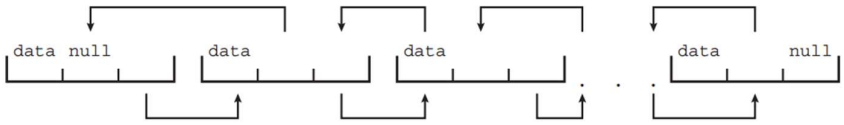
Hình 1.17 Danh sách liên kết đơn.

1.9.1 Danh sách, Ngăn xếp và Hàng chờ

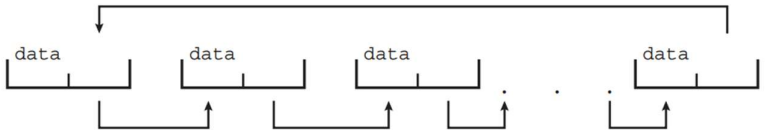
Mảng là một cấu trúc dữ liệu đơn giản, trong đó mỗi phần tử có thể được truy cập trực tiếp. Ví dụ, bộ nhớ chính được xây dựng dưới dạng một mảng. Nếu mục dữ liệu đang được lưu trữ lớn hơn một byte, thì nhiều byte có thể được phân bổ cho mục đó và mục đó có địa chỉ là "số mục × kích thước mục". Nhưng còn việc lưu trữ một món đồ có kích thước có thể khác nhau thì sao? Và việc loại bỏ một món đồ nếu phải giữ nguyên vị trí tương đối của những món đồ còn lại thì sao? Trong những tình huống như vậy, mảng nhường chỗ cho các cấu trúc dữ liệu khác.

Sau mảng, danh sách có lẽ là cấu trúc dữ liệu cơ bản nhất trong khoa học máy tính. Trong khi mỗi mục trong một mảng có thể được truy cập trực tiếp, các mục trong danh sách phải được truy cập theo một thứ tự cụ thể. Có nghĩa là, một **danh sách (list)** đại diện cho một tập hợp các giá trị dữ liệu dưới dạng một chuỗi. Phương pháp phổ biến nhất để thực hiện cấu trúc này là một danh sách liên kết, trong đó các mục được liên kết với nhau. Danh sách được liên kết có một số loại:

- Trong một ***danh sách liên kết đơn (singly linked list)***, mỗi mục trỏ đến mục kế nhiệm của nó, như minh họa trong Hình 1.17.
- Trong ***danh sách liên kết kép (doubly linked list)***, một mục nhất định có thể tham chiếu đến mục tiền nhiệm hoặc mục kế nhiệm của nó, như được minh họa trong Hình 1.18.
- Trong ***danh sách liên kết vòng (circuly linked list)***, phần tử cuối cùng trong danh sách tham chiếu đến phần tử đầu tiên, thay vì null, như minh họa trong Hình 1.19.



Hình 1.18 Danh sách liên kết kép.



Hình 1.19 Danh sách liên kết vòng.

Danh sách được liên kết chứa các mục có kích thước khác nhau và cho phép dễ dàng chèn và xóa các mục. Một nhược điểm tiềm ẩn của việc sử dụng danh sách là hiệu suất để truy xuất một mục được chỉ định trong danh sách có kích thước *n* là tuyến tính – O(*n*), vì nó yêu cầu khả năng duyệt qua tất cả *n* phần tử trong trường hợp xấu nhất. Danh sách đôi khi được sử dụng trực tiếp bởi các thuật toán của nhân. Tuy nhiên, chúng thường được sử dụng để xây dựng các cấu trúc dữ liệu mạnh mẽ hơn, chẳng hạn như ngăn xếp và hàng đợi.

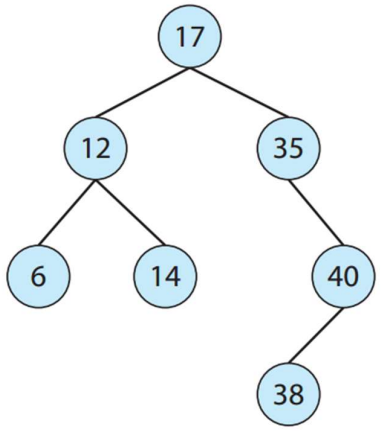
Ngăn xếp (stack) là một cấu trúc dữ liệu được sắp xếp theo thứ tự sử dụng nguyên tắc vào cuối cùng, ra trước tiên **(LIFO** *vt.* **Last in, First out)** để thêm và xóa các mục, có nghĩa là mục cuối cùng được đặt trên một ngăn xếp là mục đầu tiên bị xóa. Các hoạt động để chèn (***push***) và loại bỏ (***pop***) các mục khỏi ngăn xếp được gọi là đẩy và bật, tương ứng. Hệ điều hành thường sử dụng ngăn xếp khi gọi hàm. Các tham số, biến cục bộ và địa chỉ trả về được đẩy lên ngăn xếp khi một hàm được gọi; trở về từ lệnh gọi hàm sẽ bật các mục đó ra khỏi ngăn xếp.

Ngược lại, **hàng đợi (queue)** là một cấu trúc dữ liệu được sắp xếp theo thứ tự sử dụng nguyên tắc vào trước, ra trước **(FIFO** *vt.* **First in, First out)**: các mục được xóa khỏi hàng đợi theo thứ tự chúng được chèn vào. Có rất nhiều ví dụ hàng ngày về việc xếp hàng, bao gồm cả người mua hàng chờ xếp hàng thanh toán tại một cửa hàng và ô tô xếp hàng chờ ở tín hiệu giao thông. Hàng đợi cũng khá phổ biến trong hệ điều hành – ví dụ: các công việc được gửi đến máy in thường được in theo thứ tự mà chúng đã được gửi. Như chúng ta sẽ thấy trong Chương 5, các tác vụ đang chờ được chạy trên một CPU có sẵn thường được sắp xếp theo hàng đợi.

1.9.2 Cây

Cây (tree) là một cấu trúc dữ liệu có thể được sử dụng để biểu diễn dữ liệu theo thứ bậc. Các giá trị dữ liệu trong cấu trúc cây được liên kết thông qua các mối quan hệ cha – con. Trong một **cây tổng quát (general tree)**, bố mẹ có thể có số lượng con không hạn chế. Trong một **cây nhị phân (binary tree)**, cha mẹ có thể có nhiều nhất hai con, chúng ta gọi là ***con bên trái (left child)*** và ***con bên phải (right child)***. Ngoài ra, **cây tìm kiếm nhị phân (binary search tree)** còn yêu cầu một thứ tự giữa hai phần tử con của cha mẹ, trong đó ***con bên trái <= con bên phải***. Hình 1.20 cung cấp một ví dụ về cây tìm kiếm nhị phân. Khi chúng ta tìm kiếm một mục trong cây tìm kiếm nhị phân, hiệu suất trong trường hợp xấu nhất là O(*n*) (hãy xem xét điều này có thể xảy ra như thế nào). Để khắc phục tình trạng này, chúng ta có thể sử dụng thuật toán tạo **cây tìm kiếm nhị phân cân bằng (balanced binary search tree)**. Ở đây, một cây

chứa *n* mục có nhiều nhất *lg(n)* cấp, do đó đảm bảo hiệu suất trong trường hợp xấu nhất là O (*lg n*). Chúng ta sẽ thấy trong Phần 5.7.1 rằng Linux sử dụng cây tìm kiếm nhị phân cân bằng (được gọi là **cây đỏ-đen** *dịch* **red-back tree**) như một phần của thuật toán lập lịch trình CPU của nó.

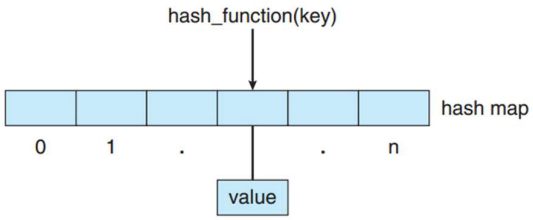


Hình 1.20 Cây nhị phân tìm kiếm.

