



SỞ GIÁO DỤC VÀ ĐÀO TẠO HÀ NỘI

GIÁO TRÌNH

Nguyên lý hệ điều hành

DÙNG TRONG CÁC TRƯỜNG TRUNG HỌC CHUYÊN NGHIỆP



NHÀ XUẤT BẢN HÀ NỘI

SỞ GIÁO DỤC VÀ ĐÀO TẠO HÀ NỘI

ĐẶNG VŨ TÙNG

**GIÁO TRÌNH
NGUYÊN LÝ HỆ ĐIỀU HÀNH**

(Dùng trong các trường THCN)

NHÀ XUẤT BẢN HÀ NỘI - 2005

Mã số xuất bản: $\frac{373 - 373.7}{\text{HN} - 05}$ 113/407 - 05

Lời giới thiệu

Nước ta đang bước vào thời kỳ công nghiệp hóa, hiện đại hóa nhằm đưa Việt Nam trở thành nước công nghiệp văn minh, hiện đại.

Trong sự nghiệp cách mạng to lớn đó, công tác đào tạo nhân lực luôn giữ vai trò quan trọng. Báo cáo Chính trị của Ban Chấp hành Trung ương Đảng Cộng sản Việt Nam tại Đại hội Đảng toàn quốc lần thứ IX đã chỉ rõ: “Phát triển giáo dục và đào tạo là một trong những động lực quan trọng thúc đẩy sự nghiệp công nghiệp hóa, hiện đại hóa, là điều kiện để phát triển nguồn lực con người - yếu tố cơ bản để phát triển xã hội, tăng trưởng kinh tế nhanh và bền vững”.

Quán triệt chủ trương, Nghị quyết của Đảng và Nhà nước và nhận thức đúng đắn về tầm quan trọng của chương trình, giáo trình đối với việc nâng cao chất lượng đào tạo, theo đề nghị của Sở Giáo dục và Đào tạo Hà Nội, ngày 23/9/2003, Ủy ban nhân dân thành phố Hà Nội đã ra Quyết định số 5620/QĐ-UB cho phép Sở Giáo dục và Đào tạo thực hiện đề án biên soạn chương trình, giáo trình trong các trường Trung học chuyên nghiệp (THCN) Hà Nội. Quyết định này thể hiện sự quan tâm sâu sắc của Thành ủy, UBND thành phố trong việc nâng cao chất lượng đào tạo và phát triển nguồn nhân lực Thủ đô.

Trên cơ sở chương trình khung của Bộ Giáo dục và Đào tạo ban hành và những kinh nghiệm rút ra từ thực tế đào tạo, Sở Giáo dục và Đào tạo đã chỉ đạo các trường THCN tổ chức biên soạn chương trình, giáo trình một cách khoa học, hệ

thống và cập nhật những kiến thức thực tiễn phù hợp với đối tượng học sinh THCN Hà Nội.

Bộ giáo trình này là tài liệu giảng dạy và học tập trong các trường THCN ở Hà Nội, đồng thời là tài liệu tham khảo hữu ích cho các trường có đào tạo các ngành kỹ thuật - nghiệp vụ và động đảo bạn đọc quan tâm đến vấn đề hướng nghiệp, dạy nghề.

Việc tổ chức biên soạn bộ chương trình, giáo trình này là một trong nhiều hoạt động thiết thực của ngành giáo dục và đào tạo Thủ đô để kỷ niệm “50 năm giải phóng Thủ đô”, “50 năm thành lập ngành” và hướng tới kỷ niệm “1000 năm Thăng Long - Hà Nội”.

Sở Giáo dục và Đào tạo Hà Nội chân thành cảm ơn Thành ủy, UBND, các sở, ban, ngành của Thành phố, Vụ Giáo dục chuyên nghiệp Bộ Giáo dục và Đào tạo, các nhà khoa học, các chuyên gia đầu ngành, các giảng viên, các nhà quản lý, các nhà doanh nghiệp đã tạo điều kiện giúp đỡ, đóng góp ý kiến, tham gia Hội đồng phản biện, Hội đồng thẩm định và Hội đồng nghiệm thu các chương trình, giáo trình.

