CHƯƠNG 8. QUẢN LÍ HỆ THỐNG NHẬP XUẤT

Vai trò của hệ điều hành trong nhập/xuất của máy tính là quản lý và điều khiển các thao tác nhập/xuất và các thiết bị nhập/xuất. Trong chương này chúng ta sẽ mô tả các khái niệm cơ bản của phần cứng nhập/xuất. Kế đến chúng ta sẽ thảo luận các dịch vụ nhập/xuất được cung cấp bởi hệ điều hành và hiện thân của các dịch vụ này trong giao diện ứng dụng nhập/xuất. Sau đó, chúng ta giải thích hệ điều hành làm cầu nối giữa giao diện phần cứng và giao diện ứng dụng như thế nào. Cuối cùng, chúng ta thảo luận các khía cạnh năng lực của nhập/xuất và các nguyên lý thiết kế hệ điều hành để cải tiến năng lực nhập/xuất.

I. Mở đầu.

Điều khiển các thiết bị được nối kết tới máy tính là mối quan tâm chủ yếu của người thiết kế hệ điều hành. Vì các thiết bị nhập/xuất rất khác nhau về chức năng và tốc độ (xem xét chuột, đĩa cứng, và CD-ROM) nên sự đa dạng về phương pháp là cần thiết để điều khiển chúng. Các phương pháp này hình thành một hệ thống nhập/xuất con (I/O subsystem) của nhân, tách rời phần còn lại của nhân khỏi sự phức tạp của việc quản lý các thiết bị nhập/xuất.

Công nghệ thiết bị nhập/xuất thể hiện hai xu hướng trái ngược nhau. Xu hướng thứ nhất, chúng ta tăng sự chuẩn hoá phần mềm và giao diện phần cứng. Xu hướng này giúp chúng ta hợp tác những thế hệ thiết bị được cải tiến vào các máy tính và hệ điều hành đã có. Xu hướng thứ hai, chúng ta tăng sự đa dạng của các thiết bị nhập/xuất. Thiết bị mới là rất khác với các thiết bị trước đó đã tạo ra một trở ngại để hợp nhất chúng vào máy tính và hệ điều hành của chúng ta. Trở ngại này được giải quyết bởi sự kết hợp kỹ thuật phần cứng và phần mềm. Các thành phần phần cứng nhập/xuất cơ bản như cổng, bus và bộ điều khiển thiết bị chứa trong một dãy rộng các thiết bị nhập/xuất. Để đóng gói các chi tiết và sự khác biệt của các thiết bị khác nhau, nhân của hệ điều hành được chỉ dẫn để dùng các modules trình điều khiển thiết bị.

Các trình điều khiển thiết bị (device driver) hiện diện một giao diện truy xuất thiết bị đồng nhất tới hệ thống con nhập/xuất, như các lời gọi hệ thống cung cấp một giao diện chuẩn giữa ứng dụng và hệ điều hành.

II. Phần cứng nhập/xuất

Các máy tính điều hành nhiều loại thiết bị. Hầu hết chúng thuộc các chủng loại phổ biến như thiết bị lưu trữ (đĩa, băng từ), thiết bị truyền (card mạng, modem) và thiết bị giao diện người dùng (màn hình, bàn phím, chuột),.... Mặc dù có sự đa dạng về các thiết bị nhập/xuất, nhưng chúng ta chỉ cần hiểu một vài khái niệm như các thiết bị được gán như thế nào và phần mềm có thể điều khiển phần cứng như thế nào.

Một thiết bị giao tiếp với một hệ thống máy tính bằng cách gởi các tín hiệu qua dây cáp hay thậm chí qua không khí. Các thiết bị giao tiếp với máy bằng một điểm nối kết (cổng-port) như cổng tuần tự. Nếu một hay nhiều thiết bị dùng một tập hợp dây dẫn, nối kết được gọi là bus. Một bus là một tập hợp dây dẫn và giao thức được định nghĩa chặt chẽ để xác định tập hợp thông điệp có thể được gởi qua dây. Trong thuật ngữ điện tử, các thông điệp được truyền bởi các mẫu điện thế điện tử được áp dụng tới các dây dẫn với thời gian được xác định. Khi thiết bị A có một cáp gán vào thiết bị B, thiết bị B có một cáp gán vào thiết bị C và thiết bị C gán vào một cổng máy tính, sự sắp xếp này được gọi là chuỗi nối tiếp. Một chuỗi nối tiếp thường điều hành như một

bus.

II.1 Thăm dò

Giao thức hoàn chỉnh cho việc giao tiếp giữa máy tính và bộ điều khiển rất phức tạp nhưng ký hiệu bắt tay (handshaking) là đơn giản. Chúng ta giải thích bắt tay bằng thí dụ sau. Chúng ta giả sử rằng 2 bits được dùng để hợp tác trong mối quan hệ người sản xuất-người tiêu thụ giữa bộ điều khiển và máy chủ. Bộ điều khiển hiển thị trạng thái của nó thông qua bit bận (busy bit) trong thanh ghi trạng thái. Bộ điều khiển đặt bit bận khi nó đang làm việc và xoá bit bận khi nó sẵn sàng nhận lệnh tiếp theo. Máy tính ra tín hiệu mong muốn bằng bit sẵn sàng nhận lệnh (command-ready bit) trong thanh ghi lệnh. Máy tính thiết lập bit sẵn sàng nhận lệnh khi một lệnh sẵn dùng cho bộ điều khiển thực thi. Thí dụ, máy tính viết dữ liệu xuất thông qua một cổng, hợp tác với bộ điều khiển bằng cách bắt tay như sau:

1. Máy tính lặp lại việc dọc bit bận cho tới khi bit này bị xoá

2. Máy tính thiết lập bit viết trong thanh ghi lệnh và viết một byte vào thanh ghi dữ liệu xuất

3. Máy tính đặt bit sẵn sàng nhận lệnh

4. Khi bộ điều khiển nhận thấy rằng bit sẵn sàng nhận lệnh được đặt, nó đặt bit bận

5. Bộ điều khiển dọc thanh ghi lệnh và thấy lệnh viết. Nó dọc thanh ghi xuất dữ liệu để lấy một byte và thực hiện nhập/xuất tới thiết bị.

6. Bộ điều khiển xoá bit sẵn sàng nhận lệnh, xoá bit lỗi trong thanh ghi trạng thái để hiển thị rằng thiết bị nhập/xuất thành công, và xoá bit bận để hiển thị rằng nó được kết thúc.

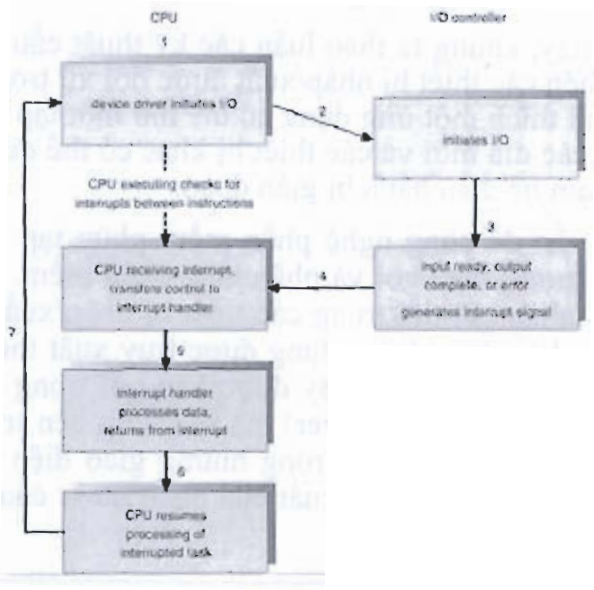
Vòng lặp này được lặp cho mỗi byte. Trong bước 1, máy tính là chờ dợi bận hay thăm dò. Nó ở trong một vòng lặp, dọc thanh ghi trạng thái cho đến khi bit bận được xoá. Nếu bộ điều khiển và thiết bị nhanh thì phương pháp này là một phương pháp phù hợp. Nhưng nếu chờ lâu máy chủ chuyển sang một tác vụ khác. Sau đó, máy tính làm thế nào để biết khi nào bộ điều khiển rảnh? Đối với một số thiết bị, máy tính phải phục vụ thiết bị nhanh chóng hoặc dữ liệu sẽ bị mất. Thí dụ, khi dữ liệu đang truyền vào cổng tuần tự từ bàn phím, một vùng đệm nhỏ trên bộ điều khiển sẽ tràn và dữ liệu sẽ bị mất nếu máy tính chờ quá lâu trước khi trả về các bytes được dọc.