1.9.3 Hàm băm và ánh xạ

Hàm băm (hash function) lấy dữ liệu làm đầu vào, thực hiện thao tác số trên dữ liệu và trả về giá trị số. Giá trị số này sau đó có thể được sử dụng như một chỉ mục trong một bảng (thường là một mảng) để nhanh chóng truy xuất dữ liệu. Trong khi việc tìm kiếm mục dữ liệu thông qua danh sách kích thước *n* có thể yêu cầu so sánh tới $O(n)$, thì việc sử dụng hàm băm để truy xuất dữ liệu từ bảng có thể tốt như $O(1)$, tùy thuộc vào chi tiết triển khai. Do hiệu suất này, các hàm băm được sử dụng rộng rãi trong các hệ điều hành.

Một khó khăn tiềm ẩn với các hàm băm là hai đầu vào duy nhất có thể dẫn đến cùng một giá trị đầu ra – nghĩa là chúng có thể liên kết đến cùng một vị trí bảng. Chúng ta có thể giải quyết **xung đột băm (hash collision)** này bằng cách có một danh sách được liên kết tại vị trí bảng chứa tất cả các mục có cùng giá trị băm. Tất nhiên, càng có nhiều va chạm, hàm băm càng kém hiệu quả.



Hình 1.21 Ánh xạ băm.

Một ứng dụng của hàm băm là dùng để thực hiện một **ánh xạ băm (hash map)**, ánh xạ này liên kết (hoặc ánh xạ) các cặp [key:value] bằng cách sử dụng một hàm băm. Khi ánh xạ được thiết lập, chúng ta có thể áp dụng hàm băm cho khóa để lấy giá trị từ bản đồ băm (Hình 1.21). Ví dụ: giả sử rằng một tên người dùng được ánh xạ tới một mật khẩu. Xác thực mật khẩu sau đó tiến hành như sau: một người dùng nhập tên người dùng và mật khẩu của mình. Hàm băm được áp dụng cho tên người dùng, sau đó được sử dụng để lấy mật khẩu. Sau đó, mật khẩu

được truy xuất sẽ được so sánh với mật khẩu do người dùng nhập để xác thực.

1.9.4 Bitmap

Một **bitmap** là một chuỗi gồm *n* chữ số nhị phân có thể được sử dụng để biểu diễn trạng thái của *n* mục. Ví dụ: giả sử chúng ta có một số tài nguyên và tính khả dụng của mỗi tài nguyên được biểu thị bằng giá trị của một chữ số nhị phân: 0 có nghĩa là tài nguyên có sẵn, trong khi 1 cho biết rằng nó không có sẵn (hoặc ngược lại). Giá trị của vị trí thứ *i* trong bitmap được liên kết với tài nguyên thứ *i*. Ví dụ, hãy xem xét bitmap được hiển thị bên dưới:

```
001011101
```

Tài nguyên 2, 4, 5, 6 và 8 không khả dụng; các tài nguyên 0, 1, 3 và 7 đều có sẵn.

Sức mạnh của bitmap trở nên rõ ràng khi chúng ta xem xét hiệu quả không gian của chúng. Nếu chúng ta sử dụng giá trị Boolean tám bit thay vì một bit, cấu trúc dữ liệu kết quả sẽ lớn hơn tám lần. Do đó, bitmap thường được sử dụng khi cần thể hiện sự sẵn có của một số lượng lớn tài nguyên. Ổ đĩa cung cấp một minh họa đẹp. Một ổ đĩa cỡ trung bình có thể được chia thành vài nghìn đơn vị riêng lẻ, được gọi là **khối đĩa (disk block)**. Một bitmap có thể được sử dụng để chỉ ra tính khả dụng của mỗi khối đĩa.

Tóm lại, cấu trúc dữ liệu có tính phổ biến trong việc triển khai hệ điều hành. Do đó, chúng ta sẽ thấy các cấu trúc được thảo luận ở đây, cùng với các cấu trúc khác, xuyên suốt quyển sách này khi chúng ta khám phá các thuật toán của nhân và cách triển khai của chúng.

CÁC CẤU TRÚC DỮ LIỆU NHÂN LINUX

Các cấu trúc dữ liệu được sử dụng trong nhân Linux có sẵn trong mã nguồn của nhân. Tập tin bao gồm `<linux/list.h>` cung cấp chi tiết về cấu trúc dữ liệu danh sách liên kết được sử dụng trong nhân. Một hàng đợi trong Linux được gọi là `kfifo` và việc triển khai nó có thể được tìm thấy trong tập tin `kfifo.c` trong thư mục `kernel` của mã nguồn. Linux alsop cung cấp triển khai cây tìm kiếm nhị phân cân bằng bằng cách sử dụng cây đỏ-đen. Thông tin chi tiết có thể được tìm thấy trong tập tin bao gồm `<linux/rbtree.h>`.

1.10 Các môi trường điện toán

Cho đến nay, chúng tôi đã mô tả ngắn gọn một số khía cạnh của hệ thống máy tính và hệ điều hành quản lý chúng. Bây giờ chúng ta chuyển sang một cuộc thảo luận về cách hệ điều hành được sử dụng trong nhiều môi trường máy tính khác nhau.

1.10.1 Điện toán truyền thống

Khi máy tính đã trưởng thành, các ranh giới ngăn cách nhiều môi trường máy tính truyền thống đã bị mờ đi. Hãy xem xét "*môi trường văn phòng điển hình*". Chỉ vài năm trước, môi trường này bao gồm các PC được kết nối với mạng, với các máy chủ cung cấp dịch vụ tập tin và in. Việc truy cập từ xa rất khó khăn và tính di động đạt được khi sử dụng máy tính xách tay.

Ngày nay, công nghệ web và băng thông WAN ngày càng tăng đang kéo dài ranh giới của điện toán truyền thống. Các công ty thiết lập các **cổng (portal)** cung cấp khả năng truy cập web vào các máy chủ nội bộ của họ. **Máy tính mạng (network computer)** (hoặc **máy khách mỏng** *dịch* **thin client**) – mà về cơ bản là các thiết bị đầu cuối hiểu được tính toán trên nền tảng web – được sử dụng thay cho các máy trạm truyền thống nơi mong muốn bảo mật cao hơn hoặc bảo trì dễ dàng hơn. Máy tính di động có thể đồng bộ hóa với PC để cho phép sử dụng thông tin công ty rất linh động. Thiết bị di động cũng có thể kết nối với **mạng không dây (wireless network)** và mạng dữ liệu di động để sử dụng cổng web của công ty (cũng như vô số tài nguyên web khác).

Ở nhà, hầu hết người dùng đều từng có một máy tính có kết nối modem chậm với văn phòng, Internet hoặc cả hai. Ngày nay, tốc độ kết nối mạng đã có sẵn chỉ với chi phí tương đối rẻ ở nhiều nơi, cho phép người dùng gia đình tiếp cận nhiều dữ liệu hơn. Các kết nối dữ liệu nhanh này cho phép các máy tính gia đình phục vụ các trang web và chạy các mạng bao gồm máy in, máy tính khách và máy chủ. Nhiều ngôi nhà sử dụng **tường lửa (firewall)** để bảo vệ mạng của họ khỏi các vi phạm bảo mật. Tường lửa giới hạn giao tiếp giữa các thiết bị trong mạng.

Trong nửa sau của thế kỷ 20, tài nguyên máy tính tương đối khan hiếm. (Trước đó, chúng không tồn tại!) Trong một khoảng thời gian, các hệ thống có thể là hàng loạt hoặc tương tác. Hệ thống hàng loạt xử lý hàng loạt công việc với đầu vào được xác định trước từ các tập tin hoặc các nguồn dữ liệu khác. Hệ thống tương tác đã chờ đợi thông tin đầu vào từ người dùng. Để tối ưu hóa việc sử dụng tài nguyên máy tính, nhiều người dùng đã chia sẻ thời gian trên các hệ thống này. Các hệ thống chia sẻ thời gian này đã sử dụng một bộ đếm thời gian và các thuật toán lập lịch để quay vòng các quá trình nhanh chóng thông qua CPU, cung cấp cho mỗi người dùng một phần tài nguyên.