Đây là lần đầu tiên Sở Giáo dục và Đào tạo Hà Nội tổ chức biên soạn chương trình, giáo trình. Dù đã hết sức cố gắng nhưng chắc chắn không tránh khỏi thiếu sót, bất cập. Chúng tôi mong nhận được những ý kiến đóng góp của bạn đọc để từng bước hoàn thiện bộ giáo trình trong các lần tái bản sau.

GIÁM ĐỐC SỞ GIÁO DỤC VÀ ĐÀO TẠO

Lời nói đầu

Trong hệ thống kiến thức chuyên ngành trang bị cho học sinh khoa Kỹ thuật máy tính và khoa Tin học quản lý, giáo trình Nguyên lý hệ điều hành góp phần cung cấp những nội dung kiến thức chung nhất về hệ điều hành máy tính. Nó giúp người học nắm bắt được những nguyên lý cơ bản và nguyên tắc làm việc của một hệ điều hành máy tính tổng quát. Từ đó có thể áp dụng để làm việc tốt với các hệ điều hành cụ thể trên thực tế, hình dung được xu hướng phát triển của các hệ điều hành mới trong tương lai.

Để phục vụ công tác giảng dạy và học tập, chúng tôi biên soạn cuốn giáo trình Nguyên lý hệ điều hành nhằm cung cấp tới người học một số kiến thức cơ bản nhất về lĩnh vực này.

Chúng tôi xin trân trọng cảm ơn TS. Phó Đức Toàn - Học viện Công nghệ Bưu chính viễn thông và TS. Lê Bá Dũng - Viện Công nghệ thông tin đã đọc và cho những nhận xét về nội dung giáo trình. Chúng tôi xin trân trọng cảm ơn PGS. TS. Thái Quang Vinh - Viện Công nghệ thông tin, TS. Lương Cao Đông - Phó chủ nhiệm khoa Công nghệ thông tin - Viện Đại học mở Hà Nội, TS. Nguyễn Văn Điện - Chủ nhiệm khoa Công nghệ thông tin trường Cao đẳng Điện lực đã cho những ý kiến đóng góp quý báu để giáo trình được hoàn thiện hơn.

Chúng tôi xin gửi lời cảm ơn tới Sở Giáo dục và Đào tạo Hà Nội, Ban giám hiệu trường Trung học bán công kỹ thuật tin học Hà Nội - ESTIH đã tạo điều kiện cho chúng tôi hoàn thành cuốn giáo trình này.

Mặc dù đã có nhiều cố gắng trong biên soạn nhưng do kiến thức và thời gian có hạn, đồng thời cũng là lần biên soạn đầu tiên nên giáo trình không tránh khỏi những thiếu sót. Chúng tôi mong muốn nhận được sự thông cảm và ý kiến đóng góp của các thầy cô, các bạn học sinh, sinh viên và bạn đọc để cuốn giáo trình được tốt hơn.

TÁC GIẢ

Bài mở đầu

Máy tính điện tử ra đời vào những năm 40 của thế kỷ XX. Ban đầu, phạm vi sử dụng máy tính còn rất hạn hẹp, đa phần chỉ nhằm phục vụ mục đích nghiên cứu khoa học. Hơn nữa, để vận hành hệ thống cần phải sử dụng các công cụ phần cứng đặc biệt và thao tác vận hành rất phức tạp.

Cùng phát triển song song với sự phát triển của kỹ thuật điện tử, các thế hệ máy tính về sau được cải tiến ngày một tinh vi hơn, có tốc độ xử lý nhanh hơn, kích thước nhỏ gọn hơn, tiêu tốn ít năng lượng hơn và đã làm nên một cuộc cách mạng trong lĩnh vực xử lý, tính toán, điều khiển tự động... Với các thế hệ máy tính này đòi hỏi phải có sự điều khiển, vận hành một cách tự động để phát huy hiệu quả của nó một cách tối ưu nhất. Như vậy, cần phải có một chương trình phần mềm đảm bảo việc giải quyết các vấn đề nói trên. Đó chính là các hệ điều hành máy tính.