Trong nhiều kiến trúc máy tính, 3 chu kỳ lệnh CPU đủ để thăm dò một thiết bị: ***read*** một thanh ghi thiết bị, thực hiện phép tính luận lý ***and*** để lấy bit trạng thái và tách ra (branch) nếu khác 0. Rõ ràng, thao tác thăm dò cơ bản là đủ. Nhưng thăm dò trở nên không đủ khi được lặp lại nhiều lần, hiếm khi tìm một thiết bị sẵn sàng phục vụ trong lần thăm dò đầu tiên, trong khi cần dùng CPU để xử lý cho các công việc khác. Trong trường hợp như thế, sẽ hiệu quả hơn để sắp xếp bộ điều khiển phần cứng thông báo cho CPU khi nào thiết bị sẵn sàng phục vụ hơn là yêu cầu CPU lặp lại việc thăm dò cho việc hoàn thành nhập/xuất. Cơ chế phần cứng cho phép một thiết bị thông báo tới CPU được gọi là ngắt (interrupt).

II.2 Ngắt

Cơ chế ngắt cơ bản làm việc như sau: phần cứng CPU có một dây dẫn được gọi là ***dòng yêu cầu ngắt*** (interrup-request line) mà CPU cảm ứng sau khi thực thi mỗi chỉ thị. Khi một CPU phát hiện một bộ điều khiển xác nhận một tín hiệu trên dòng yêu cầu ngắt thì CPU lưu một lượng nhỏ trạng thái như giá trị hiện hành của con trỏ lệnh, và

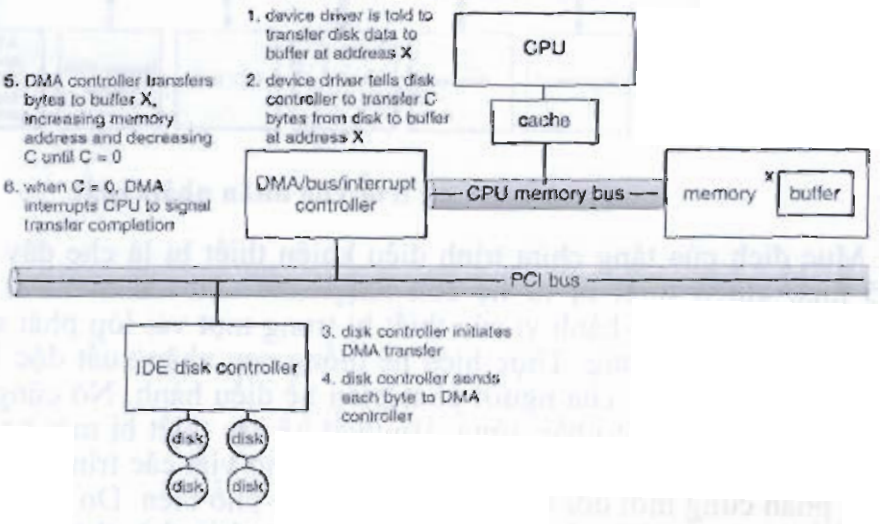
nhảy tới thủ tục của **bộ quản lý ngắt** (interrupt-handler) tại địa chỉ cố định trong bộ nhớ. Bộ quản lý ngắt xác định nguyên nhân gây ra ngắt, thực hiện xử lý cần thiết, thực thi chỉ thị *return from interrupt* để trả về CPU trạng thái thực thi trước khi ngắt. Chúng ta nói rằng bộ điều khiển thiết bị sinh ra một ngắt bằng cách xác định tín hiệu trên dòng yêu cầu ngắt và bộ quản lý xoá ngắt bằng cách phục vụ thiết bị. Hình 8.1 tóm tắt chu kỳ nhập/xuất hướng ngắt (interrupt-driven I/O cycle)



Hình 8.1. Chu kỳ nhập/xuất hướng ngắt

II.3 Truy xuất bộ nhớ trực tiếp

Đối với một thiết bị thực hiện việc truyền lớn như ổ đĩa, nó sẽ lãng phí khi dùng bộ vi xử lý để theo dõi các bit trạng thái và đẩy dữ liệu vào thanh ghi điều khiển từng byte một. Nhiều máy tính muốn giảm đi gánh nặng cho CPU bằng cách chuyển một số công việc này tới một bộ điều khiển có mục đích đặc biệt được gọi là ***bộ điều khiển truy xuất bộ nhớ trực tiếp*** (direct memory-access-DMA).



Hình 8.2. Các bước trong việc truyền dữ liệu của DMA

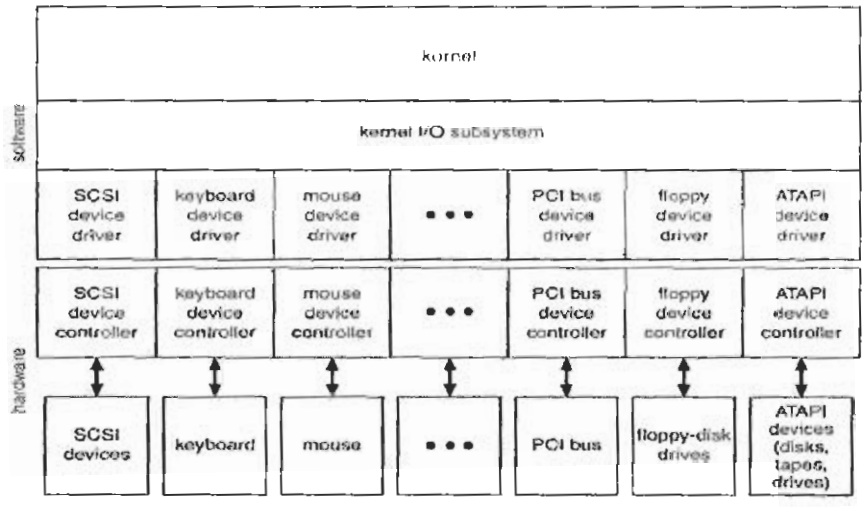
Để khởi tạo một thao tác chuyển DMA, máy tính viết một khối lệnh DMA vào bộ nhớ. Khối này chứa một con trỏ chỉ tới nguồn chuyển, một con trỏ chỉ tới đích

chuyển và đếm số lượng byte được chuyển. CPU viết địa chỉ của khối lệnh này tới bộ điều khiển DMA, sau đó CPU tiếp tục làm công việc khác. Bộ điều khiển DMA xử lý để điều hành bus bộ nhớ trực tiếp, đặt các địa chỉ trên bus để thực hiện việc chuyển mà không có sự trợ giúp của CPU. Một bộ điều khiển DMA đơn giản là một thành phần chuẩn trong PCs, và ***bảng nhập/xuất bus chính*** (bus-mastering I/O boards) để PC thường chứa phần cứng DMA tốc độ cao. Quá trình này được mô tả trong hình 8.2.