Ngày nay rất hiếm các hệ thống chia sẻ thời gian truyền thống. Kỹ thuật lập lịch tương tự vẫn được sử dụng trên máy tính để bàn, máy tính xách tay, máy chủ và thậm chí cả máy tính di động, nhưng thường thì tất cả các tiến trình đều thuộc sở hữu của cùng một người dùng (hoặc một người dùng và hệ điều hành). Các tiến trình của người dùng và tiến trình hệ thống cung cấp dịch vụ cho người dùng, được quản lý để mỗi quy trình thường có một phần thời gian sử dụng máy tính. Ví dụ: hãy xem xét các cửa sổ được tạo trong khi người dùng đang làm việc trên PC và thực tế là họ có thể đang thực hiện các tác vụ khác nhau cùng một lúc. Ngay cả một trình duyệt web cũng có thể bao gồm nhiều tiến trình, một tiến trình cho mỗi trang web hiện đang được truy cập, với chia sẻ thời gian được áp dụng cho từng tiến trình trình duyệt web.

1.10.2 Điện toán di động

Điện toán di động (mobile computing) đề cập đến tính toán trên điện thoại thông minh cầm tay và máy tính bảng. Các thiết bị này có chung các đặc điểm vật lý khác nhau là di động và nhẹ. Về mặt lịch sử, so với máy tính để bàn và máy tính xách tay, các hệ thống di động đã từ bỏ kích thước màn hình, dung lượng bộ nhớ và chức năng tổng thể để đổi lại quyền truy cập di động cầm tay vào các dịch vụ như e-mail và duyệt web. Tuy nhiên, trong vài năm qua, các tính năng trên thiết bị di động đã trở nên phong phú đến mức khó phân biệt về chức năng giữa máy tính xách tay tiêu dùng và máy tính bảng. Trên thực tế, chúng tôi có thể tranh luận rằng

các tính năng của một thiết bị di động hiện đại cho phép nó cung cấp chức năng không khả dụng hoặc không thực tế trên máy tính để bàn hoặc máy tính xách tay.

Ngày nay, các hệ thống di động không chỉ được sử dụng cho e-mail và duyệt web mà còn để chơi nhạc và video, đọc sách kỹ thuật số, chụp ảnh, ghi và chỉnh sửa video độ nét cao. Theo đó, tốc độ tăng trưởng vượt bậc tiếp tục diễn ra trong một loạt các ứng dụng chạy trên các thiết bị như vậy. Nhiều nhà phát triển hiện đang thiết kế các ứng dụng tận dụng các tính năng độc đáo của thiết bị di động, chẳng hạn như chip hệ thống định vị toàn cầu (GPS), máy đo gia tốc và con quay hồi chuyển. Một chip GPS được nhúng cho phép thiết bị di động sử dụng vệ tinh để xác định vị trí chính xác của nó trên Trái đất. Chức năng đó đặc biệt hữu ích trong việc thiết kế các ứng dụng cung cấp điều hướng – ví dụ: cho người dùng biết cách đi bộ hoặc lái xe hoặc có thể hướng họ đến các dịch vụ lân cận, chẳng hạn như nhà hàng. Gia tốc kế cho phép thiết bị di động phát hiện hướng của nó so với mặt đất và phát hiện một số lực khác, chẳng hạn như nghiêng và lắc. Trong một số trò chơi máy tính sử dụng gia tốc kế, người chơi giao tiếp với hệ thống không phải bằng chuột hoặc bàn phím mà bằng cách nghiêng, xoay và lắc thiết bị di động! Có lẽ nhiều ứng dụng thực tế hơn của những tính năng này được tìm thấy trong các ứng dụng thực tế tăng cường, bao phủ thông tin trên màn hình của môi trường hiện tại. Thật khó để tưởng tượng làm thế nào các ứng dụng tương đương có thể được phát triển trên các hệ thống máy tính xách tay hoặc máy tính để bàn truyền thống.

Để cung cấp quyền truy cập vào các dịch vụ trực tuyến, các thiết bị di động thường sử dụng mạng dữ liệu di động hoặc không dây chuẩn IEEE 802.11. Tuy nhiên, dung lượng bộ nhớ và tốc độ xử lý của thiết bị di động bị hạn chế hơn so với PC. Trong khi điện thoại thông minh hoặc máy tính bảng có thể có 256 GB dung lượng lưu trữ, thì việc tìm thấy 8 TB dung lượng lưu trữ trên máy tính để bàn không phải là hiếm. Tương tự, vì mức tiêu thụ điện năng là một vấn đề đáng lo ngại, các thiết bị di động thường sử dụng bộ vi xử lý nhỏ hơn, chậm hơn và cung cấp ít lõi xử lý hơn so với bộ xử lý trên máy tính để bàn và máy tính xách tay truyền thống.

Hai hệ điều hành hiện đang thống trị điện toán di động: **Apple iOS** và **Google Android**. iOS được thiết kế để chạy trên các thiết bị di động Apple iPhone và iPad. Android hỗ trợ điện thoại thông minh và máy tính bảng có sẵn từ nhiều nhà sản xuất. Chúng ta xem xét chi tiết hơn hai hệ điều hành di động này trong Chương 2.

1.10.3 Điện toán Máy khách - Máy chủ

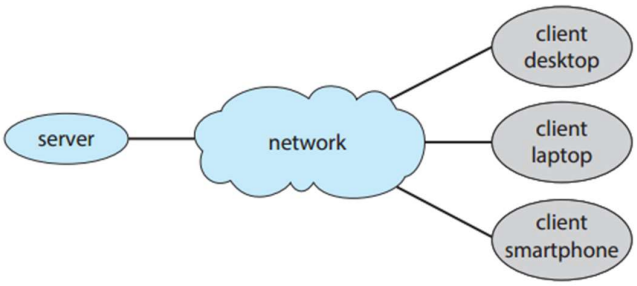
Kiến trúc mạng đương đại có tính năng sắp xếp các **hệ thống máy chủ (server system)** đáp ứng các yêu cầu do **hệ thống máy khách (client system)** tạo ra. Dạng hệ thống phân tán chuyên biệt này, được gọi là hệ thống **máy khách-máy chủ (client-server)**, có cấu trúc chung được mô tả trong Hình 1.22.

Hệ thống máy chủ có thể được phân loại rộng rãi là máy chủ máy tính và máy chủ tập tin:

- **Hệ thống máy chủ-máy tính (computer-server system)** cung cấp một giao diện mà máy khách có thể gửi yêu cầu thực hiện một hành động (ví dụ: đọc dữ liệu). Đáp lại, máy chủ thực hiện hành động và gửi kết quả đến máy khách. Một máy chủ chạy

cơ sở dữ liệu đáp ứng các yêu cầu dữ liệu của khách hàng là một ví dụ về hệ thống như vậy.

- **Hệ thống phục vụ tập tin (file-serve system)** cung cấp giao diện hệ thống tập tin nơi máy khách có thể tạo, cập nhật, đọc và xóa tập tin. Ví dụ về một hệ thống như vậy là một máy chủ web cung cấp các tập tin đến các máy khách chạy trình duyệt web. Nội dung thực tế của các tập tin có thể khác nhau rất nhiều, từ các trang web truyền thống đến nội dung đa phương tiện phong phú như video độ phân giải cao.



Hình 1. 22 Mô hình tổng quan Máy khách - Máy chủ.

