BTL hệ điều hành

Hệ Điều Hành (Trường Đại học Bách khoa Hà Nội)

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
KHOA KHOA HỌC MÁY TÍNH
----------o0o----------

BÀI TẬP LỚN

HỌC PHẦN: NGUYÊN LÝ HỆ ĐIỀU HÀNH

(Mã học phần: IT3070)

Đề tài:

HỆ THỐNG VÀO RA
TRÊN HỆ ĐIỀU HÀNH ANDROID

GVHD: Thầy Đỗ Tuấn Anh
SVTH: Ngô Hải Văn
MSSV: 20200659

Hà Nội, tháng 7 năm 2022

MỤC LỤC

LỜI CẢM ƠN

Lời đầu tiên, em xin trân trọng cảm ơn và bày tỏ lòng biết ơn sâu sắc nhất tới thầy **Đỗ Tuấn Anh** – Giảng viên Viện Công nghệ thông tin & Truyền thông, Trường Đại học Bách Khoa Hà Nội, giáo viên hướng dẫn bài tập lớn đã nhiệt tình giảng dạy, hướng dẫn, chỉ bảo.

Và em cũng xin dành lời cảm ơn chân thành tới bạn bè đã động viên, khuyến khích và tạo điều kiện cho em hoàn thành tốt đề tài của mình.

Mặc dù đã cố gắng hoàn thiện sản phẩm nhưng không thể tránh khỏi những thiếu hụt về kiến thức. Em mong muốn nhận được những nhận xét thẳng thắn, chi tiết đến từ thầy để tiếp tục hoàn thiện hơn nữa. Cuối cùng, em xin được gửi lời cảm ơn đến thầy Đỗ Tuấn Anh đã hướng dẫn em trong suốt quá trình hoàn thiện bài tập lớn. Xin trân trọng cảm ơn thầy.

Xin chân thành cảm ơn!

Hà Nội, tháng 7 năm 2022

Sinh viên

Ngô Hải Văn

Chương 1. Giới thiệu đề tài

1.1. Tổng quan đề tài

Việc kiểm soát các thiết bị được kết nối với máy tính là mối quan tâm lớn của các nhà thiết kế hệ điều hành.Bởi các thiết bị vào ra rất khác nhau về chức năng và tốc độ nên các phương pháp khác nhau để kiểm soát chúng là rất cần thiết.

1.2. Nhiệm vụ đề tài

Mô tả các nhiệm vụ của đề tài bao gồm yêu cầu, kết quả cần đạt và giới hạn đề tài. Trong từng nội dung sinh viên cũng cần trình bày thêm cách tiếp cận cũng như ý tưởng thực hiện.

Nội dung 1: Tìm hiểu về phần cứng vào ra
Nội dung 2: Tìm hiểu về giao diện vào ra của ứng dụng
Nội dung 3: Hệ thống con I/O
Nội dung 4: Chuyển đổi yêu cầu vào ra sang hoạt động phần cứng
Nội dung 5: Luồng (Streams)
Nội dung 6: Hiệu suất

Chương 2. Lý Thuyết

2.1 Phân cứng

Máy tính vận hành rất nhiều loại thiết bị. Hầu hết phù hợp với các loại thiết bị lưu trữ chung (đĩa, băng), thiết bị truyền dẫn (mạng con-nections, Bluetooth) và các thiết bị giao diện con người (màn hình, bàn phím, chuột, âm thanh trong và ngoài). Các thiết bị khác chuyên biệt hơn, chẳng hạn như những thiết bị liên quan đến việc điều khiển máy bay phản lực. Trong những chiếc máy bay này, con người cung cấp đầu vào cho máy tính bay thông qua cần điều khiển và bàn đạp chân, và máy tính gửi các lệnh đầu ra khiến động cơ di chuyển bánh lái và cánh tà và nhiên liệu đến động cơ. Mặc dù có sự đa dạng đáng kinh ngạc của các thiết bị I / O, tuy nhiên, chúng ta chỉ cần một vài khái niệm để hiểu cách các thiết bị được gắn và làm thế nào phần mềm có thể kiểm soát phần cứng.

Một thiết bị giao tiếp với một hệ thống máy tính bằng cách gửi tín hiệu qua cáp hoặc thậm chí qua không khí. Thiết bị giao tiếp với máy thông qua một điểm kết nối, hoặc cổng — ví dụ, một cổng nối tiếp. Nếu các thiết bị chia sẻ một bộ dây chung, kết nối được

gọi là bus. Bus là một tập hợp các dây và một giao thức được xác định cứng nhắc chỉ định một tập hợp các tin nhắn có thể được gửi trên dây. Về mặt điện tử, các thông điệp được truyền tải bằng các mẫu điện áp áp dụng cho các dây với thời gian xác định. Khi thiết bị A có cáp cắm vào thiết bị B và thiết bị B có cáp cắm vào thiết bị C và thiết bị C cắm vào cổng trên máy tính, sự sắp xếp này được gọi là chuỗi daisy. Một chuỗi daisy thường hoạt động như một chiếc xe buýt.
Làm thế nào bộ xử lý có thể cung cấp lệnh và dữ liệu cho bộ điều khiển để thực hiện chuyển I / O? Câu trả lời ngắn gọn là bộ điều khiển có một hoặc nhiều đăng ký dữ liệu và tín hiệu điều khiển. Bộ xử lý giao tiếp với bộ điều khiển bằng cách đọc và viết các mẫu bit trong các đăng ký này. Một cách mà giao tiếp này có thể xảy ra là thông qua việc sử dụng các hướng dẫn I / O đặc biệt chỉ định việc chuyển byte hoặc từ đến địa chỉ cổng I / O. Hướng dẫn I /O kích hoạt các dòng xe buýt để chọn thiết bị thích hợp và di chuyển các bit vào hoặc ra khỏi sổ đăng ký thiết bị. Ngoài ra, bộ điều khiển thiết bị có thể hỗ trợ I / O được ánh xạ bộ nhớ . Trong trường hợp này, các thanh ghi điều khiển thiết bị được ánh xạ vào không gian địa chỉ của bộ xử lý. CPU thực hiện các yêu cầu I/O bằng cách sử dụng các hướng dẫn truyền dữ liệu tiêu chuẩn để đọc và ghi các thanh ghi điều khiển thiết bị tại các vị trí được ánh xạ của chúng trong bộ nhớ vật lý.

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

Figure 13.2 Device I/O port locations on PCs (partial).