Hệ điều hành là một tập hợp các phần mềm hệ thống điều khiển mọi hoạt động của máy tính và tạo môi trường giao diện giữa người sử dụng và máy tính. Vì vậy, hệ điều hành rất quen thuộc với mọi người sử dụng, tất cả mọi người sử dụng khi làm việc với máy tính cần phải biết thao tác (dù là những thao tác đơn giản) với một hệ điều hành cụ thể.

Hiểu biết, nắm vững nguyên lý hoạt động của hệ điều hành để từ đó có thể khai thác các hệ điều hành cụ thể một cách có hiệu quả là một nhu cầu thực tế không thể thiếu được đối với các cán bộ phụ trách kỹ thuật và quản lý các hệ thống máy tính.

Nội dung cuốn giáo trình “Nguyên lý hệ điều hành” gồm các kiến thức về phần mềm hệ thống. Nó cung cấp cho người học những kiến thức chung nhất về hệ điều hành máy tính, giúp người học nắm bắt được những nguyên lý cơ bản và nguyên tắc làm việc của một hệ điều hành máy tính tổng quát, từ đó áp dụng để làm việc tốt với các hệ điều hành cụ thể trên thực tế, hiểu và xử lý được các vấn đề có thể xảy ra trong hệ thống. Đồng thời nắm bắt được xu hướng phát triển của các hệ điều hành mới trong tương lai.

Các vấn đề cụ thể trình bày trong giáo trình này được chia thành 9 chương:

Chương 1. Tổng quan về hệ điều hành: Cung cấp các kiến thức tổng quát về hệ điều hành, quá trình phát triển và phân loại hệ điều hành; các tính chất cơ bản của hệ điều hành và nguyên tắc thiết kế, xây dựng hệ điều hành. Trong chương này cũng trình bày về cấu trúc, các thành phần cơ bản và các phục vụ của hệ điều hành.

Chương 2. Quản lý tiến trình: Trong hệ thống luôn tồn tại các tiến trình hoạt động song song mà trạng thái của chúng ảnh hưởng lẫn nhau và ảnh hưởng tới hoạt động của toàn hệ thống. Mục tiêu của chương này nhằm giới thiệu biện pháp quản lý các tiến trình song hành với nội dung chính là giải quyết bài toán tranh chấp tài nguyên giữa các tiến trình - Bài toán đoạn tối hạn.

Trong chương này cũng đề cập tới hiện tượng bế tắc. Đó là tình trạng các tiến trình trong hệ thống rơi vào trạng thái chờ đợi vô hạn dẫn tới làm “treo” hệ thống.

Chương 3. Lập lịch cho CPU: CPU là tài nguyên quan trọng nhất của hệ thống, nó thể hiện sức mạnh xử lý của toàn bộ hệ thống. Do vậy, thời gian mà CPU phục vụ cho các tiến trình hoạt động cần phải được khai thác một cách tối ưu nhất. Chương 3 nêu rõ tầm quan trọng của “giờ CPU” và trình bày các phương pháp, các thuật toán lập lịch cho CPU của hệ điều hành nhằm đáp ứng yêu cầu được phục vụ của các tiến trình. Đồng thời, trong chương này cũng giới thiệu về ngắt - một công cụ để CPU có thể điều khiển hoạt động của các tiến trình một cách chính xác.

Chương 4. Quản lý bộ nhớ trong: Nhiệm vụ của hệ điều hành là phải cấp phát không gian nhớ cho các chương trình hoạt động và thu hồi khi chương trình kết thúc. Chương này giải thích quá trình gán địa chỉ (cấp phát không gian nhớ) cho các biến khi thực hiện một chương trình, các cấu trúc cơ bản của chương trình phần mềm và các phương pháp cấp phát bộ nhớ của hệ điều hành.