1.10.4 Điện toán ngang hàng

Một cấu trúc khác cho hệ thống phân tán là mô hình **hệ thống ngang hàng (P2P** *vt.* **Peer-to-peer)**. Trong mô hình này, máy khách và máy chủ không được phân biệt với nhau. Thay vào đó, tất cả các nút trong hệ thống được coi là đồng cấp và mỗi nút có thể hoạt động như một máy khách hoặc một máy chủ, tùy thuộc vào việc nó đang yêu cầu hay cung cấp dịch vụ. Hệ thống ngang hàng mang lại lợi thế hơn so với hệ thống máy khách-máy chủ truyền thống. Trong hệ thống máy khách-máy chủ, máy chủ là một nút cổ chai; nhưng trong hệ thống ngang hàng, các dịch vụ có thể được cung cấp bởi một số nút được phân phối trên toàn mạng.

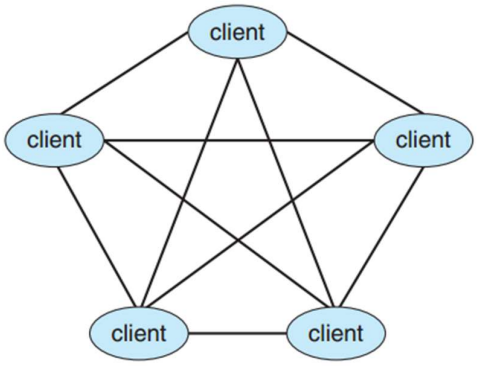
Để tham gia vào một hệ thống ngang hàng, một nút trước tiên phải tham gia vào mạng lưới các đồng đẳng. Khi một nút đã tham gia vào mạng, nó có thể bắt đầu cung cấp dịch vụ cho – và yêu cầu dịch vụ từ – các nút khác trong mạng. Việc xác định những dịch vụ nào có sẵn được thực hiện theo một trong hai cách chung:

- Khi một nút tham gia vào mạng, nó sẽ đăng ký dịch vụ của mình với một dịch vụ tra cứu tập trung trên mạng. Bất kỳ nút nào mong muốn một dịch vụ cụ thể trước tiên sẽ liên hệ với dịch vụ tra cứu tập trung này để xác định nút nào cung cấp dịch vụ. Phần còn lại của giao tiếp diễn ra giữa khách hàng và nhà cung cấp dịch vụ.

- Một chương trình thay thế không sử dụng dịch vụ tra cứu tập trung. Thay vào đó, một ứng dụng ngang hàng đóng vai trò là một máy khách phải khám phá xem nút nào cung cấp dịch vụ mong muốn bằng cách phát một yêu cầu dịch vụ tới tất cả các nút khác trong mạng. Nút (hoặc các nút) cung cấp dịch vụ đó sẽ phản hồi lại người ngang hàng đưa ra yêu cầu. Để hỗ trợ cách tiếp cận này, một ***giao thức khám phá (discovery protocol)*** phải được cung cấp cho phép các nút khám phá các dịch vụ

được cung cấp bởi các nút khác trong mạng. Hình 1.23 minh họa một kịch bản như vậy.

Mạng ngang hàng đã trở nên phổ biến rộng rãi vào cuối những năm 1990 với một số dịch vụ chia sẻ tập tin, chẳng hạn như Napster và Gnutella, cho phép các mạng ngang hàng trao đổi tập tin với nhau. Hệ thống Napster sử dụng cách tiếp cận tương tự như kiểu đầu tiên được mô tả ở trên: một máy chủ tập trung duy trì một chỉ mục của tất cả các tập tin được lưu trữ trên các nút ngang hàng trong mạng Napster và việc trao đổi tập tin thực sự diễn ra giữa các nút ngang hàng. Hệ thống Gnutella đã sử dụng một kỹ thuật tương tự như kiểu thứ hai: một tập tin tin quảng bá máy khách yêu cầu đến các nút khác trong hệ thống và các nút có thể phục vụ yêu cầu đó sẽ phản hồi trực tiếp cho máy khách. Mạng ngang hàng có thể được sử dụng để trao đổi ẩn danh tài liệu có bản quyền (ví dụ: âm nhạc) và có các luật điều chỉnh việc phân phối tài liệu có bản quyền. Đáng chú ý, Napster gặp rắc rối pháp lý vì vi phạm bản quyền và các dịch vụ của nó đã ngừng hoạt động vào năm 2001. Vì lý do này, tương lai của việc trao đổi tập tin vẫn chưa chắc chắn.



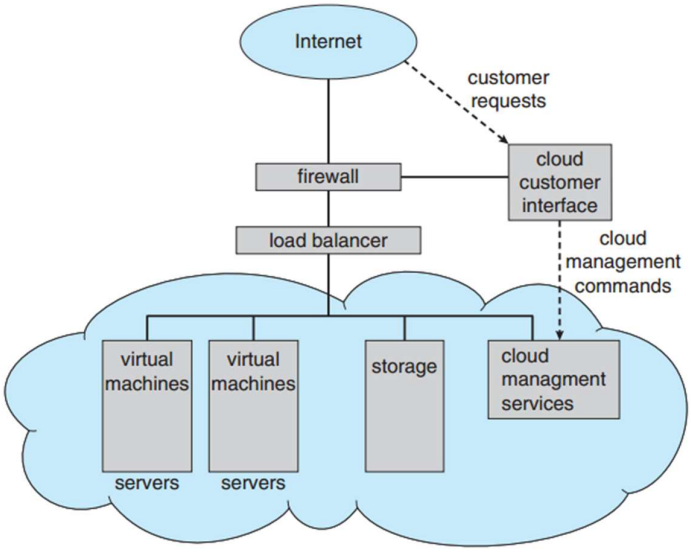
Hình 1.23 Hệ thống P2P không có dịch vụ trung tâm.

Skype là một ví dụ khác về tính toán ngang hàng. Nó cho phép khách hàng thực hiện cuộc gọi thoại và cuộc gọi video và gửi tin nhắn văn bản qua Internet bằng công nghệ được gọi là **thoại qua IP (VoIP** *vt.* **Voice over IP)**. Skype sử dụng cách tiếp cận ngang hàng kết hợp. Nó bao gồm một máy chủ đăng nhập tập trung, nhưng nó cũng kết hợp các nút phi tập trung và cho phép hai nút giao tiếp.

1.10.5 Điện toán đám mây

Điện toán đám mây (Cloud Computing) là một loại máy tính cung cấp khả năng tính toán, lưu trữ và thậm chí là các ứng dụng như một dịch vụ trên một mạng. Theo một số cách, nó là một phần mở rộng hợp lý của ảo hóa, bởi vì nó sử dụng ảo hóa làm cơ sở cho chức năng của nó. Ví dụ: cơ sở Amazon Elastic Compute Cloud (**ec2**) có hàng nghìn máy chủ, hàng triệu máy ảo và petabyte dung lượng lưu trữ có sẵn để mọi người trên Internet sử dụng. Người dùng trả tiền mỗi tháng dựa trên số lượng tài nguyên mà họ sử dụng. Thực tế có nhiều loại điện toán đám mây, bao gồm các loại sau:

- **Đám mây công cộng (Public cloud)** – một đám mây có sẵn qua Internet cho bất kỳ ai sẵn sàng trả tiền cho các dịch vụ
- **Đám mây riêng tư (Private cloud)** – đám mây do một công ty điều hành để sử dụng riêng cho công ty đó
- **Đám mây lai (Hybrid cloud)** – đám mây bao gồm cả các thành phần đám mây công cộng và riêng tư
- **Phần mềm như một dịch vụ (SaaS** *vt.* **Software as a service)** –một hoặc nhiều ứng dụng (chẳng hạn như bộ xử lý văn bản hoặc bảng tính) khả dụng qua Internet
- **Nền tảng như một dịch vụ (PaaS** *vt.* **Platform as a service)** –một ngăn xếp phần mềm sẵn sàng cho ứng dụng sử dụng qua Internet (ví dụ: máy chủ cơ sở dữ liệu)
- **Cơ sở hạ tầng như một dịch vụ (IaaS** *vt.* **Infrastructure as a service)** – máy chủ hoặc bộ lưu trữ có sẵn trên Internet (ví dụ: bộ nhớ có sẵn để tạo bản sao dự phòng của dữ liệu sản xuất)



Hình 1.24 Điện toán đám mây.