III. Giao diện nhập/xuất ứng dụng

Trong phần này, chúng ta thảo luận các kỹ thuật cấu trúc và các giao diện cho hệ điều hành cho phép các thiết bị nhập/xuất được đối xử trong cách chuẩn, không đổi. Thí dụ, chúng ta giải thích một ứng dụng có thể mở một tập tin trên đĩa mà không biết loại đĩa đó là gì và các đĩa mới và các thiết bị khác có thể được thêm tới máy tính như thế nào mà không làm hệ điều hành bị gián đoạn.

Như những vấn đề công nghệ phần mềm phức tạp khác, tiếp cận ở đây liên quan đến tính trừu tượng, bao gói và phân tầng phần mềm. Đặc biệt, chúng ta có thể trừu tượng sự khác nhau chi tiết trong các thiết bị nhập/xuất bằng cách xác định một vài loại thông dụng. Mỗi loại thông dụng được truy xuất thông qua một tập hợp hàm chuẩn-một giao diện. Sự khác biệt này được bao gói trong module nhân được gọi là ***trình điều khiển thiết bị*** (device driver) mà qui định bên trong được áp đặt cho mỗi thiết bị, nhưng được nhập vào một trong những giao diện chuẩn. Hình 8.3 hiển thị cách các thành phần liên quan nhập/xuất của nhân được cấu trúc trong các tầng phần mềm.



Hình 8.3. Cấu trúc của nhân nhập/xuất

Mục đích của tầng chứa trình điều khiển thiết bị là che dậy sự khác biệt giữa các bộ điều khiển thiết bị từ hệ con nhập/xuất của nhân, nhiều lời gọi hệ thống nhập/xuất đóng gói các hành vi của thiết bị trong một vài lớp phát sinh để che dậy sự khác biệt từ các ứng dụng. Thực hiện hệ thống con nhập/xuất độc lập với phần cứng đơn giản hóa công việc của người phát triển hệ điều hành. Nó cũng đem lại sự thuận lợi cho các nhà sản xuất phần cứng. Họ thiết kế các thiết bị mới tương thích với giao diện bộ điều khiển chủ đã có (như SCSI-2) hay họ viết các trình điều khiển thiết bị để giao tiếp phần cứng mới đối với các hệ điều hành phổ biến. Do đó, các thiết bị ngoại vi mới có thể được gán tới một máy tính mà không phải chờ nhà cung cấp hệ điều hành phát triển thêm mã.

Tuy nhiên, đối với một số nhà sản xuất thiết bị phần cứng, mỗi loại hệ điều hành có chuẩn riêng của nó cho giao diện trình điều khiển thiết bị. Một thiết bị được

cho có thể mang nhiều trình diều khiển-thí dụ, trình diều khiển cho MS-DOS, Windows 95/98, Windows NT/2000 và Solaris. Các thiết bị khác nhau trong nhiều hướng như được hiển thị trong hình 8.4.

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read&write	CD-ROM graphics controller disk

Hình 8.4. Các đặc diểm của các thiết bị nhập/xuất

- Dòng ký tự hay khối: các thiết bị dòng ký tự chuyển từng byte một, ngược lại một thiết bị khối chuyển đơn vị là khối byte.
- Truy xuất tuần tự và ngẫu nhiên: thiết bị tuần tự chuyển dữ liệu theo một thứ tự cố định được định nghĩa bởi thiết bị, ngược lại người dùng của một thiết bị truy xuất ngẫu nhiên có thể chỉ dẫn thiết bị để tìm bất cứ vị trí lưu trữ dữ liệu sẵn có.
- Đồng bộ và bất đồng bộ: một thiết bị đồng bộ là một thiết bị thực hiện việc chuyển dữ liệu với số lần đáp ứng có thể đoán trước. Một thiết bị bất đồng bộ hiển thị số lần đáp ứng không đều đặn hay không thể đoán trước.
- Có thể chia sẻ hay tận hiến: một thiết bị có thể chia sẻ được dùng đồng hành bởi nhiều quá trình hay luồng; một thiết bị tận hiến thì không thể.
- Tốc độ thao tác: tốc độ thiết bị trải dài từ một vài byte trên giây tới một vài gigabyte trên giây.
- Đọc-viết, chỉ đọc, hay chỉ viết: một số thiết bị thực hiện cả hai nhập, xuất, nhưng một số thiết bị khác hỗ trợ chỉ một hướng dữ liệu

IV. Hệ thống con nhập/xuất của nhân (kernel I/O subsytem)

Nhân cung cấp nhiều dịch vụ liên quan dến nhập/xuất. Một vài dịch vụ-định thời biểu, vùng dệm (buffering), vùng lưu trữ (cache), dặt trước thiết bị, quản lý lỗi-được cung cấp bởi hệ thống con nhập/xuất của nhân và xây dựng trên phần cứng và cơ sở hạ tầng trình diều khiển thiết bị.

IV.1 Định thời biểu nhập/xuất

Định thời biểu cho tập hợp các yêu cầu nhập xuất có nghĩa là xác định một thứ tự tốt dể thực thi chúng. Thứ tự các ứng dụng phát ra lời gọi hệ thống rất hiếm khi là một chọn lựa tốt nhất. Định thời biểu có thể cải tiến năng toàn bộ lực hệ thống, có thể chia sẻ truy xuất thiết bị dồng dều giữa các quá trình và có thể giảm thời gian chờ dợi trung bình cho nhập/xuất hoàn thành. .

Người phát triển hệ diều hành cài dặt bộ định thời biểu bằng cách duy trì một

hàng dợi cho mỗi thiết bị. Khi một ứng dụng phát ra một lời gọi hệ thống nhập/xuất nghẽn, yêu cầu được đặt vào hàng dợi cho thiết bị đó. Bộ định thời biểu nhập/xuất sắp xếp lại thứ tự của hàng dợi để cải tiến toàn bộ tính hiệu quả hệ thống và thời gian đáp ứng trung bình dựa theo kinh nghiệm bởi ứng dụng. Hệ điều hành cũng cố gắng giữ bình đẳng để mà không ứng dụng nào nhận được dịch vụ nghèo nàn hay nó cho dịch vụ ưu tiên đối với các yêu cầu trì hoãn nhạy cảm. Thí dụ, các yêu cầu từ hệ thống con bộ nhớ ảo có thể lấy độ ưu tiên qua các yêu cầu của ứng dụng.

Một cách mà hệ thống con nhập/xuất cải tiến tính hiệu quả của máy tính là bằng cách định thời biểu các hoạt động nhập/xuất. Một cách khác là dùng không gian lưu trữ trong bộ nhớ chính hay trên đĩa, với các kỹ thuật được gọi là vùng đệm, vùng lưu trữ và vùng chứa.

IV.2 Vùng đệm

Vùng đệm là một vùng bộ nhớ lưu trữ dữ liệu trong khi chúng được chuyển giữa hai thiết bị hay giữa thiết bị và ứng dụng. Vùng đệm được thực hiện với 3 lý do.

- Lý do thứ nhất là đối phó với việc không tương thích về tốc độ giữa người sản xuất và người tiêu dùng của dòng dữ liệu.
- Lý do thứ hai cho việc sử dụng vùng là làm thích ứng giữa các thiết bị có kích thước truyền dữ liệu khác nhau.
- Lý do thứ ba cho việc dùng vùng đệm là hỗ trợ ngữ nghĩa sao chép cho nhập/xuất ứng dụng.