Chương 5. Quản lý bộ nhớ ngoài: Bộ nhớ ngoài là thiết bị dùng để lưu trữ thông tin trước và sau quá trình xử lý, tính toán. Trong chương này trình bày các biện pháp quản lý và cấp phát không gian nhớ tự do trên đĩa từ, các thuật toán lập lịch cho đĩa từ và nguyên tắc quản lý thông tin trên bộ nhớ phụ - Hệ File.

Chương 6. Quản lý thiết bị: Ngoài các thiết bị chuẩn mang tính chất bắt buộc, các hệ thống máy tính cần phải có khả năng kết nối với một số lượng tùy ý các thiết bị mở rộng. Trong chương này trình bày nguyên tắc tổ chức và quản lý thiết bị ngoại vi của hệ điều hành, đồng thời đề cập tới một số kỹ thuật áp

dụng trong quản lý thiết bị ngoại vi.

Chương 7. Bảo vệ và an toàn hệ thống: Bảo vệ hệ thống tránh khỏi sự can thiệp bất hợp pháp từ bên ngoài cũng như các nguyên nhân tiềm ẩn bên trong là một vấn đề cực kỳ phức tạp. Chương này nêu rõ mục đích của việc bảo vệ hệ thống, đồng thời trình bày một số biện pháp bảo vệ và an toàn hệ thống.

Chương 8. Hệ điều hành đa xử lý: Có hai phương pháp để tổ chức các hệ thống đa xử lý đó là hệ nhiều CPU (tập hợp các CPU trong một máy tính duy nhất) và hệ phân tán (thực chất là các mạng máy tính). Chương này trình bày mục đích và cấu trúc của hệ nhiều CPU và hệ phân tán. Đồng thời nêu sơ lược một số vấn đề về quản lý tài nguyên, truyền thông tin, xử lý và truy nhập thông tin trong các hệ thống này.

Chương 9. Hệ điều hành DOS: Chương này trình bày một số vấn đề về hệ điều hành DOS - hệ điều hành đã đặt một mốc rất quan trọng trong quá trình phát triển của máy tính và các hệ điều hành. Nó được sử dụng trong một thời gian tương đối dài cho các máy tính PC và là một hệ điều hành cơ bản nhất. Các vấn đề trình bày mang tính áp dụng thực tế của các kiến thức đã học trong các chương trước như: quản lý tiến trình, quản lý bộ nhớ, quản lý đĩa từ, quản lý thiết bị, v.v.

Cuối cùng, cần xác định nội dung của giáo trình này về cơ bản là định hướng tới hệ điều hành truyền thống. Một số nét sơ bộ về các hệ điều hành đa xử lý được trình bày chỉ nhằm mục đích định hướng cho người học tìm hiểu thêm các tài liệu bổ sung, nhằm hoàn thiện kiến thức chung về hệ điều hành.

Chương 1

TỔNG QUAN VỀ HỆ ĐIỀU HÀNH

Mục tiêu

Sau chương này, người học có thể nắm bắt được các kiến thức tổng quát về hệ điều hành, quá trình phát triển và phân loại hệ điều hành; các tính chất cơ bản của hệ điều hành và nguyên tắc thiết kế xây dựng hệ điều hành. Hiểu được cấu trúc của các hệ điều hành, các thành phần cơ bản và các phục vụ của hệ điều hành.

Nội dung

Trình bày các khái niệm cơ bản về hệ điều hành, các chức năng cơ bản của hệ điều hành và các thành phần của hệ điều hành.

I. CÁC KHÁI NIỆM CƠ BẢN

1. Tài nguyên hệ thống

Một hệ thống máy tính bao gồm các thiết bị phần cứng và các chương trình phần mềm. Trong các tài liệu giới thiệu về một hệ thống máy tính bất kỳ, các số liệu thống kê về phần cứng luôn là những yếu tố được quan tâm đầu tiên và là thành tố để nhận biết sức mạnh của hệ thống máy tính đó.