2.1.1 Polling

Giao thức hoàn chỉnh để tương tác giữa máy chủ và bộ điều khiển có thể phức tạp, nhưng khái niệm bắt tay cơ bản rất đơn giản. Chúng tôi giải thích bắt tay với một ví dụ. Giả sử rằng 2 bit được sử dụng để phối hợp mối quan hệ giữa nhà sản xuất và người tiêu

dùng giữa bộ điều khiển và máy chủ. Bộ điều khiển cho biết trạng thái của nó thông qua bit bận rộn trong sổ đăng ký trạng thái. (Hãy nhớ rằng để thiết lập một chút có nghĩa là viết một 1 vào bit và để xóa một chút có nghĩa là để viết một 0 vào nó.) Bộ điều khiển đặt bit bận rộn khi nó bận làm việc và xóa bit bận rộn khi nó sẵn sàng chấp nhận lệnh tiếp theo. Máy chủ báo hiệu mong muốn của mình thông qua bit sẵn sàng lệnh trong sổ đăng ký lệnh. Máy chủ đặt bit sẵn sàng ra lệnh khi có lệnh để bộ điều khiển thực thi. Đối với ví dụ này, máy chủ ghi đầu ra thông qua một cổng, phối hợp với bộ điều khiển bằng cách bắt tay như sau:

1. Máy chủ liên tục đọc bit cho đến khi bit đó trở nên rõ ràng.
2. Máy chủ đặt bit ghi trong sổ đăng ký lệnh và viết byte vào sổ đăng ký dữ liệu.
3. Máy chủ đặt bit sẵn sàng ra lệnh.
4. Khi bộ điều khiển nhận thấy rằng bit sẵn sàng lệnh được đặt, nó sẽ đặt bit bận rộn.
5. Bộ điều khiển đọc sổ đăng ký lệnh và thấy lệnh ghi. Nó đọc sổ đăng ký dữ liệu ra để có được byte và thực hiện I / O cho thiết bị.
6. Bộ điều khiển xóa bit sẵn sàng lệnh, xóa bit lỗi trong sổ đăng ký trạng thái để chỉ ra rằng thiết bị I / O đã thành công và xóa bit bận để chỉ ra rằng nó đã hoàn thành.

2.1.2 Interrupts (Ngắt)

Cơ chế ngắt cơ bản vừa được mô tả cho phép CPU phản ứng với một sự kiện không đồng bộ, như khi bộ điều khiển thiết bị sẵn sàng phục vụ. Tuy nhiên, trong một hệ điều hành hiện đại, chúng ta cần các tính năng xử lý gián đoạn tinh vi hơn.
Trong phần cứng máy tính hiện đại, ba tính năng này được cung cấp bởi CPU và phần cứng điều khiển ngắt. Hầu hết các CPU có hai dòng yêu cầu ngắt quãng. Một là ngắt không thể che giấu, được dành riêng cho các sự kiện như lỗi bộ nhớ không thể phục hồi. Dòng ngắt thứ hai có thể che giấu: cpu có thể tắt nó trước khi thực hiện các chuỗi hướng dẫn quan trọng không được gián đoạn. Ngắt có thể che giấu được sử dụng bởi bộ điều khiển thiết bị để yêu cầu dịch vụ.

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

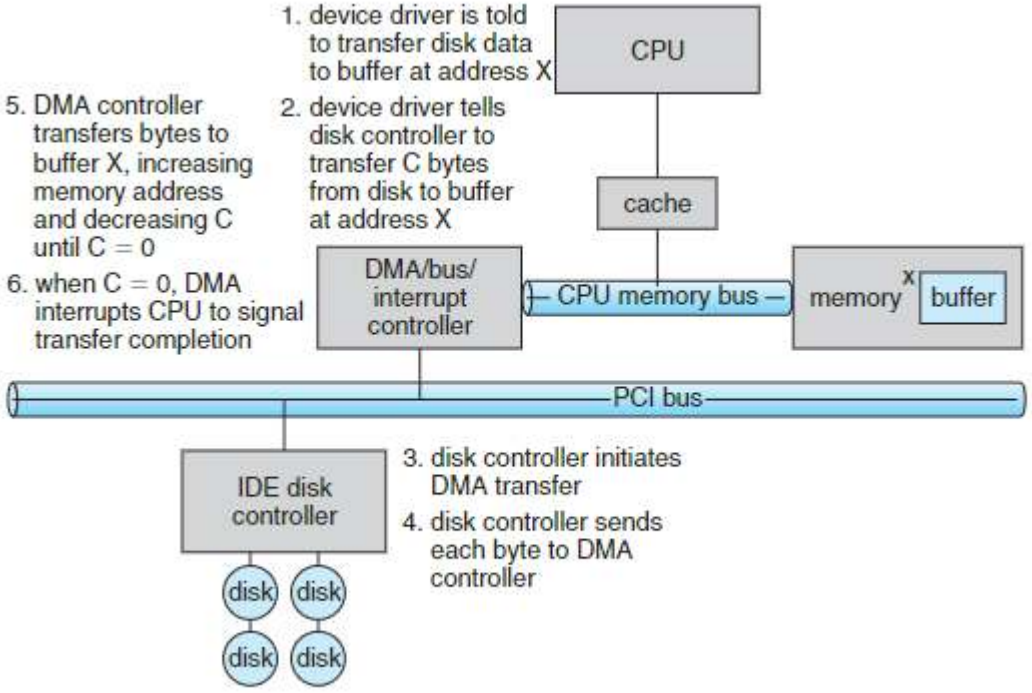
Intel Pentium processor event-vector table.

Một hệ điều hành có các ứng dụng tốt khác cho một cơ chế phần cứng và phần mềm hiệu quả giúp tiết kiệm một lượng nhỏ trạng thái bộ xử lý và sau đó gọi một thói quen đặc quyền trong hạt nhân. Ví dụ, nhiều hệ điều hành sử dụng cơ chế ngắt để phân trang bộ nhớ ảo. Lỗi trang là một ngoại lệ làm tăng sự gián đoạn. Ngắt làm đình chỉ quá trình hiện tại và nhảy đến bộ xử lý lỗi trang trong hạt nhân. Trình xử lý này lưu trạng thái của quy trình, di chuyển quy trình đến hàng đợi chờ, thực hiện quản lý bộ nhớ cache trang, lên lịch thao tác I/O để lấy trang, lên lịch một quy trình khác để tiếp tục thực hiện và sau đó trả về từ ngắt.

2.1.3 Direct Memory Access

Kết hợp giữa bộ điều khiển DMA và bộ điều khiển thiết bị được thực hiện thông qua một cặp dây gọi là DMA-request và DMA-acknowledge. Bộ điều khiển thiết bị đặt tín hiệu trên dây yêu cầu DMA khi có một từ dữ liệu để truyền. Tín hiệu này khiến bộ điều khiển DMA nắm bắt bus bộ nhớ, đặt địa chỉ mong muốn trên dây địa chỉ bộ nhớ và đặt tín hiệu trên dây thừa nhận DMA. Khi bộ điều khiển thiết bị nhận được tín hiệu thừa nhận DMA, nó sẽ chuyển từ dữ liệu vào bộ nhớ và loại bỏ tín hiệu yêu cầu DMA.
Khi toàn bộ quá trình chuyển giao kết thúc, bộ điều khiển DMA sẽ làm gián đoạn CPU. Quá trình này được mô tả trong Hình 13.5. Khi