IV.3 Vùng lưu trữ

Vùng lưu trữ (cache) là một vùng bộ nhớ nhanh quản lý các bản sao dữ liệu. Truy xuất tới một bản sao được lưu trữ hiệu quả hơn truy xuất tới bản gốc. Thí dụ, các chỉ thị của quá trình hiện đang chạy được lưu trên đĩa, được lưu trữ trong bộ nhớ vật lý và được sao chép một lần nữa trong vùng lưu trữ phụ và chính. Sự khác nhau giữa vùng đệm là vùng lưu trữ là vùng đệm có thể giữ chỉ bản sao của thành phần dữ liệu đã có, ngược lại một vùng lưu trữ giữ vừa đủ một bản sao trên thiết bị lưu trữ nhanh hơn của một thành phần nằm ở một nơi nào khác.Vùng lưu trữ và vùng đệm có chức năng khác nhau nhưng đôi khi một vùng bộ nhớ có thể được dùng cho cả hai mục đích.

IV.4 Vùng chứa và đặt trước thiết bị

Một vùng chứa (spool) là một vùng đệm giữ dữ liệu xuất cho một thiết bị như máy in mà không thể chấp nhận các dòng dữ liệu dan xen nhau. Mặc dù một máy in có thể phục vụ chỉ một công việc tại một thời điểm, nhưng nhiều ứng dụng muốn in đồng thời mà không có dữ liệu xuất của chúng dan xen với nhau. Hệ điều hành giải quyết vấn đề này bằng cách ngăn chặn tất cả dữ liệu xuất tới máy in. Dữ liệu xuất của mỗi ứng dụng được chứa trong một tập tin riêng. Khi một ứng dụng kết thúc việc in, hệ thống vùng chứa xếp tập tin chứa tương ứng cho dữ liệu xuất tới máy in. Hệ thống vùng chứa chép các tập tin được xếp hàng tới máy in một tập tin tại một thời điểm.

Trong một hệ điều hành, vùng chứa được quản lý bởi một quá trình hệ thống chạy ở chế độ nền. Trong một số hệ điều hành khác, nó được quản lý bởi luồng nhân. Trong mỗi trường hợp, hệ điều hành cung cấp một giao diện điều khiển cho phép người dùng và người quản trị hệ thống hiển thị hàng dợi để xóa các công việc không mong muốn trước khi các công việc đó in, để tạm dừng việc in trong khi máy in được phục vụ,..

IV.5 Quản lý lỗi

Một hệ điều hành sử dụng bộ nhớ bảo vệ có thể chống lại nhiều lỗi phần cứng và ứng dụng vì thế một lỗi toàn hệ thống không là kết quả của mỗi sự trục trặc cơ học thứ yếu. Các thiết bị và truyền nhập/xuất có thể bị lỗi trong nhiều cách, có thể là các lý do tạm thời như mạng trở nên quá tải, hay các lý do thường xuyên như trình điều khiển đĩa bị lỗi. Các hệ điều hành thường trả giá cho tính hiệu quả vì các lỗi tạm thời.

Thí dụ, lỗi đọc đĩa *read()* dẫn đến cố gắng làm lại *read()* và lỗi gởi dữ liệu trên mạng *send()* dẫn tới việc gởi lại *resend()* nếu giao thức được xác định rõ. Tuy nhiên, nếu một thành phần quan trọng bị lỗi thường xuyên thì hệ điều hành không nghĩ đến việc phục hồi.

Như một qui tắc thông thường, một lời gọi hệ thống nhập/xuất trả về 1 bit của thông tin về trạng thái của lời gọi, biểu thị thành công hay thất bại. Trong hệ điều hành UNIX, một biến nguyên có tên *errno* được dùng để trả về một mã lỗi- một trong 100 giá trị-hiển thị tính tự nhiên của lỗi (thí dụ: đối số vượt quá giới hạn, lỗi con trỏ, tập tin không thể mở,..). Ngược lại, một số phần cứng cung cấp thông tin lỗi được mô tả chi tiết mặc dù nhiều hệ điều hành hiện tại không được thiết kế để truyền đạt thôn tin này tới ứng dụng.

IV.6 Cấu trúc dữ liệu nhân

Nhân cần giữ thông tin trạng thái về việc dùng các thành phần nhập/xuất. Nó thực hiện như thế thông qua một dãy các cấu trúc dữ liệu trong nhân như bảng tập tin đang mở. Nhân dùng nhiều cấu trúc tương tự để ghi vết các nối kết mạng, giao tiếp thiết bị dạng ký tự và các hoạt động nhập/xuất khác.

Tóm lại, hệ thống con nhập/xuất điều phối tập hợp dịch vụ mở rộng sẵn có đối với ứng dụng và những phần khác của nhân. Hệ thống con nhập/xuất điều khiển

- Quản lý không gian tên cho các tập tin và các thiết bị
- Điều khiển truy xuất tới các tập tin và các thiết bị
- Điều khiển hoạt động (thí dụ, một modem không thể tìm seek())
- Cấp phát không gian hệ thống tập tin
- Cấp phát thiết bị
- Vùng đệm, vùng lưu trữ và vùng chứa
- Định thời biểu nhập/xuất
- Điều khiển trạng thái thiết bị, quản lý lỗi, và phục hồi lỗi
- Cấu hình và khởi tạo trình điều khiển thiết bị

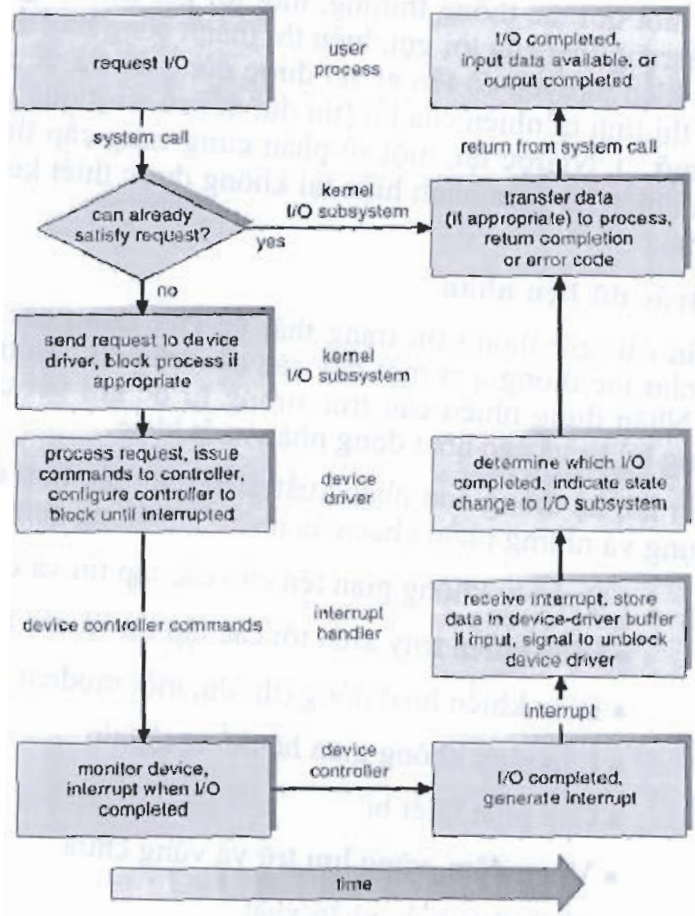
Cấp cao hơn của hệ thống con nhập/xuất truy xuất thiết bị qua giao diện đồng nhất được cung cấp bởi các trình điều khiển thiết bị

V. Chuyển nhập/xuất tới hoạt động phần cứng

Phần trước chúng ta mô tả việc bắt tay giữa một trình điều khiển thiết bị và bộ điều khiển thiết bị, nhưng chúng ta không giải thích cách hệ điều hành nối kết yêu cầu ứng dụng tới tập hợp dây mạng hay một sector đĩa xác định như thế nào. Chúng ta hãy xem xét một thí dụ đọc một tập tin từ đĩa. Ứng dụng tham chiếu tới dữ liệu bằng tên tập tin. Trong một đĩa, hệ thống tập tin ánh xạ từ tên tập tin thông qua các thư mục hệ thống tập tin để lấy không gian cấp phát của tập tin.