Tài nguyên về phần mềm cũng được chú ý thông qua mối quan tâm về hệ điều hành, các chương trình ứng dụng, các cơ sở dữ liệu đã cài đặt trong hệ thống. Trong giai đoạn hiện nay, giá trị thực sự của tài nguyên phần mềm luôn cao hơn nhiều so với giá trị của tài nguyên phần cứng.

Đứng trên phương diện hệ điều hành, mọi công việc của hệ điều hành đều bắt đầu từ một hệ thống phần cứng sẵn có và cần phải hoạt động như thế nào để phát huy được hiệu quả cao nhất của hệ thống phần cứng đó. Vì vậy, trong giáo trình này, chúng ta sẽ tập chung chủ yếu đến tài nguyên phần cứng và định hướng tới vấn đề khai thác hiệu quả các tài nguyên đó.

Để phát huy hiệu quả của các tài nguyên phần cứng, cần xem xét một số đặc trưng cơ bản và đánh giá giá trị của mỗi thành phần trong hệ thống phần cứng nhằm mục đích đưa ra các chiến lược ưu tiên thích đáng đối với từng thành phần khi xây dựng hệ điều hành và các chương trình điều khiển hoạt động hệ thống.

Theo cách tiếp cận của hệ điều hành, các tài nguyên phần cứng điển hình bao gồm: bộ xử lý trung tâm (CPU), bộ nhớ (trong và ngoài), hệ thống vào/ra (các thiết bị vào/ra và các chương trình điều khiển thiết bị)...

Trong các thiết bị nói trên, CPU là tài nguyên quan trọng nhất vì nó liên quan trực tiếp đến khả năng xử lý, tính toán của hệ thống. CPU được đặc trưng bởi hai yếu tố là tốc độ xử lý và độ dài từ máy.

- Tốc độ xử lý của CPU thường được tính theo tần số xung nhịp đồng hồ hoặc số lượng phép tính cơ bản được thực hiện trong một giây. Đơn vị đo tốc độ được tính theo MHz (tần số xung nhịp của đồng hồ trong một giây) hoặc MIPS (Million Instruction Per Second - triệu phép tính cơ bản trong một giây).

- Độ dài từ máy: là số lượng bit nhị phân của toán hạng trong phép tính cơ bản của CPU (trên thực tế đã có các loại CPU 8, 16, 32 và 64 bits, đây chính là độ dài từ máy của các chủng loại CPU này). Độ dài từ máy có quan hệ khá mật thiết với khả năng xử lý thông tin của CPU, chẳng hạn một CPU có độ dài từ máy 4 bytes (32 bits) mặc dù có tốc độ xử lý thấp hơn một CPU có độ dài từ máy 2 bytes (16 bits) nhưng khả năng xử lý thông tin của nó sẽ cao hơn rất nhiều. Như vậy, để tăng khả năng xử lý của CPU, chúng ta cần có giải pháp tăng cả tốc độ xử lý và độ dài từ máy.

Bộ nhớ cũng là một tài nguyên quan trọng của hệ thống, các đặc trưng cơ bản của bộ nhớ là thời gian truy nhập dữ liệu và dung lượng bộ nhớ. Bộ nhớ thường được tổ chức phân cấp dựa vào tốc độ truy nhập. Bộ nhớ được gọi là thực hiện nếu CPU có thể thực hiện một câu lệnh bất kỳ ghi trong đó. Bộ nhớ trong là một loại bộ nhớ thực hiện và nó thuộc quyền quản lý của hệ thống. Ngoài bộ nhớ trong, các hệ thống máy tính còn tồn tại bộ nhớ ngoài (bộ nhớ phụ - bộ nhớ thứ hai), thời gian truy nhập bộ nhớ ngoài thường lâu hơn bộ nhớ trong. Loại bộ nhớ ngoài được sử dụng phổ biến hiện nay là đĩa từ, băng từ và CD - ROM. Đối với bộ nhớ ngoài, hệ điều hành không quan tâm tới việc nó là loại lắp cố định

hay có thể thay đổi mà chỉ quan tâm tới hai yếu tố: loại bộ nhớ và phương pháp truy nhập, loại bộ nhớ chỉ thiết bị vật lý lưu trữ dữ liệu, còn phương pháp truy nhập liên quan đến quá trình tìm kiếm dữ liệu.