bộ điều khiển DMA chiếm giữ bus bộ nhớ, CPU sẽ bị ngăn chặn trong giây lát truy cập vào bộ nhớ chính, mặc dù nó vẫn có thể truy cập các mục dữ liệu trong bộ nhớ cache chính và phụ của nó. Mặc dù việc đánh cắp chu kỳ này có thể làm chậm quá trình tính toán CPU, nhưng việc giảm tải công việc truyền dữ liệu sang bộ điều khiển DMA thường cải thiện tổng hiệu suất hệ thống. Một số kiến trúc máy tính sử dụng địa chỉ bộ nhớ vật lý cho DMA, nhưng những kiến trúc khác thực hiện truy cập bộ nhớ ảo trực tiếp (DVMA), sử dụng địa chỉ ảo trải qua dịch sang địa chỉ vật lý. DVMA có thể thực hiện chuyển giao giữa hai thiết bị ánh xạ bộ nhớ mà không cần sự can thiệp của CPU hoặc sử dụng bộ nhớ chính. Trên các hạt nhân chế độ được bảo vệ, hệ điều hành thường ngăn chặn các quy trình phát hành lệnh thiết bị trực tiếp. Kỷ luật này bảo vệ dữ liệu khỏi các vi phạm kiểm soát truy cập và cũng bảo vệ hệ thống khỏi việc sử dụng sai bộ điều khiển thiết bị có thể gây ra sự cố hệ thống. Thay vào đó, hệ điều hành xuất các chức năng mà một quy trình đủ đặc quyền có thể sử dụng để truy cập các hoạt động cấp thấp trên phần cứng cơ bản. Trên hạt nhân mà không cần bảo vệ bộ nhớ, các quy trình có thể truy cập trực tiếp vào bộ điều khiển thiết bị. Truy cập trực tiếp này có thể được sử dụng để đạt được hiệu suất cao, vì nó có thể tránh giao tiếp hạt nhân, chuyển đổi ngữ cảnh và các lớp phần mềm hạt nhân.



Hình 3. Các bước trong chuyển giao DMA.

2.1.4 Tóm tắt về phần cứng vào ra

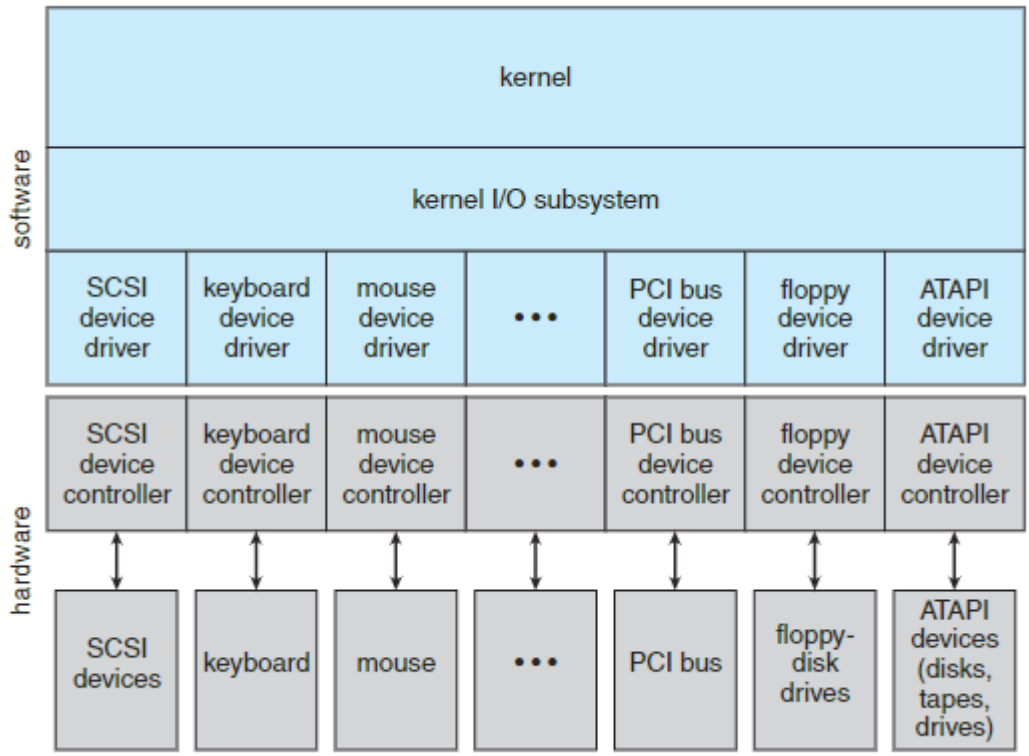
Mặc dù các khía cạnh phần cứng của I / O rất phức tạp khi được xem xét ở mức độ chi tiết của thiết kế phần cứng điện tử, các khái

niệm mà chúng tôi vừa mô tả là đủ để cho phép chúng tôi hiểu nhiều tính năng I / O của hệ điều hành. Chúng ta hãy xem xét các khái niệm chính:

- A bus.
- A controller
- An IO port and its registers
- Mối quan hệ bắt tay giữa máy chủ và bộ điều khiển thiết bị.
- Việc thực hiện cái bắt tay này trong một vòng bỏ phiếu hoặc thông qua ngắt.
- Việc giảm tải công việc này cho bộ điều khiển DMA để chuyển giao lớn.

2.2 Giao diện các ứng dụng vào ra

Giống như các vấn đề kỹ thuật phần mềm phức tạp khác, cách tiếp cận ở đây liên quan đến trừu tượng, đóng gói và phân lớp phần mềm. Cụ thể, chúng ta có thể trừu tượng hóa sự khác biệt chi tiết trong các thiết bị I / O bằng cách xác định một vài loại chung. Mỗi loại chung được truy cập thông qua một tập hợp các chức năng được tiêu chuẩn hóa - một giao diện. Sự khác biệt được gói gọn trong các mô-đun hạt nhân được gọi là trình điều khiển thiết bị nội bộ được tùy chỉnh cho các thiết bị cụ thể nhưng xuất một trong các giao diện tiêu chuẩn. Hình 13.6 minh họa cách các phần liên quan đến I / O của hạt nhân được cấu trúc trong các lớp phần mềm.
Mục đích của lớp trình điều khiển thiết bị là để che giấu sự khác biệt giữa các bộ điều khiển thiết bị từ hệ thống con I / O của hạt nhân, giống như các cuộc gọi hệ thống I / O gói gọn hành vi của các thiết bị trong một vài lớp chung ẩn sự khác biệt phần cứng từ các ứng dụng. Tạo hệ thống con I/O độc lập với phần cứng giúp đơn giản hóa công việc của nhà phát triển hệ điều hành. Nó cũng mang lại lợi ích cho các nhà sản xuất phần cứng. Họ hoặc thiết kế các thiết bị mới để tương thích với giao diện bộ điều khiển máy chủ hiện có (như SATA) hoặc họ viết trình điều khiển thiết bị để giao diện phần cứng mới cho các hệ điều hành phổ biến. Do đó, chúng ta có thể gắn các thiết bị ngoại vi mới vào máy tính mà không cần chờ nhà cung cấp hệ điều hành phát triển mã hỗ trợ.