Các hệ điều hành hiện đại đạt được khả năng linh hoạt cao từ nhiều giai đoạn của bảng tra cứu trong đường dẫn giữa yêu cầu và bộ điều khiển thiết bị vật lý. Các cơ chế truyền yêu cầu giữa ứng dụng và trình điều khiển là phổ biến. Do đó, chúng ta có thể giới thiệu các thiết bị mới và trình điều khiển vào máy tính mà không biên dịch lại nhân. Thật vậy, một số hệ điều hành có khả năng nạp trình điều khiển thiết bị theo yêu cầu. Tại thời điểm khởi động, hệ thống đầu tiên thăm dò các bus phần cứng để xác định thiết bị nào hiện diện và sau đó hệ thống nạp các trình điều khiển cần thiết ngay lập tức hay khi được yêu cầu bởi một yêu cầu nhập/xuất đầu tiên.

Bây giờ chúng ta mô tả chu trình sống điển hình của một yêu cầu đọc bị nghẽn, như trong hình 8.5. Hình này đề nghị rằng một thao tác nhập/xuất yêu cầu nhiều bước và tiêu tốn số lượng lớn chu kỳ CPU.



Hình 8.5. Chu trình sống của yêu cầu nhập/xuất

1) Một quá trình phát ra một lời gọi hệ thống read() tới bộ mô tả tập tin đã được mở trước đó.

2) Mã lời gọi hệ thống trong nhân kiểm tra tính đúng đắn của các tham số. Trong trường hợp nhập, nếu dữ liệu đã có sẵn trong vùng đệm thì dữ liệu được trả về tới quá trình và yêu cầu nhập/xuất được hoàn thành.

3) Ngược lại, nhập/xuất vật lý cần được thực hiện để mà quá trình được xóa từ hàng đợi thực thi và được đặt vào hàng đợi chờ cho thiết bị, và yêu cầu nhập/xuất được lập thời biểu. Cuối cùng, hệ con nhập/xuất gởi yêu cầu tới trình điều khiển thiết bị. Phụ thuộc vào hệ điều hành, yêu cầu được gởi bằng lời gọi thủ tục con hay bằng thông điệp trong nhân.

4) Trình điều khiển thiết bị cấp phát vùng đệm nhân để nhận dữ liệu và lập thời biểu nhập/xuất. Cuối cùng, trình điều khiển gởi lệnh tới bộ điều khiển thiết bị bằng cách viết vào thanh ghi điều khiển của thiết bị.

5) Trình điều khiển thiết bị thao tác trên phần cứng thiết bị để thực hiện truyền dữ liệu.

6) Trình điều khiển có thể thăm dò trạng thái và dữ liệu hay thiết lập truyền DMA vào bộ nhớ nhân. Chúng ta thừa nhận rằng truyền được quản lý bởi bộ điều khiển DMA sinh ra một ngắt khi việc truyền hoàn thành.

7) Bộ quản lý ngắt tương ứng nhận ngắt bằng bằng vector ngắt, lưu bất cứ dữ liệu cần thiết, báo hiệu trình điều khiển thiết bị và trả về từ ngắt.

8) Trình điều khiển thiết bị nhận tín hiệu, xác định yêu cầu nhập/xuất hoàn thành, xác định trạng thái yêu cầu và báo hiệu cho hệ con nhập/xuất nhân rằng yêu cầu đã hoàn thành.

9) Nhân truyền dữ liệu hay trả về mã tới không gian địa chỉ của quá trình được yêu cầu và di chuyển quá trình từ hàng đợi chờ tới hàng đợi sẵn sàng.

10) Di chuyển quá trình tới hàng đợi sẵn sàng không làm nghẽn quá trình. Khi bộ định thời biểu gán quá trình tới CPU, quá trình tiếp tục thực thi tại thời điểm hoàn thành của lời gọi hệ thống.

VI. Năng lực

Nhập/xuất là một yếu tố quan trọng trong năng lực hệ thống. Nó đặt nhiều yêu cầu trên CPU để thực thi mã trình điều khiển thiết bị và định thời biểu quá trình công bằng và hiệu quả khi các quá trình này nghẽn và không nghẽn. Chuyển đổi ngữ cảnh chú trọng đến CPU và vùng lưu trữ phần cứng. Nhập/xuất cũng hiển thị tính không hiệu quả trong các cơ chế quản lý ngắt trong nhân, và nhập/xuất tải xuống bus bộ nhớ trong suốt thời gian chép giữa vùng đệm nhân và không gian dữ liệu ứng dụng. Chép một cách hợp lý tất cả yêu cầu này là một trong những quan tâm chính của kiến trúc máy tính.

Mặc dù các máy tính hiện đại có thể quản lý hàng ngàn ngắt trên giây, quản lý ngắt là một tác vụ tương đối dắt: mỗi ngắt gây cho hệ thống thực hiện một thay đổi trạng thái, để thực thi bộ quản lý ngắt và sau đó phục hồi trạng thái. Nhập/xuất được lập trình có thể hiệu quả hơn nhập/xuất hướng ngắt (interrupt-driven I/O) nếu số chu kỳ tiêu tốn cho việc chờ đợi bận là không quá mức. Hoàn thành một thao tác nhập/xuất không nghẽn một quá trình dẫn đến toàn bộ chi phí của việc chuyển đổi ngữ cảnh.

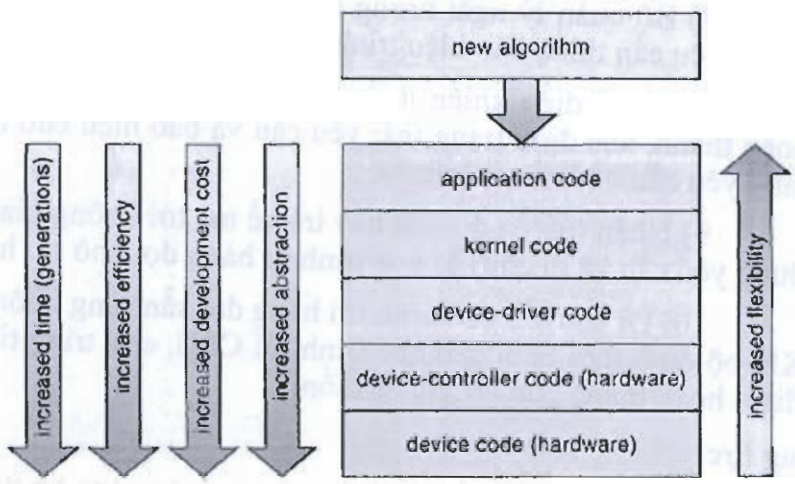
Chúng ta có thể tận dụng nhiều nguyên tắc để cải tiến tính hiệu quả của nhập/xuất:

- Cắt giảm số lượng chuyển ngữ cảnh
- Cắt giảm số lần dữ liệu phải được chép vào bộ nhớ trong khi truyền giữa thiết bị và ứng dụng.
- Cắt giảm tần số xuất hiện ngắt bằng cách dùng sự truyền lớn, bộ điều khiển thông tin và vùng chứa (nếu chờ đợi bận có thể là nhỏ nhất).
- Gia tăng tính đồng hành dùng các bộ điều khiển tri thức DMA (DMA-knowledgeable controllers) hay các kênh để giảm gánh nặng chép dữ liệu đơn giản từ CPU.

• Di chuyển các hàm xử lý cơ bản vào phần cứng, để cho phép hoạt động của chúng trong các bộ điều khiển thiết bị đồng hành với các thao tác CPU và bus.