Các loại điện toán đám mây này không rời rạc , vì môi trường điện toán đám mây có thể cung cấp sự kết hợp của một số loại. Ví dụ: một tổ chức có thể cung cấp cả SaaS và IaaS dưới dạng các dịch vụ công khai.

Chắc chắn, có những hệ điều hành truyền thống trong nhiều loại cơ sở hạ tầng đám mây. Ngoài những thứ đó là các VMM quản lý các máy ảo mà các quy trình người dùng chạy. Ở cấp độ cao hơn, bản thân các VMM được quản lý bởi các công cụ quản lý đám mây, chẳng hạn như VMware vCloud Director và bộ công cụ Eucalyptus mã nguồn mở. Các công cụ này quản lý các tài nguyên trong một đám mây nhất định và cung cấp giao diện cho các thành phần đám mây, tạo ra một lý lẽ tốt để coi chúng là một loại hệ điều hành mới. Hình 1.24 minh họa một đám mây công cộng cung cấp IaaS. Lưu ý rằng cả dịch vụ đám mây và giao diện người dùng đám mây đều được bảo vệ bởi tường lửa.

1.10.6 Hệ thống nhúng thời gian thực

Máy tính nhúng là dạng máy tính phổ biến nhất đang tồn tại. Những thiết bị này được tìm thấy ở khắp mọi nơi, từ động cơ xe hơi và robot sản xuất đến ổ đĩa quang và lò vi sóng. Họ có xu hướng có những nhiệm vụ rất cụ thể. Các hệ thống mà chúng chạy thường là nguyên thủy, và do đó các hệ điều hành cung cấp các tính năng hạn chế. Thông thường, họ có ít hoặc không có giao diện người dùng, thích dành thời gian theo dõi và quản lý các thiết bị phần cứng, chẳng hạn như động cơ ô tô và cánh tay robot.

Các hệ thống nhúng này khác nhau đáng kể. Một số là máy tính có mục đích chung, chạy hệ điều hành tiêu chuẩn – chẳng hạn như Linux – với các ứng dụng dành cho mục đích đặc biệt để triển khai chức năng. Những thiết bị khác là các thiết bị phần cứng với hệ điều hành nhúng có mục đích đặc biệt chỉ cung cấp chức năng mong muốn. Tuy nhiên, những thiết bị khác là thiết bị phần cứng có **mạch tích hợp dành riêng cho ứng dụng (ASIC *vt.* application-specific integrated circuits)** thực hiện các tác vụ của chúng mà không cần hệ điều hành.

Việc sử dụng các hệ thống nhúng tiếp tục được mở rộng. Sức mạnh của các thiết bị này, cả với tư cách là các thiết bị độc lập và các yếu tố của mạng và web, chắc chắn sẽ tăng lên. Ngay cả bây giờ, toàn bộ ngôi nhà có thể được vi tính hóa, để một máy tính trung tâm – hoặc một máy tính đa năng hoặc một hệ thống nhúng – có thể điều khiển hệ thống sưởi và ánh sáng, hệ thống báo động, và thậm chí cả máy pha cà phê. Truy cập web có thể cho phép chủ nhân ngôi nhà thông báo ngôi nhà nóng lên trước khi cô ấy về nhà. Một ngày nào đó, tủ lạnh sẽ có thể thông báo cho cửa hàng tạp hóa khi họ thông báo sữa đã hết.

Hệ thống nhúng hầu như luôn chạy **hệ điều hành thời gian thực (real-time operating systems)**. Hệ thống thời gian thực được sử dụng khi các yêu cầu về thời gian nghiêm ngặt được đặt ra đối với hoạt động của bộ xử lý hoặc luồng dữ liệu; do đó, nó thường được sử dụng như một thiết bị điều khiển trong một ứng dụng chuyên dụng. Cảm biến đưa dữ liệu đến máy tính. Máy tính phải phân tích dữ liệu và có thể điều chỉnh các điều khiển để sửa đổi các đầu vào cảm biến. Hệ thống điều khiển các thí nghiệm khoa học, hệ thống hình ảnh y tế, hệ thống điều khiển công nghiệp và một số hệ thống hiển thị nhất định là hệ thống thời gian thực. Một số hệ thống phun nhiên liệu cho ô tô – động cơ, bộ điều khiển thiết bị gia dụng và hệ thống vũ khí cũng là hệ thống thời gian thực.

Hệ thống thời gian thực có các giới hạn thời gian cố định, được xác định rõ ràng. Tiến trình xử lý ***phải*** được thực hiện trong các ràng buộc đã xác định, nếu không hệ thống sẽ thất bại. Ví dụ, sẽ không có hiệu quả khi một cánh tay robot được hướng dẫn dừng lại ***sau khi*** nó đã đập vào chiếc xe mà nó đang chế tạo. Hệ thống thời gian thực chỉ hoạt động chính xác nếu nó trả về kết quả đúng trong giới hạn thời gian của nó. Tương phản hệ thống này với hệ thống máy tính xách tay truyền thống, nơi nó được mong muốn (nhưng không bắt buộc) phải phản hồi nhanh chóng.

Trong Chương 5, chúng tôi xem xét cơ sở lập lịch cần thiết để triển khai chức năng thời gian thực trong hệ điều hành và trong Chương 20, chúng tôi mô tả các thành phần thời gian thực của Linux.

1.11 Hệ điều hành miễn phí và mã nguồn mở

Việc nghiên cứu các hệ điều hành đã trở nên dễ dàng hơn nhờ sự sẵn có của một số lượng lớn các phần mềm miễn phí và các bản phát hành mã nguồn mở. Cả hai **hệ điều hành miễn phí (Free Operating Systems)** và **hệ điều hành mã nguồn mở (Open–Source Operating Systems)** đều có sẵn ở định dạng mã nguồn chứ không phải là mã nhị phân được biên dịch. Tuy nhiên, lưu ý rằng phần mềm miễn phí và phần mềm nguồn mở là hai ý tưởng khác nhau được các nhóm người khác nhau ủng hộ (xem *http://gnu.org/phiosystemhy/open–source–misses–the–point.html/* để thảo luận về chủ đề). Phần mềm miễn phí (đôi khi được gọi là phần mềm tự do ***free/libre sofware***) không chỉ cung cấp mã nguồn mà còn được cấp phép để cho phép sử dụng, phân phối lại và sửa đổi miễn phí. Phần mềm nguồn mở không nhất thiết phải cung cấp giấy phép như vậy. Do đó, mặc dù tất cả phần mềm miễn phí đều là mã nguồn mở, nhưng một số phần mềm mã nguồn mở không phải là "miễn phí". GNU/Linux là hệ điều hành mã nguồn mở nổi tiếng nhất, với một số bản phân phối miễn phí và những bản khác chỉ có mã nguồn mở (*http://www.gnu.org/distros/*). Microsoft Windows là một ví dụ nổi tiếng về cách tiếp cận ngược lại là **mã nguồn đóng (closed-source)**. Windows là phần mềm **độc quyền (proprietary)** – Microsoft sở hữu nó, hạn chế việc sử dụng nó và bảo vệ cẩn thận mã nguồn của nó. Hệ điều hành macOS của Apple bao gồm một phương pháp kết hợp. Nó chứa một nhân mã nguồn mở tên là Darwin nhưng cũng bao gồm các thành phần nguồn đóng, độc quyền.

Bắt đầu với mã nguồn cho phép lập trình viên tạo ra mã nhị phân có thể được thực thi trên một hệ thống. Làm ngược lại – **dịch ngược mã nguồn (reverse engineering)** từ các mã nhị phân – là công việc khá nhiều và các mục hữu ích như ghi chú không bao giờ được phục hồi. Học hệ điều hành bằng cách kiểm tra mã nguồn cũng có những lợi ích khác. Với mã nguồn trong tay, sinh viên có thể sửa đổi hệ điều hành, sau đó biên dịch và chạy mã để thử những thay đổi đó, đây là một công cụ học tập tuyệt vời. Sách này bao gồm các dự án liên quan đến việc sửa đổi mã nguồn hệ điều hành, đồng thời mô tả các thuật toán ở cấp độ cao để đảm bảo tất cả các chủ đề quan trọng của hệ điều hành đều được đề cập. Xuyên suốt sách, chúng tôi cung cấp các gợi ý đến các ví dụ về mã nguồn mở để nghiên cứu sâu hơn.