Cấu trúc I/O hạt nhân

Với mục đích truy cập ứng dụng, nhiều điểm khác biệt trong số này được hệ điều hành ẩn và các thiết bị được nhóm thành một vài loại thông thường. Các kiểu truy cập thiết bị kết quả đã được tìm thấy là hữu ích và có thể áp dụng rộng rãi. Mặc dù các cuộc gọi hệ thống chính xác có thể khác nhau giữa các hệ điều hành, nhưng các danh mục thiết bị khá chuẩn. Các quy ước truy cập chính bao gồm I / O khối, I / O luồng ký tự, truy cập tệp ánh xạ bộ nhớ và ổ cắm mạng. Hệ điều hành cũng cung cấp các cuộc gọi hệ thống đặc biệt để truy cập một vài thiết bị bổ sung, chẳng hạn như đồng hồ thời gian trong ngày và bộ hẹn giờ. Một số hệ điều hành cung cấp một tập hợp các cuộc gọi hệ thống cho các thiết bị hiển thị đồ họa, video và âm thanh.

2.2.1 Thiết bị khối và ký tự

Giao diện thiết bị khối ghi lại tất cả các khía cạnh cần thiết để truy cập ổ đĩa và các thiết bị định hướng khối khác. Thiết bị dự kiến sẽ hiểu các lệnh như read() và write(). Nếu nó là một thiết bị truy cập ngẫu nhiên, nó cũng dự kiến sẽ có một lệnh seek() để chỉ định khối nào sẽ chuyển tiếp theo. Các ứng dụng thường truy cập một thiết bị như vậy thông qua giao diện hệ thống tệp. Chúng ta có thể thấy rằng read(), write() và seek() nắm bắt các hành vi thiết yếu của các thiết bị lưu trữ khối, để các ứng dụng được cách ly khỏi sự khác biệt cấp thấp giữa các thiết bị đó.

Bàn phím là một ví dụ về thiết bị được truy cập thông qua giao diện ký tự- luồng. Các lệnh gọi hệ thống cơ bản trong giao diện này cho phép một ứng dụng get() hoặc put() một ký tự. Trên đầu giao diện này, các thư viện có thể được xây dựng cung cấp quyền truy cập dòng tại một thời điểm, với các dịch vụ chỉnh sửa và lưu vào bộ đệm (ví dụ: khi người dùng nhập khoảng lùi, ký tự trước đó sẽ bị xóa khỏi luồng đầu vào). Kiểu truy cập này thuận tiện cho các thiết bị đầu vào như bàn phím, chuột và modem tạo ra dữ liệu cho đầu vào "một cách tự phát" —nghĩa là, vào những thời điểm mà ứng dụng không nhất thiết phải dự đoán được. Kiểu truy cập này cũng tốt cho các thiết bị đầu ra như máy in và bảng âm thanh, tự nhiên phù hợp với khái niệm về luồng byte tuyến tính.

2.2.2 Thiết bị mạng

Nhiều cách tiếp cận khác để giao tiếp giữa các bộ xử lý và truyền thông mạng đã được thực hiện. Ví dụ: Windows cung cấp một giao diện cho card giao diện mạng và giao diện thứ hai cho các giao thức mạng. Trong UNIX , có lịch sử lâu đời như một nền tảng chứng minh cho công nghệ mạng, chúng tôi tìm thấy các ống nửa song công, FIFE song công đầy đủ, LUỒNG song công đầy đủ, hàng đợi tin nhắn và ổ cắm. Thông tin về mạng UNIX được đưa ra trong Phần A.9.

2.2.3 Đồng hồ và bộ hẹn giờ

Hầu hết các máy tính đều có đồng hồ phần cứng và bộ hẹn giờ cung cấp ba chức năng cơ bản:

- Cho thời gian hiện tại.
- Cho thời gian trôi qua.
- Đặt hẹn giờ để kích hoạt hoạt động X tại thời điểm T.

Trên nhiều máy tính, tốc độ ngắt được tạo ra bởi đồng hồ phần cứng là từ 18 đến 60 tích tắc mỗi giây. Độ phân giải này là thô, vì một máy tính hiện đại có thể thực hiện hàng trăm triệu lệnh mỗi giây. Độ chính xác của trình kích hoạt bị giới hạn bởi độ phân giải thô của bộ hẹn giờ, cùng với chi phí duy trì đồng hồ ảo. Hơn nữa, nếu bọ ve hẹn giờ được sử dụng để duy trì đồng hồ thời gian trong ngày của hệ thống, đồng hồ hệ thống có thể trôi dạt. Trong hầu hết các máy tính, xung nhịp phần cứng được xây dựng từ một bộ đếm tần số cao. Trong một số máy tính, giá trị của bộ đếm này có thể được đọc từ thanh ghi thiết bị, trong trường hợp đó, bộ đếm có thể được coi là đồng hồ có độ phân giải cao. Mặc dù đồng hồ này không tạo ra ngắt, nhưng nó cung cấp các phép đo chính xác về khoảng thời gian.

2.2.4 Tóm tắt phần cứng I / O

Mặc dù các khía cạnh phần cứng của I / O rất phức tạp khi được xem xét ở mức độ chi tiết của thiết kế phần cứng điện tử, các khái niệm mà chúng tôi vừa mô tả là đủ để cho phép chúng tôi hiểu nhiều tính năng I / O của hệ điều hành. Hãy xem lại các khái niệm chính:

• Bus
• Bộ điều khiển
• Cổng I / O và các thanh ghi của nó
• Mối quan hệ bắt tay giữa máy chủ và bộ điều khiển thiết bị
• Việc thực hiện thao tác bắt tay này trong vòng lặp bỏ phiếu hoặc thông qua ngắt
• Việc giảm tải công việc này cho bộ điều khiển DMA để chuyển lớn.

Chúng tôi đã đưa ra một ví dụ cơ bản về việc bắt tay diễn ra giữa bộ điều khiển thiết bị và máy chủ trước đó trong phần này. Trên thực tế, sự đa dạng của các thiết bị có sẵn đặt ra một vấn đề cho những người triển khai hệ điều hành. Mỗi loại thiết bị có bộ khả năng, định nghĩa bit điều khiển và giao thức riêng để tương tác với máy chủ — và tất cả chúng đều khác nhau. Làm thế nào hệ điều hành có thể được thiết kế để chúng ta có thể gắn các thiết bị mới vào máy tính mà không cần viết lại hệ điều hành? Và khi các thiết bị khác nhau rất nhiều, làm thế nào hệ điều hành có thể cung cấp một giao diện I / O thuận tiện, thống nhất cho các ứng dụng? Chúng tôi giải quyết những câu hỏi đó tiếp theo.