• Cân bằng CPU, hệ con bộ nhớ, bus và năng lực nhập/xuất vì quá tải trong một vùng bất kỳ sẽ gây rành rỗi trong vùng khác.

Ở đây các chức năng nhập/xuất nên được cài đặt-trong phần cứng thiết bị, trong trình điều khiển thiết bị hay trong phần mềm ứng dụng? Chúng ta quan sát tiến trình được mô tả trong hình 8.6.



Hình 8.6. Tiến trình mô tả chức năng thiết bị

• Khởi đầu, chúng ta cài đặt giải thuật nhập/xuất thử nghiệm tại cấp ứng dụng vì mã ứng dụng là linh hoạt và những lỗi ứng dụng là không chắc gây ra sự sụp đổ hệ thống. Ngoài ra, bằng phát triển mã tại cấp ứng dụng, chúng ta tránh yêu cầu khởi động hay nạp lại trình điều khiển thiết bị sau mọi thay đổi tới mã. Tuy nhiên, cài đặt cấp ứng dụng có thể không đủ vì chi phí chuyển ngữ cảnh và vì ứng dụng không thể lấy lợi điểm của những cấu trúc dữ liệu nhân bên trong và chức năng nhân (như truyền thông điệp hữu hiệu trong nhân, luồng và khóa).

• Khi một giải thuật cấp ứng dụng chứng minh tính giá trị của nó, chúng ta có thể cài đặt lại nó trong nhân. Điều này có thể cải tiến năng lực nhưng nỗ lực phát triển có thử thách nhiều hơn vì nhân hệ điều hành lớn, phần mềm hệ thống phức tạp. Ngoài ra, việc cài đặt trong nhân phải được gỡ rối toàn bộ để tránh hư hỏng dữ liệu và sụp đổ hệ thống.

• Năng lực cao nhất có thể đạt được bởi cài đặt chuyên dụng trong phần cứng, trong thiết bị hay trong bộ điều khiển. Sự bất lợi của việc cài đặt phần cứng gồm khó khăn và chi phí của việc tạo những cải tiến xa hơn hay sửa lỗi, thời gian phát triển tăng (tháng hơn là ngày) và khả năng linh hoạt giảm.

Ghi nhớ

Các thành phần phần cứng cơ bản được nạp vào nhập/xuất là các bus, các bộ điều khiển thiết bị, và chính các thiết bị. Công việc chuyển dữ liệu giữa thiết bị và bộ nhớ chính được thực hiện bởi CPU khi nhập/xuất được lập trình, hay được chuyển tải tới bộ điều khiển DMA. Module nhân điều khiển một thiết bị là một trình điều khiển thiết bị. Giao diện lời gọi hệ thống cung cấp tới ứng dụng được thiết kế để quản lý nhiều chủng loại cơ bản của phần cứng, sockets mạng và bộ đếm thời gian đến được

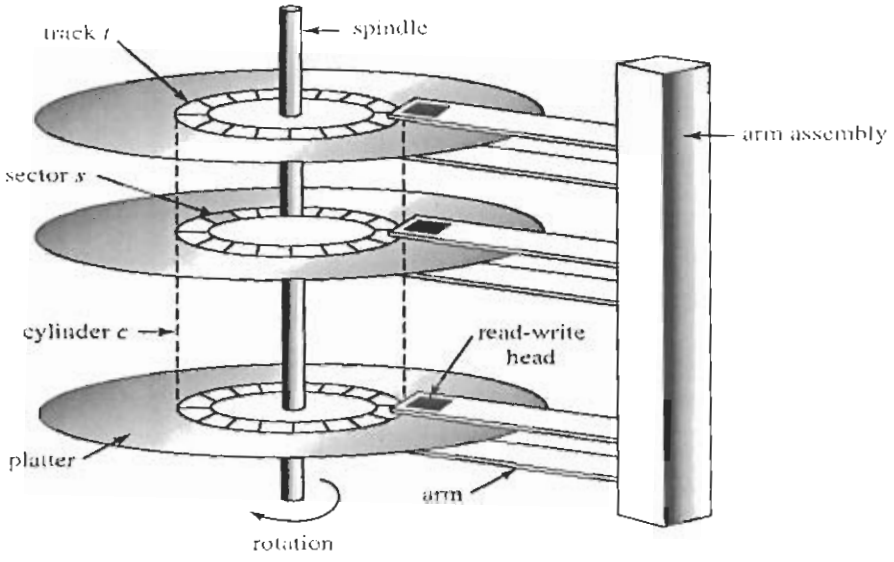
lập trình.

Hệ con nhập/xuất của nhân cung cấp nhiều dịch vụ. Các dịch vụ này là định thời biểu nhập/xuất, vùng đệm, vùng chứa, quản lý lỗi và đặt trước thiết bị. Một dịch vụ khác là dịch tên, để tạo nối kết giữa các thiệt bị phần cứng và tên tập tin tượng trưng được dùng bởi ứng dụng. Nó liên quan nhiều cấp ánh xạ mà dịch từ tên chuỗi ký tự tới một trình điều khiển thiết bị xác định và địa chỉ thiết bị và sau đó tới địa chỉ vật lý của cổng nhập/xuất hay bộ điều khiển bus

Các lời gọi hệ thống nhập/xuất tính chi phí theo thuật ngữ tiêu tốn CPU vì nhiều lớp phần mềm giữa thiết bị vật lý và ứng dụng. Các lớp này ngụ ý chi phí chuyển ngữ cảnh để đi qua phạm vi bảo vệ của nhân, của tín hiệu và bộ quản lý ngắt để phục vụ các thiết bị nhập/xuất, và của tải trên CPU và hệ thống bộ nhớ để chép dữ liệu giữa vùng đệm nhân và không gian ứng dụng.

VII. Bài toán lập lịch cho ổ đĩa.

Một đĩa cứng bao gồm nhiều mặt đĩa, mỗi mặt đĩa có cấu trúc lưu trữ thông tin gọi là các track, mỗi track lại chia thành nhiều đoạn gọi là các sector, mỗi sector có số byte là như nhau. Tập hợp track có cùng bán kính tính từ tâm đĩa gọi là cylinder và các cylinder được đánh địa chỉ theo thứ tự. Khi muốn truy xuất thông tin ghi trên một sector thì cánh tay đòn phải đặt đầu đọc/ghi lên đúng cylinder đó đồng thời motor quay đĩa để sector đó chạy qua đầu đọc/ghi. Cấu trúc cơ bản được chỉ ra trong hình 8.7.



Hình 8.7. Cấu trúc cơ bản của ổ đĩa cứng

Hiệu suất về mặt thời gian của máy tính phụ thuộc rất lớn vào việc nạp chương trình và nhập suất dữ liệu từ các tập tin, do đó dịch vụ nhập xuất phải càng nhanh càng tốt. Một trong những hướng giải quyết là lập lịch điều khiển việc truy xuất đĩa.