Có nhiều lợi ích đối với hệ điều hành mã nguồn mở, bao gồm một cộng đồng các lập trình viên quan tâm (và thường không được trả tiền), những người đóng góp vào mã bằng cách giúp viết mã, gỡ lỗi, phân tích, cung cấp hỗ trợ và đề xuất các thay đổi. Có thể cho rằng, mã nguồn mở an toàn hơn mã nguồn đóng vì có nhiều con mắt đang xem mã. Chắc chắn, mã nguồn mở có lỗi, nhưng những người ủng hộ mã nguồn mở cho rằng lỗi có xu hướng được tìm thấy và sửa nhanh hơn do số lượng người sử dụng và xem mã. Các công ty kiếm được doanh thu từ việc bán các chương trình của họ thường ngại mở mã nguồn của họ, nhưng Red Hat và vô số các công ty khác đang làm điều đó và cho thấy rằng các công ty thương mại được lợi hơn là bị thiệt hại khi họ mở nguồn mã của họ. Ví dụ: doanh thu có thể được tạo ra thông qua các hợp đồng hỗ trợ và việc bán phần cứng mà phần mềm chạy trên đó.

1.11.1 Lịch sử

Trong những ngày đầu của máy tính hiện đại (tức là những năm 1950), phần mềm thường đi kèm với mã nguồn. Các tin tặc ban đầu (những người đam mê máy tính) tại MIT's Tech Model Railroad Club đã để các chương trình của họ trong ngăn kéo để những người khác

làm việc. Các nhóm người dùng "Homebrew" đã trao đổi mã trong cuộc họp của họ. Các nhóm người dùng cụ thể của công ty, chẳng hạn như DECUS của Công ty Cổ phần Thiết bị Kỹ thuật số, đã chấp nhận đóng góp của các chương trình mã nguồn, thu thập chúng vào băng và phân phối băng cho các thành viên quan tâm. Năm 1970, hệ điều hành của Digital được phân phối dưới dạng mã nguồn mà không có hạn chế hoặc thông báo bản quyền.

Các công ty máy tính và phần mềm cuối cùng đã tìm cách hạn chế việc sử dụng phần mềm của họ đối với các máy tính được ủy quyền và khách hàng trả tiền. Việc chỉ phát hành các tập tin nhị phân được biên dịch từ mã nguồn thay vì chính mã nguồn đã giúp họ đạt được mục tiêu này, cũng như bảo vệ mã và ý tưởng của họ khỏi các đối thủ cạnh tranh. Mặc dù các nhóm người dùng Homebrew của những năm 1970 đã trao đổi mã trong các cuộc họp của họ, hệ điều hành dành cho các máy có sở thích (chẳng hạn như CPM) là độc quyền. Đến năm 1980, phần mềm độc quyền là trường hợp thông thường.

1.11.2 Free Operating Systems

Để chống lại động thái hạn chế việc sử dụng và phân phối lại phần mềm, Richard Stallman vào năm 1984 bắt đầu phát triển một hệ điều hành miễn phí, tương thích với UNIX được gọi là GNU (là từ viết tắt đệ quy của "GNU's Not Unix!"). Đối với Stallman, "miễn phí" đề cập đến quyền tự do sử dụng, không phải giá cả. Phong trào phần mềm tự do không phản đối việc giao dịch một bản sao để lấy một số tiền nhưng cho rằng người dùng được hưởng bốn quyền tự do nhất định: (1) tự do chạy chương trình, (2) nghiên cứu và thay đổi mã nguồn, và cung cấp hoặc bán các bản sao (3) có hoặc (4) mà không có thay đổi. Năm 1985, Stallman xuất bản Tuyên ngôn GNU, lập luận rằng tất cả phần mềm phải miễn phí. Ông cũng thành lập **Quỹ Phần mềm Tự do (FSF *vt.* Free Software Foundation)** với mục tiêu khuyến khích việc sử dụng và phát triển phần mềm miễn phí.

FSF sử dụng bản quyền trên các chương trình của mình để triển khai "copyleft[6]", một hình thức cấp phép do Stallman phát minh. Sao chép tác phẩm mang lại cho bất kỳ ai sở hữu bản sao tác phẩm bốn quyền tự do thiết yếu giúp tác phẩm trở nên miễn phí, với điều kiện việc phân phối lại phải bảo toàn các quyền tự do này. **Giấy phép Công cộng** GNU **(GPL *vt.* General Public License)** là một giấy phép phổ biến theo đó phần mềm tự do được phát hành. Về cơ bản, GPL yêu cầu mã nguồn được phân phối với bất kỳ tập tin nhị phân nào và tất cả các bản sao (bao gồm cả các phiên bản đã sửa đổi) phải được phát hành theo cùng một giấy phép GPL. Giấy phép Creative Commons "Attribution Sharealike" cũng là giấy phép copyleft; "Sharealike" là một cách khác để nói rõ ý tưởng về copyleft.

1.11.3 GNU/Linux

Để làm ví dụ về hệ điều hành mã nguồn mở và miễn phí, hãy xem xét **GNU/Linux**. Đến năm 1991, hệ điều hành GNU gần như hoàn thiện. Dự án GNU đã phát triển trình biên dịch, trình chỉnh sửa, tiện ích, thư viện và trò chơi – bất kỳ phần nào nó không thể tìm thấy ở nơi khác. Tuy nhiên, nhân GNU chưa bao giờ sẵn sàng cho thời gian đầu. Năm 1991, một sinh viên ở Phần Lan, Linus Torvalds, đã phát hành một hạt nhân giống UNIX thô sơ bằng cách sử dụng các công cụ và trình biên dịch GNU và mời đóng góp trên toàn thế giới. Sự ra đời của

[6] Copyleft là một cách chơi chữ, xem thêm tại https://vi.wikipedia.org/wiki/Copyleft

Internet có nghĩa là bất kỳ ai quan tâm đều có thể tải xuống mã nguồn, sửa đổi nó và gửi các thay đổi tới Torvalds. Việc phát hành các bản cập nhật mỗi tuần một lần cho phép cái gọi là hệ điều hành "Linux" này phát triển nhanh chóng, được nâng cao bởi hàng nghìn lập trình viên. Năm 1991, Linux không phải là phần mềm miễn phí, vì giấy phép của nó chỉ cho phép phân phối lại phi thương mại. Tuy nhiên, vào năm 1992, Torvalds đã phát hành lại Linux theo GPL, biến nó thành phần mềm miễn phí (và cũng có thể sử dụng một thuật ngữ được đặt ra sau này, "mã nguồn mở").

Kết quả là hệ điều hành GNU/Linux (với nhân được gọi đúng là Linux nhưng hệ điều hành đầy đủ bao gồm các công cụ GNU được gọi là GNU/Linux) đã tạo ra hàng trăm **bản phân phối (distribution)** độc nhất, hoặc các bản dựng tùy chỉnh, của hệ thống. Các bản phân phối chính bao gồm Red Hat, SUSE, Fedora, Debian, Slackware và Ubuntu. Các bản phân phối khác nhau về chức năng, tiện ích, ứng dụng đã cài đặt, hỗ trợ phần cứng, giao diện người dùng và mục đích. Ví dụ, Red Hat Enterprise Linux hướng đến việc sử dụng thương mại lớn. PCLinuxOS là một **CD trực tiếp (live CD)** – hệ điều hành có thể được khởi động và chạy từ CD – ROM mà không cần cài đặt trên đĩa khởi động của hệ thống. Một biến thể của PCLinuxOS – được gọi là PCLinuxOS Supergamer DVD – là một **DVD trực tiếp (live DVD)** bao gồm trình điều khiển đồ họa và trò chơi. Một game thủ có thể chạy nó trên bất kỳ hệ thống tương thích nào chỉ bằng cách khởi động từ đĩa DVD. Khi người chơi kết thúc, quá trình khởi động lại hệ thống sẽ đặt lại nó về hệ điều hành đã cài đặt.