2.3 Hệ thống con I/O hạt nhân

Kernels cung cấp nhiều dịch vụ liên quan đến I/O. Một số dịch vụ — lập lịch, lưu vào bộ đệm, bộ nhớ đệm, cuộn dây, đặt trước thiết bị và xử lý lỗi — được cung cấp bởi hệ thống con I / O của hạt nhân và xây dựng trên cơ sở hạ tầng phần cứng và trình điều khiển thiết bị. Hệ thống con I/O cũng chịu trách nhiệm tự bảo vệ mình khỏi các quy trình sai sót và người dùng độc hại.

2.3.1 Lập lịch I / O

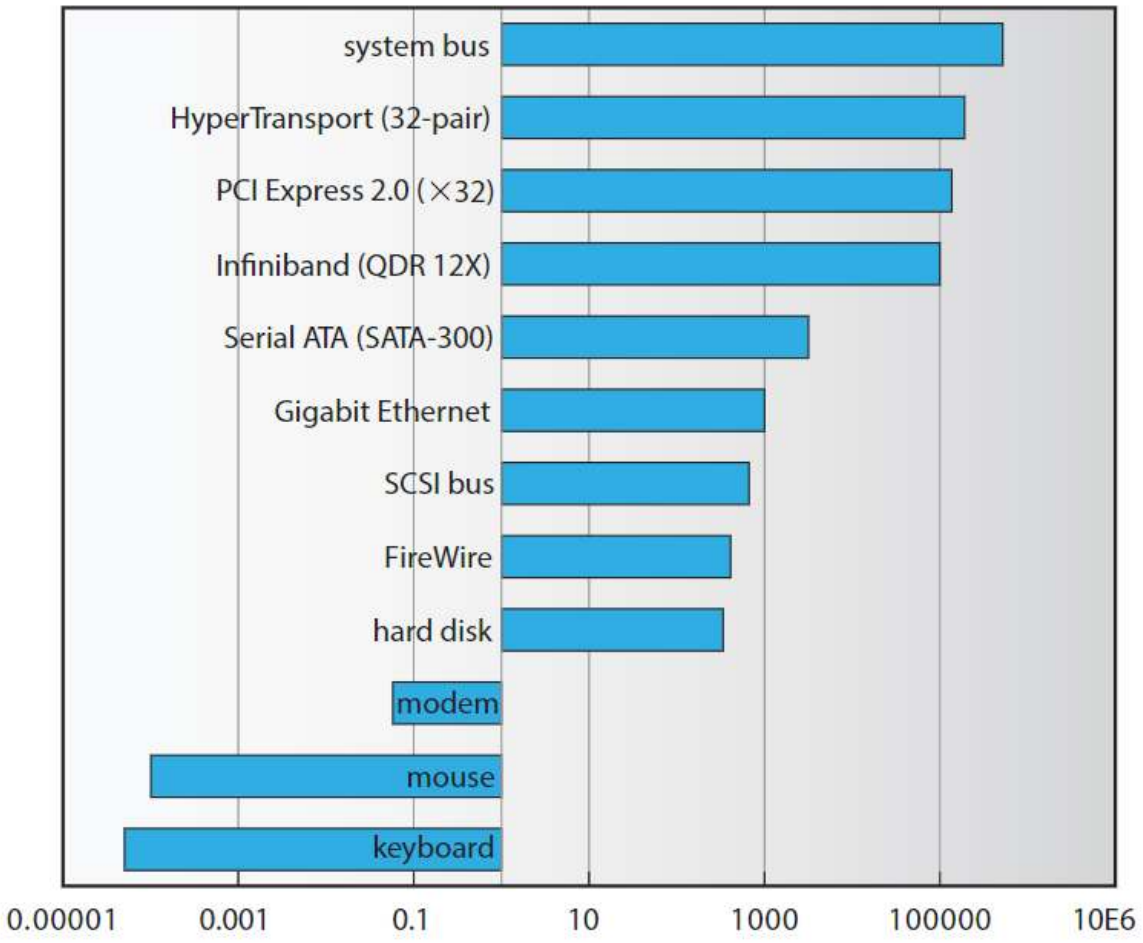
Để lên lịch cho một tập hợp các yêu cầu I / O có nghĩa là xác định một thứ tự tốt để thực hiện chúng. Thứ tự mà các ứng dụng phát hành các cuộc gọi hệ thống hiếm khi là sự lựa chọn tốt nhất. Lập lịch trình có thể cải thiện hiệu suất tổng thể của hệ thống, có thể chia sẻ quyền truy cập thiết bị một cách công bằng giữa các quy trình và có thể giảm thời gian chờ đợi trung bình để I / O hoàn thành. Đây là một ví dụ đơn giản để minh họa. Giả sử rằng một

cánh tay đĩa ở gần đầu đĩa và ba ứng dụng phát hành việc chặn các cuộc gọi đọc đến đĩa đó. Ứng dụng 1 yêu cầu một khối gần cuối đĩa, ứng dụng 2 yêu cầu một khối gần đầu và ứng dụng 3 yêu cầu một khối ở giữa đĩa. Hệ điều hành có thể giảm khoảng cách mà cánh tay đĩa di chuyển bằng cách phục vụ các ứng dụng theo thứ tự 2, 3, 1. Sắp xếp lại thứ tự dịch vụ theo cách này là bản chất của lập lịch I / O.
Các nhà phát triển hệ điều hành thực hiện lập lịch bằng cách duy trì hàng đợi các yêu cầu cho mỗi thiết bị. Khi một ứng dụng thực hiện cuộc gọi hệ thống I/O chặn, yêu cầu được đặt trên hàng đợi cho thiết bị đó. Bộ lập lịch I /O sắp xếp lại thứ tự của hàng đợi để cải thiện hiệu quả hệ thống tổng thể và thời gian phản hồi trung bình mà các ứng dụng gặp phải. Hệ điều hành cũng có thể cố gắng công bằng, để không một ứng dụng nào nhận được dịch vụ đặc biệt kém hoặc nó có thể cung cấp dịch vụ ưu tiên cho các yêu cầu nhạy cảm với sự chậm trễ. Ví dụ: các yêu cầu từ hệ thống con bộ nhớ ảo có thể được ưu tiên hơn các yêu cầu ứng dụng.

2.3.2 Đệm

Tất nhiên, bộ đệm là một vùng bộ nhớ lưu trữ dữ liệu được truyền giữa hai thiết bị hoặc giữa thiết bị và ứng dụng. Bộ đệm được thực hiện vì ba lý do.Một lý do là để đối phó với sự không phù hợp về tốc độ giữa nhà sản xuất và người tiêu dùng của luồng dữ liệu. Giả sử, ví dụ, một tệp đang được nhận qua modem để lưu trữ trên đĩa cứng. Modem chậm hơn khoảng một nghìn lần so với đĩa cứng. Vì vậy, một bộ đệm được tạo trong bộ nhớ chính để tích lũy các byte nhận được từ modem. Khi toàn bộ bộ đệm dữ liệu đã đến, bộ đệm có thể được ghi vào đĩa trong một thao tác duy nhất. Vì ghi đĩa không phải là tức thời và modem vẫn cần một nơi để lưu trữ dữ liệu đến bổ sung, hai bộ đệm được sử dụng.
Bộ đệm kép này tách rời nhà sản xuất dữ liệu từ người tiêu dùng, do đó nới lỏng các yêu cầu về thời gian giữa chúng. Sự cần thiết của việc tách rời này được minh họa trong Hình 3.10, trong đó liệt kê sự khác biệt rất lớn về tốc độ thiết bị đối với phần cứng máy

tính điển hình.