Tốc độ truy xuất đĩa bị chi phối bởi ba yếu tố chính. Đầu tiên muốn truy xuất thông tin trên một khối của tập tin thì bộ điều khiển phải chuyển đầu đọc/ghi đến cylinder (track) chứa khối đó, thao tác này gọi là seek và thời gian để thực hiện nó gọi là seek time. Sau khi đầu đọc/ghi đặt đúng track rồi thì còn phải chờ khối (sector) được đưa đến dưới đầu đọc/ghi, thời gian chờ này gọi là độ trễ (latency time). Cuối cùng là thời gian vận chuyển dữ liệu giữa ổ đĩa và bộ nhớ trong, gọi là transfer time. Tổng thời ba loại thời gian trên sẽ quy định thời gian truy xuất đĩa. Trong 3 yếu tố đó thì latency

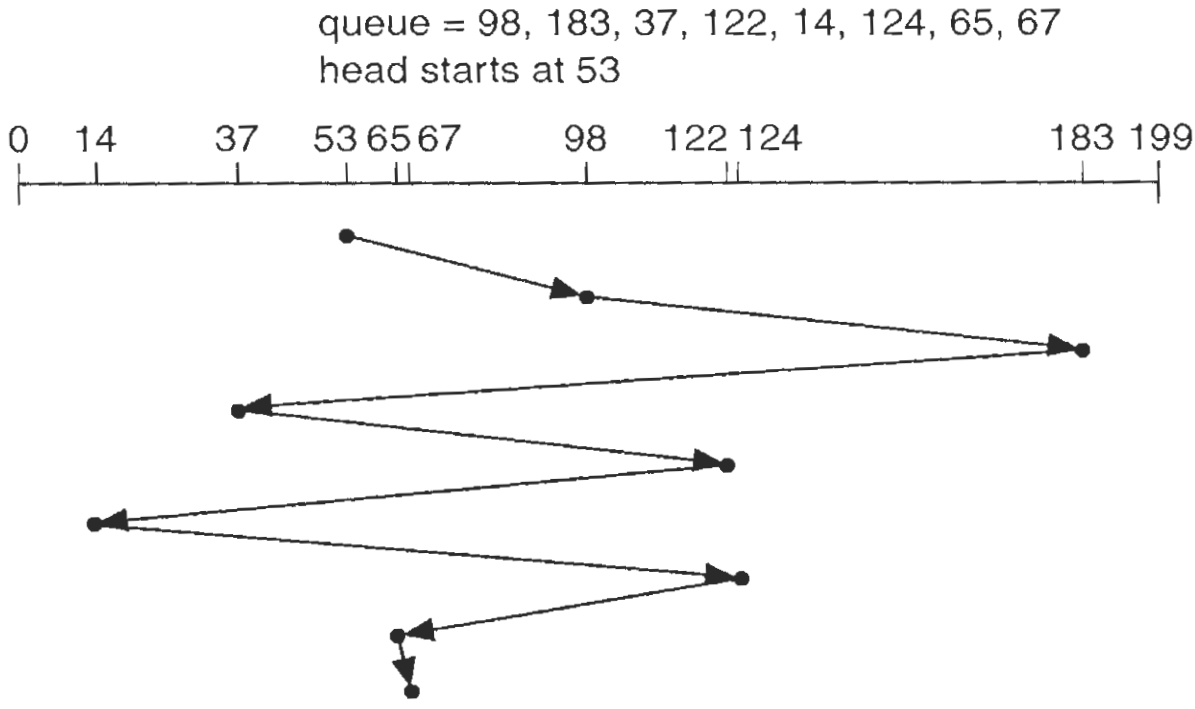
time bị chi phối bởi tốc độ quay của ổ đĩa và transfer time cố định bởi cấu trúc máy tính, có nghĩa là hệ điều hành không có khả năng tác động, vậy thì bài toán của chúng ta tập trung vào việc tối ưu hóa tổng thời gian seek time.

Giả sử một tập tin được truy xuất, tập tin này được lưu trữ một dãy các khối có thứ tự, được phân bố trên các track khác nhau. Khi truy xuất, hệ điều hành sẽ tạo ra một hàng đợi các yêu cầu đọc/ghi trên track, bài toán lập lịch của chúng ta chính là tổ chức hàng đợi truy xuất đó.

VII.1. Phương pháp lập lịch FCFS (First – Come, First – Served)

Phương pháp lập lịch FCFS là cách chúng ta tổ chức hàng đợi truy xuất theo đúng thứ tự của dãy khối lưu trữ. Ví dụ chúng ta có một tập tin được lưu trữ trên các khối theo đúng thứ tự (để đơn giản chúng ta coi địa chỉ khối trùng với địa chỉ track) như sau: 98, 183, 37, 122, 14, 124, 65, 67. Đầu đọc/ghi trước khi truy xuất nằm ở khối 53.

Phương pháp FCFS sẽ đưa đầu đọc/ghi đi qua lần lượt các khối 53, 98, 183, 37, 122, 14, 124, 65, 67. Mô phỏng quãng đường seek như trong hình 8.8.



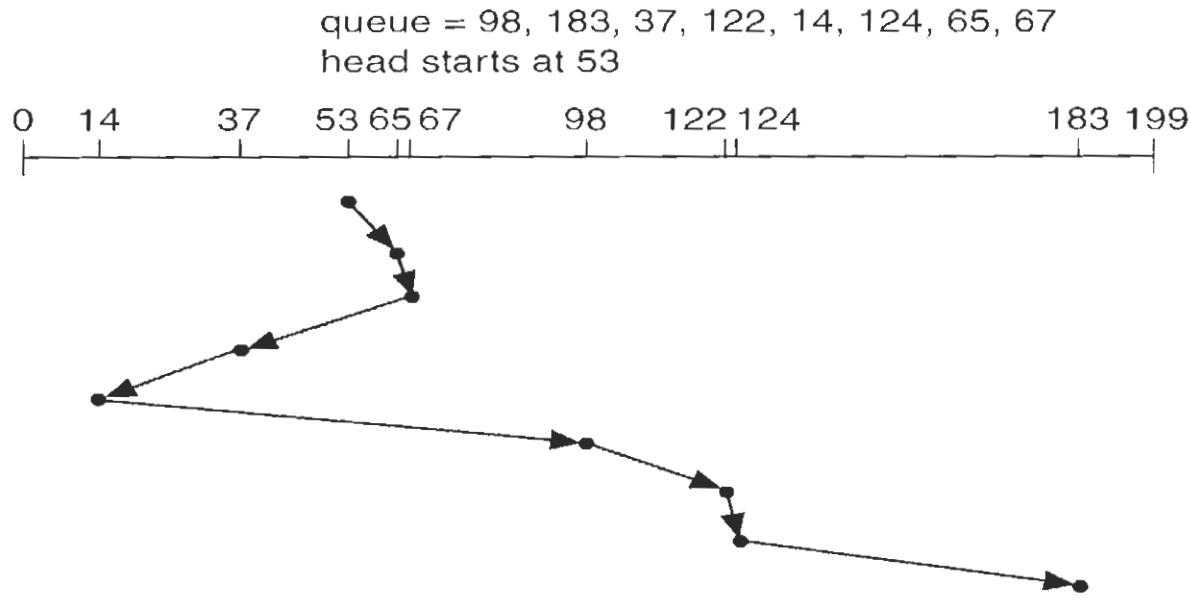
Hình 8.8. Mô phỏng phương pháp FCFS

Để tiện cho việc so sánh giữa các phương pháp chúng ta lấy một giá trị gọi là đại diện cho tổng quãng đường phải seek, từ đó suy ra thời gian seek. Mỗi khi đầu đọc/ghi chuyển từ track này qua track khác nó phải đi qua các track nằm giữa hai track đó, số track nó phải vượt qua lấy làm quãng đường đi. Ví dụ từ track số 53 di chuyển đến track số 98 quãng đường là 45 (98-53). Với cách tính đó ta có quãng đường theo FCFS là 640.

Nhìn trên sơ đồ mô phỏng ta thấy ngay là quãng đường phải đi theo FCFS là khá lớn.