Bạn có thể chạy Linux trên hệ thống Windows (hoặc hệ thống khác) bằng cách sử dụng phương pháp đơn giản, miễn phí sau:

1. Tải xuống công cụ Virtualbox VMM miễn phí từ

   `https://www.virtualbox.org/`

   và cài đặt nó trên hệ thống của bạn.

2. Chọn cài đặt hệ điều hành từ đầu, dựa trên tập tin ảnh cài đặt như đĩa CD hoặc chọn ảnh hệ điều hành được tạo sẵn có thể được cài đặt và chạy nhanh hơn từ một trang web như

   `http://virtualboxes.org/images /`

Những tập tin ảnh này được cài đặt sẵn hệ điều hành và ứng dụng và bao gồm nhiều "hương vị" của GNU/Linux.

3. Khởi động máy ảo trong Virtualbox.

Một giải pháp thay thế cho việc sử dụng Virtualbox là sử dụng chương trình miễn phí Qemu (*http://wiki.qemu.org/Download/*), bao gồm lệnh `qemu-img` để chuyển đổi tập tin ảnh Virtualbox thành tập tin ảnh Qemu để dễ dàng nhập chúng.

Với sách này này, chúng tôi cung cấp một hình ảnh máy ảo của GNU/Linux đang chạy bản phát hành Ubuntu. Tập tin ảnh này chứa mã nguồn GNU/Linux cũng như các công cụ để phát triển phần mềm. Chúng tôi đề cập đến các ví dụ liên quan đến hình ảnh GNU/Linux trong toàn bộ sách này, cũng như trong một nghiên cứu điển hình chi tiết trong Chương 20.

1.11.4 BSD UNIX

BSD UNIX có lịch sử lâu đời và phức tạp hơn Linux. Nó bắt đầu vào năm 1978 như một dẫn xuất của AT&T's UNIX. Các bản phát hành của Đại học California tại Berkeley (UCB) có dạng nguồn và dạng nhị phân, nhưng chúng không phải là nguồn mở vì cần phải có giấy phép từ AT&T. Sự phát triển của BSD UNIX bị chậm lại bởi một vụ kiện của AT&T, nhưng cuối cùng một phiên bản mã nguồn mở, đầy đủ chức năng, 4.4BSD-lite, đã được phát hành vào năm 1994.

Cũng như với Linux, có nhiều bản phân phối của BSD UNIX, bao gồm FreeBSD, NetBSD, OpenBSD và DragonflyBSD. Để khám phá mã nguồn của FreeBSD, chỉ cần tải xuống hình ảnh máy ảo của phiên bản quan tâm và khởi động nó trong Virtualbox, như được mô tả ở trên cho Linux. Mã nguồn đi kèm với bản phân phối và được lưu trữ trong `/usr/src/`. Mã nguồn hạt nhân là `/usr/src/sys`. Ví dụ, để kiểm tra mã triển khai bộ nhớ ảo trong hạt nhân FreeBSD, hãy xem các tập tin trong `/usr/src/sys/vm`. Ngoài ra, bạn có thể chỉ cần xem mã nguồn trực tuyến tại *https://svnweb.freebsd.org*.

Như với nhiều dự án mã nguồn mở, mã nguồn này được chứa trong và kiểm soát bởi **hệ thống kiểm soát phiên bản (version control system)** – trong trường hợp này là "subversion" (*https://subversion.apache.org/source–code*). Hệ thống kiểm soát phiên bản cho phép người dùng "kéo" toàn bộ cây mã nguồn vào máy tính của mình và "đẩy" bất kỳ thay đổi nào trở lại kho lưu trữ để những người khác kéo. Các hệ thống này cũng cung cấp các tính năng khác, bao gồm toàn bộ lịch sử của từng tập tin và tính năng giải quyết xung đột trong trường hợp cùng một tập tin được thay đổi đồng thời. Hệ thống điều khiển phiên bản khác là **git**, được sử dụng cho GNU/Linux, cũng như các chương trình khác (*http://www.git–scm.com*).

Darwin, thành phần hạt nhân cốt lõi của macOS, dựa trên BSD UNIX và cũng có nguồn mở. Mã nguồn đó có sẵn từ *http://www.opensource.apple.com/*. Mỗi bản phát hành macOS đều có các thành phần mã nguồn mở được đăng trên trang web đó. Tên của gói chứa hạt nhân bắt đầu bằng "xnu". Apple cũng cung cấp các công cụ, tài liệu và hỗ trợ dành cho nhà phát triển tại *http://developer.apple.com*.

1.11.5 Solaris

Solaris là hệ điều hành thương mại dựa trên UNIX của Sun Microsystems. Ban đầu, hệ điều hành **SunOS** của Sun dựa trên BSD UNIX. Sun chuyển đến AT&T's System V UNIX làm cơ sở vào năm 1991. Năm 2005, Sun lấy phần lớn mã Solaris là dự án OpenSolaris. Tuy nhiên, việc Oracle mua Sun vào năm 2009 đã khiến tình trạng của dự án này trở nên không rõ ràng.

Một số nhóm quan tâm đến việc sử dụng OpenSolaris đã mở rộng các tính năng của nó và nhóm làm việc của họ là Project Illumos, đã mở rộng từ cơ sở OpenSolaris để bao gồm nhiều tính năng hơn và làm cơ sở cho một số sản phẩm. Illumos có sẵn tại *http://wiki.illumos.org*.

NGHIÊN CỨU VỀ CÁC HỆ ĐIỀU HÀNH

Chưa bao giờ thú vị hơn để nghiên cứu hệ điều hành và nó chưa bao giờ dễ dàng hơn thế. Phong trào mã nguồn mở đã vượt qua các hệ điều hành, khiến nhiều hệ điều hành trong số đó được cung cấp ở cả định dạng nguồn và nhị phân (có thể thực thi). Danh sách các hệ điều hành có sẵn ở cả hai định dạng bao gồm Linux, BSD UNIX, Solaris và một phần của macOS. Tính sẵn có của mã nguồn cho phép chúng ta nghiên cứu hệ điều hành từ trong ra ngoài. Những câu hỏi mà trước đây chúng ta chỉ có thể trả lời bằng cách xem tài liệu hoặc hoạt động của hệ điều hành, giờ đây chúng ta có thể trả lời bằng cách kiểm tra mã nguồn.

Hệ điều hành không còn khả thi về mặt thương mại đã có nguồn mở, cho phép chúng tôi nghiên cứu cách các hệ thống hoạt động trong một thời gian của ít tài nguyên CPU, bộ nhớ và lưu trữ hơn. Một danh sách dài nhưng chưa đầy đủ các dự án hệ điều hành mã nguồn mở có sẵn tại

http://dmoz.org/Computers/Software/OperatingSystems/OpenSource/.

Ngoài ra, sự gia tăng của ảo hóa với tư cách là chức năng máy tính phổ biến (và thường xuyên miễn phí) làm cho nó có thể chạy nhiều hệ điều hành trên cùng một hệ thống lõi. Ví dụ: VMware (*http://www.vmware.com*) cung cấp một "trình phát" miễn phí cho Windows, trên đó hàng trăm "thiết bị ảo" miễn phí có thể chạy. Virtualbox (*http://www.virtualbox.com*) cung cấp trình quản lý máy ảo mã nguồn mở miễn phí trên nhiều hệ điều hành. Sử dụng các công cụ như vậy, sinh viên có thể thử hàng trăm hệ điều hành mà không cần phần cứng bị lỗi.