Hình 13.10 Tỷ lệ truyền thiết bị của Sun Enterprise 6000 (logarit)

2.3.3 Caching

Bộ nhớ cache là một vùng bộ nhớ nhanh chứa các bản sao dữ liệu. Truy cập vào bản sao được lưu trong bộ nhớ cache hiệu quả hơn quyền truy cập vào bản gốc.
Bộ nhớ đệm và bộ đệm là các chức năng riêng biệt, nhưng đôi khi một vùng bộ nhớ có thể được sử dụng cho cả hai mục đích. Ví dụ: để bảo tồn ngữ nghĩa sao chép và cho phép lập lịch hiệu quả I / O đĩa, hệ điều hành sử dụng bộ đệm trong bộ nhớ chính để giữ dữ liệu đĩa. Các bộ đệm này cũng được sử dụng như một bộ đệm ẩn, để cải thiện hiệu quả I / O cho các tệp được chia sẻ bởi các ứng dụng hoặc đang được ghi và đọc lại nhanh chóng. Khi hạt nhân nhận được yêu cầu I/O tệp, trước tiên hạt nhân sẽ truy cập vào bộ đệm đệm để xem liệu vùng đó của tệp đã có sẵn trong bộ nhớ chính hay chưa. Nếu có, I / O đĩa vật lý có thể tránh được hoặc trì hoãn.

Ngoài ra, ghi đĩa được tích lũy trong bộ đệm đệm trong vài giây, do đó các lần chuyển lớn được thu thập để cho phép lịch ghi hiệu quả.

2.3.4 Caching

Spool chỉ là bộ đệm chứa đầu ra cho thiết bị, chẳng hạn như máy in, không thể chấp nhận các luồng dữ liệu xen kẽ. Mặc dù máy in chỉ có thể phục vụ một công việc tại một thời điểm, một số ứng dụng có thể muốn in đầu ra của chúng đồng thời mà không trộn lẫn đầu ra của chúng với nhau. Hệ điều hành giải quyết vấn đề này bằng cách chặn tất cả đầu ra cho máy in. Đầu ra của mỗi ứng dụng được cuộn vào một tệp đĩa riêng biệt. Khi một ứng dụng hoàn tất quá trình in, hệ thống ống chỉ sẽ xếp hàng đợi tệp ống chỉ tương ứng để xuất ra máy in. Hệ thống ống chỉ sao chép các tệp ống chỉ được xếp hàng đợi vào từng tệp máy in một. Trong một số hệ điều hành, spooling được quản lý bởi một quy trình daemon hệ thống. Trong những người khác, nó được xử lý bởi một chủ đề trong hạt nhân. Trong cả hai trường hợp, hệ điều hành cung cấp một giao diện điều khiển cho phép người dùng và quản trị viên hệ thống hiển thị hàng đợi, xóa các tác vụ không mong muốn trước khi các tác vụ đó được in, tạm dừng in trong khi máy in được bảo dưỡng, v.v.
Một số thiết bị, chẳng hạn như ổ đĩa băng và máy in, không thể ghép các yêu cầu I/O của nhiều ứng dụng đồng thời một cách hữu ích. Spooling là một cách các hệ điều hành có thể phối hợp đầu ra đồng thời. Một cách khác để đối phó với quyền truy cập thiết bị đồng thời là cung cấp các phương tiện rõ ràng để phối hợp. Một số hệ điều hành (bao gồm vms) cung cấp hỗ trợ cho quyền truy cập thiết bị độc quyền bằng cách cho phép một quy trình để phân bổ một thiết bị nhàn rỗi và để phân bổ thiết bị đó khi nó không còn cần thiết nữa. Các hệ điều hành khác thực thi giới hạn của một xử lý tệp mở cho một thiết bị như vậy. Nhiều hệ điều hành cung cấp các chức năng cho phép các quy trình phối hợp quyền truy cập độc quyền giữa chúng. Ví dụ: Windows cung cấp các cuộc gọi hệ thống để đợi cho đến khi một đối tượng thiết bị khả dụng. Nó cũng có một tham số cho lệnh gọi hệ thống OpenFile() khai báo các loại truy cập được phép cho các luồng đồng thời khác. Trên các hệ thống này, tùy thuộc vào các ứng dụng để tránh bế tắc.

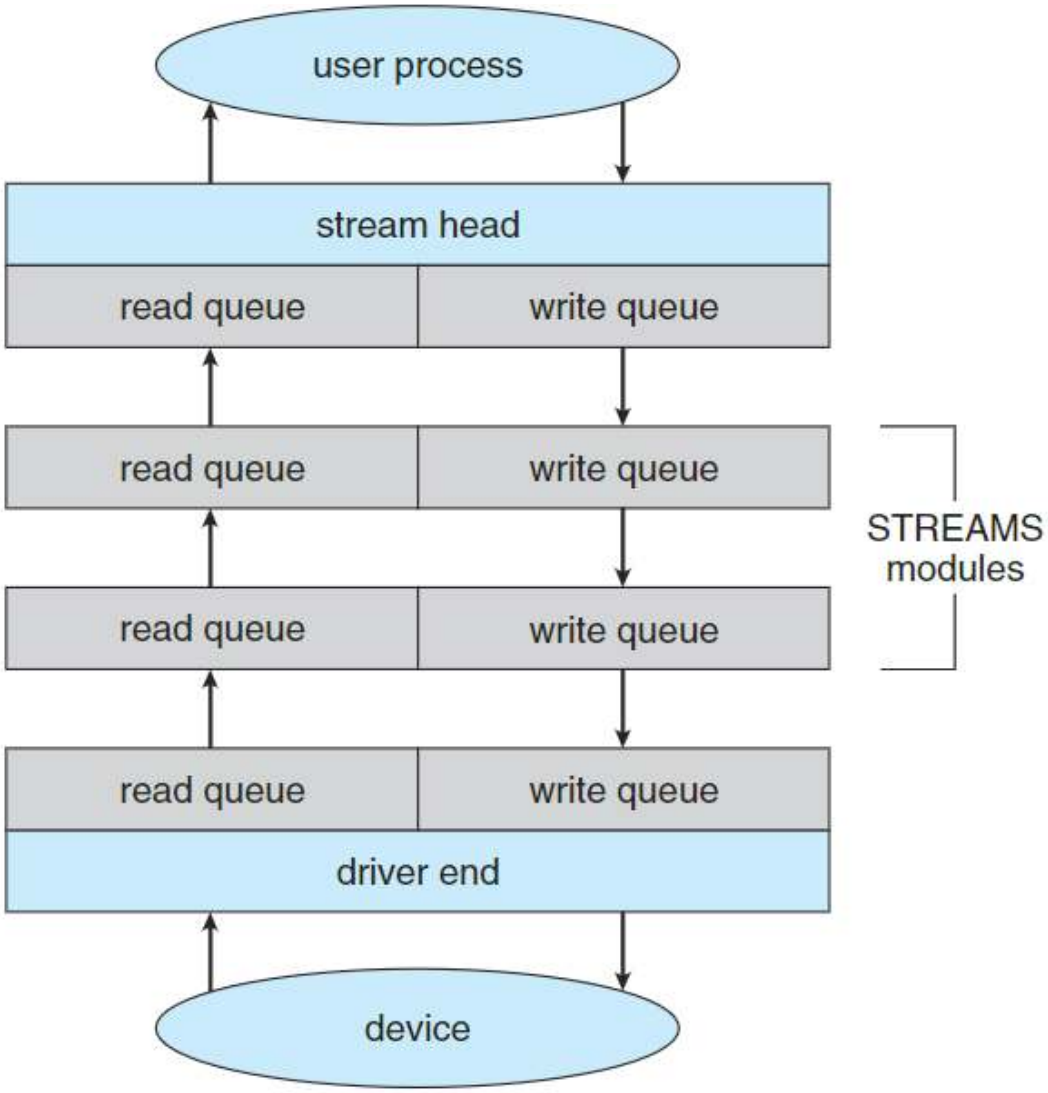
2.4 Chuyển đổi yêu câu I/O sang hoạt động phân cứng