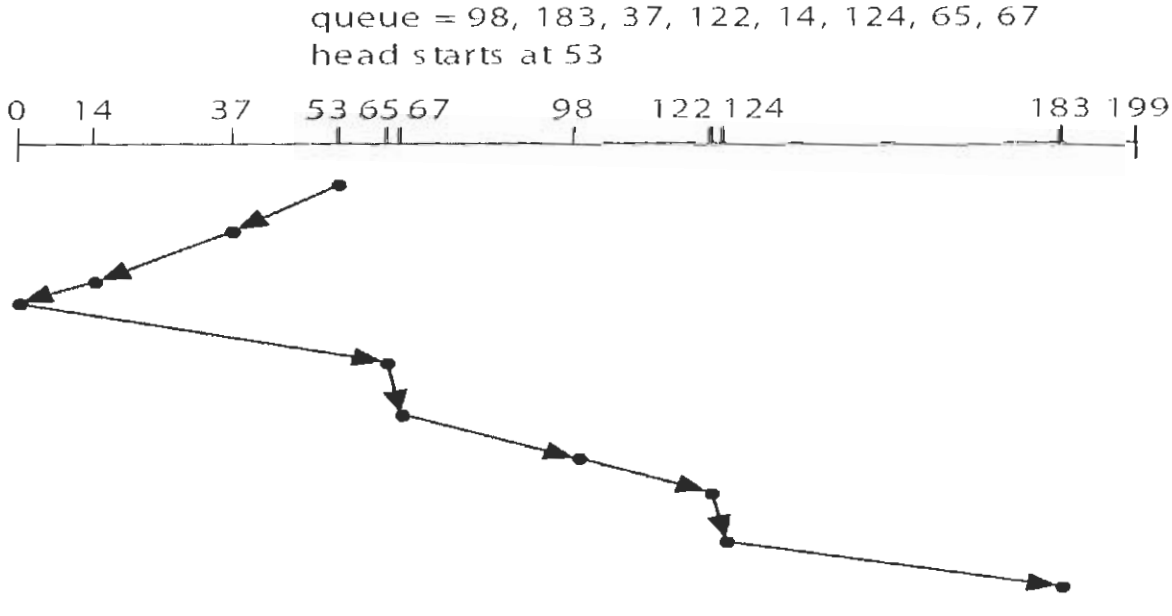
VII.2. Phương pháp lập lịch SSTF (Shortest Seek Time First)

Phương pháp SSTF tổ chức hàng đợi theo ưu tiên quãng đường di chuyển ngắn nhất tại mỗi thời điểm. Tại mỗi thời điểm, vị trí được chọn tiếp theo là track gần với vị trí đầu đọc/ghi nhất. Phương pháp FCFS sẽ đưa đầu đọc/ghi đi qua lần lượt các khối 53, 65, 67, 37, 14, 98, 122, 124, 183. Mô phỏng quãng đường seek như trong hình 8.9. Quãng đường theo SSTF là 236.



Hình 8.9. Mô phỏng phương pháp FCFS

VII.3. Phương pháp lập lịch quét SCAN.



Hình 8.10. Mô phỏng phương pháp SCAN

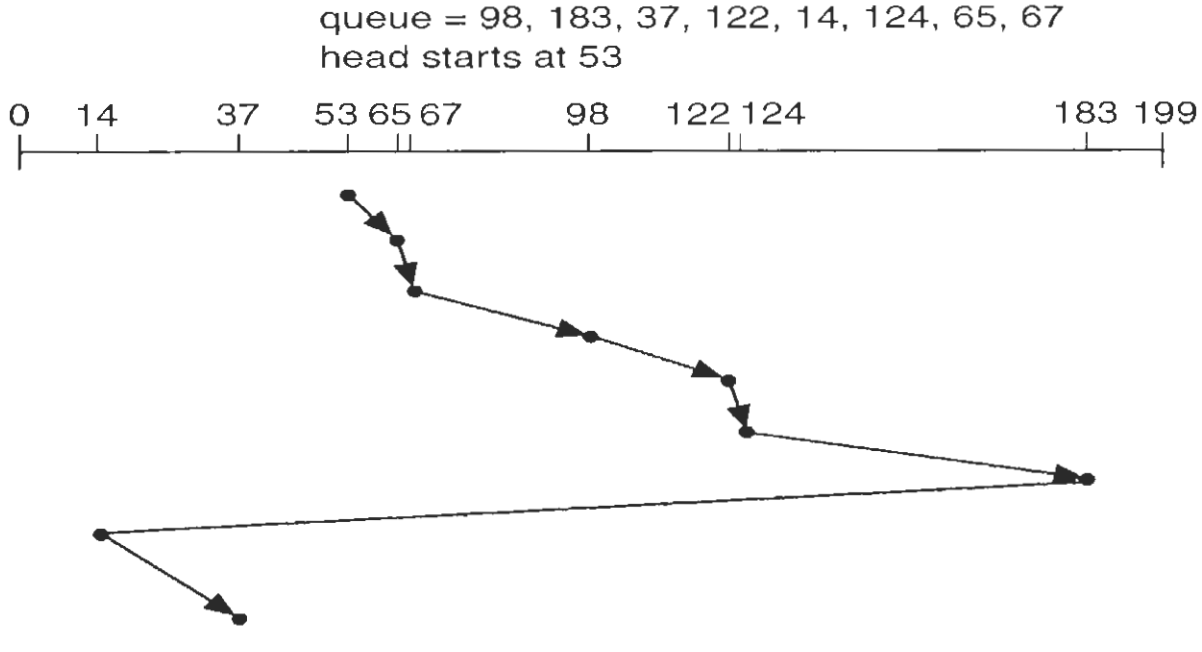
Phương pháp SCAN sẽ dịch chuyển đầu đọc ghi đi theo một hướng, đầu tiên dịch chuyển về phía trái (về phía địa chỉ nhỏ hơn) cho đến khi gặp track 0 thì quay lại dịch chuyển về phía phải (về phía địa chỉ lớn), giống thao tác của người cầm chổi quét. Ở đây việc di chuyển về trái trước hay phải trước là tương đương nhau.

Theo SCAN với ví dụ trên chúng ta có hàng đợi như sau: 53, 37, 14, 0, 65, 67, 98, 122, 124, 183. Quãng đường phải đi là 208. Mô phỏng quãng đường seek như trong hình 8.10.

Trên sơ đồ mô phỏng ta thấy quãng đường đi từ track 14 đến track 0 là quãng đường vô ích. Một chút sửa đổi là về mỗi phía ta chỉ dịch chuyển đến điểm xa nhất có trong dãy địa chỉ, trong ví dụ trên thì về phía trái là track 14 còn về phía phải là track 183. Phương pháp đó gọi là giải thuật "thang máy".

VII.4. Phương pháp lập lịch quét vòng tròn C-SCAN.

Về cơ bản giống như phương pháp SCAN, nhưng khi đến điểm xa nhất về mỗi phía thì đầu đọc/ghi được chuyển dịch đến track đầu phía ngược lại của dãy. Với ví dụ trên thì C-SCAN đưa ra hàng đợi như sau: 53, 65, 67, 98, 122, 124, 183, 14, 37. Mô phỏng quãng đường seek như trong hình 8.11.



Hình 8.11. Mô phỏng phương pháp C-SCAN

VII.5. Lựa chọn phương pháp lập lịch.

Không có phương pháp lập lịch nào tối ưu cho mọi trường hợp. Chẳng hạn qua ví dụ trên thì có vẻ như FCFS là phương pháp tồi nhất (đưa ra quãng đường đi dài nhất), nhưng cũng phương pháp đó mà áp dụng trên tập tin có dãy địa chỉ 14, 37, 65, 67, 98, 122, 124, 183 thì lại cho kết quả tốt nhất. Phương pháp SSTF thì rất tự nhiên, phương pháp SCAN hoặc C-SCAN thích hợp cho những tập tin lớn. Thường thì hệ điều hành áp dụng một phương pháp dựa trên cơ sở phân tích dãy địa chỉ dựa trên tiêu chí: liên tục/rời rạc, tuần tự/không tuần tự, độ dài, ...