Trong một số trường hợp, các trình mô phỏng của phần cứng cụ thể cũng có sẵn, cho phép hệ điều hành chạy trên phần cứng "nguyên bản", tất cả đều nằm trong phạm vi phù hợp của một máy tính hiện đại và hệ điều hành hiện đại. Ví dụ: trình mô phỏng DECSYSTEM-20 chạy trên macOS có thể khởi động TOPS-20, tải các băng nguồn, sửa đổi và biên dịch hạt nhân TOPS-20 mới. Một sinh viên quan tâm có thể tìm kiếm trên Internet để tìm các tài liệu gốc mô tả hệ điều hành, cũng như các tài liệu hướng dẫn ban đầu.

Sự ra đời của các hệ điều hành mã nguồn mở cũng giúp cho việc chuyển từ sinh viên sang nhà phát triển hệ điều hành trở nên dễ dàng hơn. Với một số kiến thức, một số nỗ lực và kết nối Internet, một sinh viên thậm chí có thể tạo ra một bản phân phối hệ điều hành mới. Cách đây không nhiều năm, rất khó hoặc không thể truy cập vào mã nguồn. Giờ đây, quyền truy cập như vậy chỉ bị giới hạn bởi mức độ quan tâm, thời gian và dung lượng đĩa mà sinh viên có.

1.11.6 Hệ thống mã nguồn mở làm công cụ học tập

Phong trào phần mềm tự do đang thúc đẩy quân đoàn lập trình viên tạo ra hàng nghìn dự án mã nguồn mở, bao gồm cả hệ điều hành. Các trang web như *http://freshmeat.net/* và *http://distrowatch.com/* cung cấp cổng thông tin cho nhiều dự án này. Như chúng tôi đã nói trước đó, các dự án mã nguồn mở cho phép sinh viên sử dụng mã nguồn như một công cụ học tập. Họ có thể sửa đổi các chương trình và kiểm tra chúng, giúp tìm và sửa lỗi cũng như khám phá các hệ điều hành, trình biên dịch, công cụ, giao diện người dùng và các loại chương trình hoàn thiện, đầy đủ tính năng. Sự sẵn có của mã nguồn cho các dự án lịch sử, chẳng hạn như Đa phương tiện, có thể giúp sinh viên hiểu các dự án đó và xây dựng kiến thức sẽ giúp thực hiện các dự án mới.

Một lợi thế khác khi làm việc với hệ điều hành mã nguồn mở là tính đa dạng của chúng. Ví dụ: GNU/Linux và BSD UNIX đều là hệ điều hành mã nguồn mở, nhưng mỗi hệ điều hành đều có mục tiêu, tiện ích, giấy phép và mục đích riêng. Đôi khi, các giấy phép không loại trừ lẫn nhau và xảy ra thụ phấn chéo, cho phép cải thiện nhanh chóng các dự án hệ điều hành. Ví dụ, một số thành phần chính của OpenSolaris đã được chuyển sang BSD UNIX. Lợi thế của phần mềm miễn phí và nguồn mở có khả năng làm tăng số lượng và chất lượng của các dự án nguồn mở, dẫn đến sự gia tăng số lượng cá nhân và công ty sử dụng các dự án này.

1.12 Tóm tắt chương

- ✓ Hệ điều hành là phần mềm quản lý phần cứng máy tính, như cũng như cung cấp một môi trường cho các chương trình ứng dụng có thể chạy được.
- ✓ Ngắt là phương thức mấu chốt để cho phần cứng tương tác với phần mềm. Thiết bị phần cứng kích hoạt ngắt bằng cách gửi tín hiệu đến CPU để cảnh báo cho CPU biết rằng có một sự kiện cần được xử lý. Ngắt được quản lý bởi các bộ điều khiển ngắt.
- ✓ Để máy tính thực hiện công việc thực thi chương trình, các chương trình phải được mang vào lưu trữ trong bộ nhớ chính, đây là vùng lưu trữ lớn duy nhất mà bộ xử lý có thể truy cập trực tiếp.
- ✓ Bộ nhớ chính là một thiết bị lưu trữ dễ bay hơi, có nghĩa là khi mất điện thì nội dung lưu trữ cũng mất theo.
- ✓ Bộ nhớ không bay hơi là phần mở rộng của bộ nhớ chính và có khả năng lưu trữ dữ liệu dung lượng lớn và vĩnh viễn.
- ✓ Thiết bị lưu trữ không bay hơi phổ biến nhất là đĩa cứng, có thể cung cấp lưu trữ cho cả chương trình và dữ liệu.
- ✓ Rất nhiều thiết bị lưu trữ trong hệ thống máy tính có thể được phân cấp theo tốc độ và chi phí. Các cấp cao hơn thường đắt tiền và rất nhanh, càng xuống cấp dưới thì giá tiền trên mỗi GB giảm và thời gian truy xuất tăng.
- ✓ Kiến trúc máy tính hiện đại là các hệ thống đa nhân trong đó mỗi CPU chứa nhiều nhân tính toán.
- ✓ Để tận dụng tốt nhất hiệu năng CPU, các hệ điều hành hiện đại sử dụng đa chương trình,

cho phép nhiều chương trình có lưu trữ trong bộ nhớ cùng một lúc, và khi đó CPU luôn có lệnh để thực thi.

- ✓ Đa nhiệm là một sự mở rộng của đa chương trình, trong đó các giải thuật định thời CPU tuần tự chuyển đổi nhanh chóng giữa các tiến trình, cung cấp cho người dùng một thời gian đáp ứng nhanh.
- ✓ Để ngăn các chương trình người dùng can thiệp vào hoạt động đúng đắn của hệ thống, phần cứng hệ thống có hai chế độ: chế độ người dùng và chế độ nhân.
- ✓ Có vài lệnh được đặc quyền và chỉ có thể được thực thi trong chế độ nhân. Ví dụ như các lệnh chuyển sang chế độ nhân, điều khiển nhập xuất, quản lý các bộ định thời và quản lý các ngắt.
- ✓ Một tiến trình là công việc đơn vị công việc nền tảng trong một hệ điều hành. Quản lý tiến trình bao gồm tạo và xóa các tiến trình và cung cấp cơ chế cho các tiến trình giao tiếp và đồng bộ hóa với nhau.
- ✓ Một hệ điều hành quản lý bộ nhớ bằng cách theo dõi những phần nào của bộ nhớ đang được sử dụng và bởi tiến trình nào. Nó cũng chịu trách nhiệm cấp phát động và giải phóng không gian bộ nhớ.
- ✓ Không gian lưu trữ được quản lý bởi hệ điều hành; bao gồm việc cung cấp hệ thống tập tin để thể hiện tập tin và thư mục và quản lý không gian trên các thiết bị lưu trữ thứ cấp.
- ✓ Hệ điều hành cung cấp các cơ chế để bảo vệ và an toàn cho cả hệ điều hành lẫn người dùng. Các biện pháp bảo vệ kiểm soát việc truy cập của các tiến trình và người dùng đến các tài nguyên được tạo ra bởi hệ thống máy tính.
- ✓ Ảo hóa bao gồm việc trừu tượng hóa một phần cứng máy tính thành nhiều môi trường thực thi khác nhau.
- ✓ Cấu trúc dữ liệu được sử dụng trong một hệ điều hành bao gồm danh sách, ngăn xếp, hàng đợi, cây và bảng ánh xạ.
- ✓ Điện toán diễn ra trong nhiều môi trường khác nhau, bao gồm cả điện tóa truyền thống, điện toán di động, hệ thống máy chủ – máy khách, hệ thống ngang hàng, điện toán đám mây và các hệ thống nhúng thời gian thực.
- ✓ Các hệ điều hành nguồn mở và miễn phí cung cấp cả mã nguồn. Phần mềm miễn phí cho phép người dùng sử dụng miễn phí, phân phối lại và cả sửa đổi. GNU / Linux, FreeBSD và Solaris là những ví dụ phổ biến cho các hệ thống nguồn mở.