Trước đó, chúng tôi đã mô tả việc bắt tay giữa trình điều khiển thiết bị và bộ điều khiển thiết bị, nhưng chúng tôi không giải thích cách hệ điều hành kết nối yêu cầu ứng dụng với một bộ dây

mạng hoặc với một khu vực đĩa cụ thể. Hãy xem xét, ví dụ, đọc một tập tin từ đĩa. Ứng dụng đề cập đến dữ liệu bằng tên tệp. Trong một đĩa, hệ thống tệp ánh xạ từ tên tệp thông qua các thư mục hệ thống tệp để có được sự phân bổ không gian của tệp. Ví dụ: trong MS-DOS , tên ánh xạ đến một số cho biết một mục nhập trong bảng truy cập tệp và mục nhập bảng đó cho biết khối đĩa nào được phân bổ cho tệp. Trong UNIX , tên ánh xạ đến một số inode và inode tương ứng chứa thông tin phân bổ không gian. Nhưng làm thế nào là kết nối được thực hiện từ tên tệp với bộ điều khiển đĩa (địa chỉ cổng phần cứng hoặc bộ điều khiển ánh xạ bộ nhớ đăng ký)?
Một phương pháp được sử dụng bởi MS-DOS , một hệ điều hành tương đối đơn giản. Phần đầu tiên của tên tệp MS-DOS, trước dấu hai chấm, là một chuỗi xác định một thiết bị phần cứng cụ thể. Ví dụ: C: là phần đầu tiên của mọi tên tệp trên đĩa cứng chính. Thực tế là C: đại diện cho đĩa cứng chính được tích hợp vào hệ điều hành; C: được ánh xạ đến một địa chỉ cổng cụ thể thông qua bảng thiết bị. Do dấu hai chấm phân tách, không gian tên thiết bị tách biệt với không gian tên hệ thống tệp. Sự tách biệt này giúp hệ điều hành dễ dàng liên kết chức năng bổ sung với từng thiết bị. Ví dụ, thật dễ dàng để gọi cuộn trên bất kỳ tệp nào được ghi vào máy in.

2.5 Streams

UNIX System V có một cơ chế thú vị, được gọi là STREAMS, cho phép một ứng dụng lắp ráp các đường ống của mã trình điều khiển một cách linh hoạt. Luồng là kết nối song công hoàn toàn giữa trình điều khiển thiết bị và quy trình cấp người dùng. Nó bao gồm một đầu luồng giao tiếp với quy trình người dùng, một đầu trình điều khiển điều khiển thiết bị và không có hoặc nhiều mô-đun luồng giữa đầu luồng và đầu trình điều khiển. Mỗi thành phần này chứa một cặp hàng đợi —một hàng đợi đọc và một hàng đợi ghi. Truyền tin nhắn được sử dụng để truyền dữ liệu giữa các hàng đợi. Cấu trúc STREAMS được thể hiện trong Hình 3.11.



Hình 3.11 Cấu trúc STREAMS.

Lợi ích của việc sử dụng STREAMS là nó cung cấp một khuôn khổ cho cách tiếp cận mô-đun và gia tăng để viết trình điều khiển thiết bị và giao thức mạng. Các mô-đun có thể được sử dụng bởi các luồng khác nhau và do đó bởi các thiết bị khác nhau. Ví dụ: một mô-đun mạng có thể được sử dụng bởi cả card mạng Ethernet và card mạng không dây 802.11. Hơn nữa, thay vì coi I / O của thiết bị ký tự là một luồng byte phi cấu trúc, STREAMS cho phép hỗ trợ ranh giới tin nhắn và kiểm soát thông tin khi giao tiếp giữa các mô-đun. Hầu hết các biến thể UNIX đều hỗ trợ STREAMS và đây là phương pháp ưa thích để viết các giao thức và trình điều khiển thiết bị. Ví dụ, Hệ thống V UNIX và Solaris thực hiện cơ chế socket bằng STREAMS.

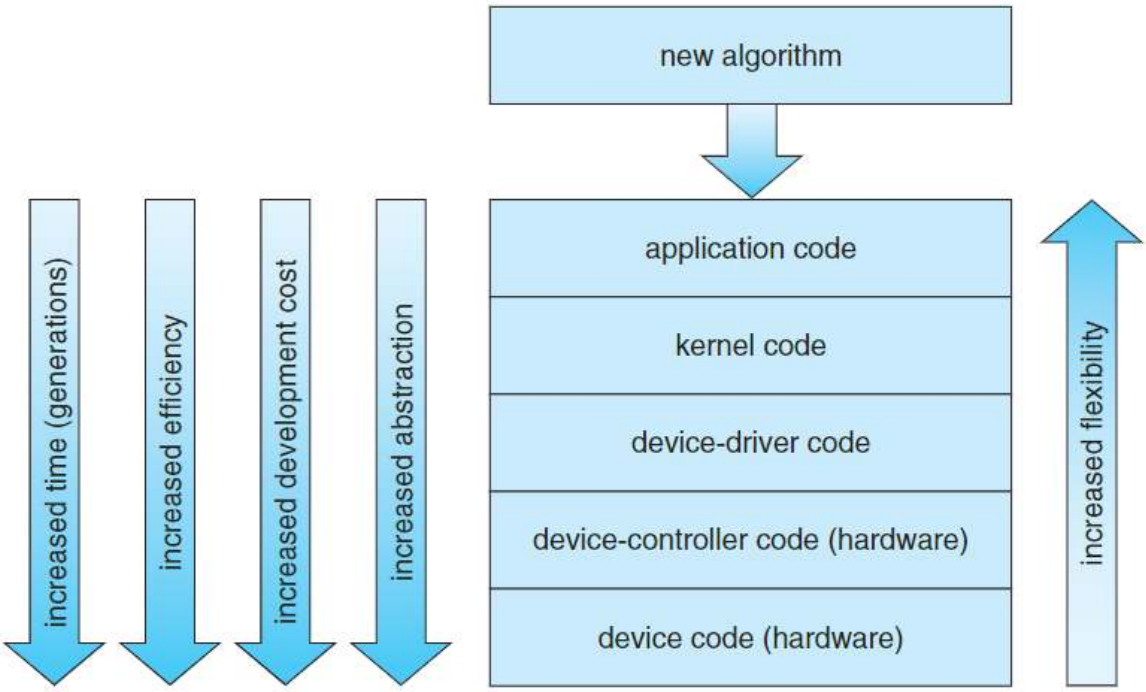
2.6 Performance

I /O là một yếu tố chính trong hiệu suất hệ thống. Nó đặt ra yêu cầu lớn đối với CPU để thực thi mã trình điều khiển thiết bị và lên lịch cho các quy trình một cách công bằng và hiệu quả khi chúng

chặn và bỏ chặn. Kết quả là chuyển đổi ngữ cảnh gây căng thẳng cho CPU và bộ nhớ đệm phần cứng của nó. I/O cũng bộc lộ bất kỳ sự kém hiệu quả nào trong các cơ chế xử lý ngắt trong hạt nhân. Ngoài ra, I/O tải xuống bus bộ nhớ trong quá trình sao chép dữ liệu giữa bộ điều khiển và bộ nhớ vật lý và một lần nữa trong quá trình sao chép giữa bộ đệm hạt nhân và không gian dữ liệu ứng dụng. Đối phó một cách duyên dáng với tất cả những nhu cầu này là một trong những mối quan tâm chính của một kiến trúc sư máy tính.

Các thiết bị I /O khác nhau rất nhiều về độ phức tạp. Ví dụ, một con chuột rất đơn giản. Các chuyển động chuột và nhấp chuột vào nút được chuyển đổi thành các giá trị số được truyền từ phần cứng, thông qua trình điều khiển thiết bị chuột, đến ứng dụng. Ngược lại, chức năng được cung cấp bởi trình điều khiển thiết bị đĩa Windows rất phức tạp. Nó không chỉ quản lý các đĩa riêng lẻ mà còn thực hiện các mảng RAID (Phần 10.7). Để làm như vậy, nó chuyển đổi yêu cầu đọc hoặc ghi của ứng dụng thành một tập hợp các hoạt động I / O đĩa phối hợp. Hơn nữa, nó thực hiện các thuật toán xử lý lỗi và khôi phục dữ liệu phức tạp và thực hiện nhiều bước để tối ưu hóa hiệu suất đĩa.

Chức năng I/O nên được triển khai ở đâu —trong phần cứng của thiết bị, trong trình điều khiển thiết bị hoặc trong phần mềm ứng dụng? Đôi khi chúng ta quan sát sự tiến triển được mô tả trong Hình 3.12.



Hình 3.12 Tiến trình chức năng của thiết bị

2.7 Tóm tắt

Các yếu tố phần cứng cơ bản liên quan đến I / O là bus, bộ điều khiển thiết bị và chính thiết bị. Công việc di chuyển dữ liệu giữa các thiết bị và bộ nhớ chính được CPU thực hiện dưới dạng I / O được lập trình hoặc được giảm tải cho bộ điều khiển DMA. Mô-đun hạt nhân điều khiển thiết bị là trình điều khiển thiết bị. Giao diện cuộc gọi hệ thống được cung cấp cho các ứng dụng được thiết kế để xử lý một số loại phần cứng cơ bản, bao gồm thiết bị khối, thiết bị ký tự, tệp ánh xạ bộ nhớ, ổ cắm mạng và bộ hẹn giờ khoảng thời gian được lập trình. Các cuộc gọi hệ thống thường chặn các quy trình phát hành chúng, nhưng các cuộc gọi không chặn và không đồng bộ được sử dụng bởi chính hạt nhân và bởi các ứng dụng không được ngủ trong khi chờ thao tác I / O hoàn tất.

Hệ thống con I/O của kernel cung cấp nhiều dịch vụ. Trong số này có lập lịch I / O, bộ đệm, bộ nhớ đệm, bộ đệm, đặt trước thiết bị và xử lý lỗi. Một dịch vụ khác, dịch tên, thực hiện các kết nối giữa các thiết bị phần cứng và tên tệp tượng trưng được sử dụng bởi các ứng dụng. Nó liên quan đến một số cấp độ ánh xạ dịch từ tên chuỗi ký tự, đến trình điều khiển thiết bị và địa chỉ thiết bị cụ thể, sau đó đến địa chỉ vật lý của cổng I / O hoặc bộ điều khiển bus. Ánh xạ này có thể xảy ra trong không gian tên hệ thống tệp, như trong UNIX hoặc trong một không gian tên thiết bị riêng biệt, như trong MS-DOS.

Các cuộc gọi hệ thống I / O rất tốn kém về mức tiêu thụ CPU vì có nhiều lớp phần mềm giữa một thiết bị vật lý và một ứng dụng. Các lớp này ngụ ý chi phí từ một số nguồn: chuyển đổi ngữ cảnh để vượt qua ranh giới bảo vệ của hạt nhân, xử lý tín hiệu và ngắt để phục vụ các thiết bị I / O và tải trên CPU và hệ thống bộ nhớ để sao chép dữ liệu giữa bộ đệm hạt nhân và không gian ứng dụng.

Chương 3. Kết luận và hướng phát triển

Chương trình được phát triển dựa trên các kiến thức đã được giảng dạy trên lớp và tự tìm hiểu nên còn sơ sài, tồn tại nhiều thiếu sót và hạn chế, một phần cũng do bản thân em chưa có kinh nghiệm xây dựng và phát triển nhiều chương trình trên Bài tập lớn. Sau quá trình phát triển và thử nghiệm, em xin đưa ra kết luận và hướng phát triển như sau:

1. Ưu điểm

- Đáp ứng, hoàn thành đầy đủ các yêu cầu của đề tài về cả lý thuyết và ứng dụng.

2. Nhược điểm

- Còn nhiều thiếu sót

3. Hướng phát triển

Với những ưu nhược điểm như trên, cá nhân em có một vài ý tuưởng phát triển mới đó là:

- Sửa các lỗi sẵn có.

DANH MỤC TÀI LIỆU THAM KHẢO

1. Bài giảng học phần Nguyên lý Hệ điều hành của thầy Đỗ Tuấn Anh.

2. Nguyên lý Hệ điều hành (NXB Giáo Dục Việt Nam) – Hồ Đắc Phương.

3. Operating System Concepts – Abraham Siberschatz.