Các thiết bị đảm nhận việc chuyển giao thông tin giữa hệ thống và người sử dụng được gọi là các thiết bị ngoại vi hoặc các thiết bị vào/ra. Các thiết bị này được gắn vào máy tính thông qua các thiết bị điều khiển để tạo thành một kenh vào/ra. Kênh có chức năng thay CPU điều khiển sự trao đổi thông tin giữa bộ nhớ trong và các thiết bị ngoài.

Tất cả các tài nguyên phần cứng nói trên có thể được khai thác theo nhiều phương pháp khác nhau như: khai thác đồng thời (bộ nhớ trong và ngoài), khai thác tuần tự (máy in) hoặc cung cấp cho người sử dụng dưới dạng đã được biến đổi (tài nguyên ảo - tài nguyên logic).

Tóm lại, tất cả các thiết bị và các chương trình phần mềm đều có thể coi là tài nguyên của máy tính. Các tài nguyên này được phân loại dựa trên mức độ quan trọng và cách thức khai thác. Ví dụ: loại tài nguyên có thể khai thác một cách đồng thời, loại tài nguyên chỉ có thể khai thác một cách tuần tự, loại tài nguyên được cung cấp dưới dạng đã được biến đổi đôi chút so với ban đầu (gọi là tài nguyên ảo).

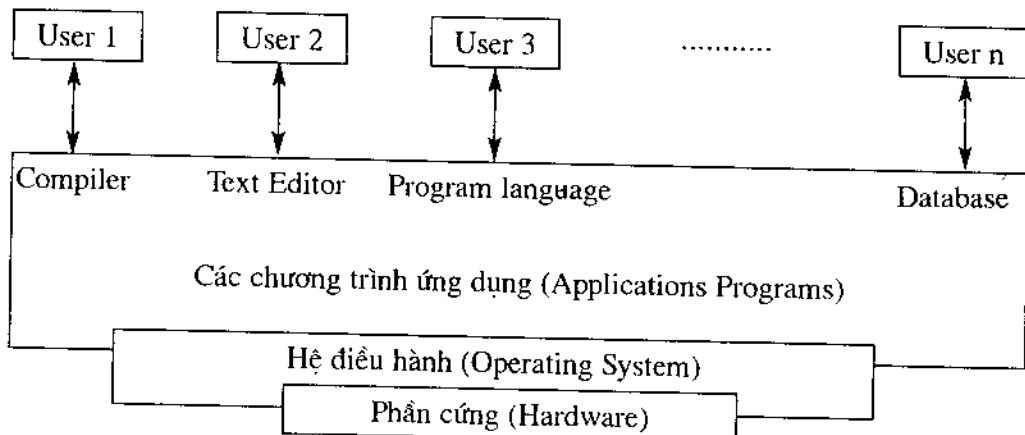
2. Khái niệm hệ điều hành

Hệ thống máy tính là một hệ thống kết hợp cả thiết bị phần cứng và vấn đề điều khiển, phân phối công việc trong toàn hệ thống. Để giải quyết bài toán này, không thể sử dụng một phương pháp thủ công mà cần có một cơ chế tự động hoá, tức là cần có một chương trình điều khiển hoạt động của hệ thống máy tính. Chương trình đó được gọi là **hệ điều hành** - Một thành phần quan trọng của hệ thống máy tính.

Để làm rõ khái niệm về hệ điều hành, chúng ta nhận thấy một hệ thống máy tính có thể phân chia thành 4 lớp:

- Phần cứng
- Hệ điều hành
- Các chương trình ứng dụng
- Người sử dụng (người sử dụng trực tiếp, các thiết bị chuyên dụng kết nối với máy tính, các máy tính khác).

Bốn lớp này có mối quan hệ mật thiết với nhau và được thể hiện qua sơ đồ sau:



Hình 1.1 - Mối quan hệ giữa hệ điều hành và các lớp trong hệ thống

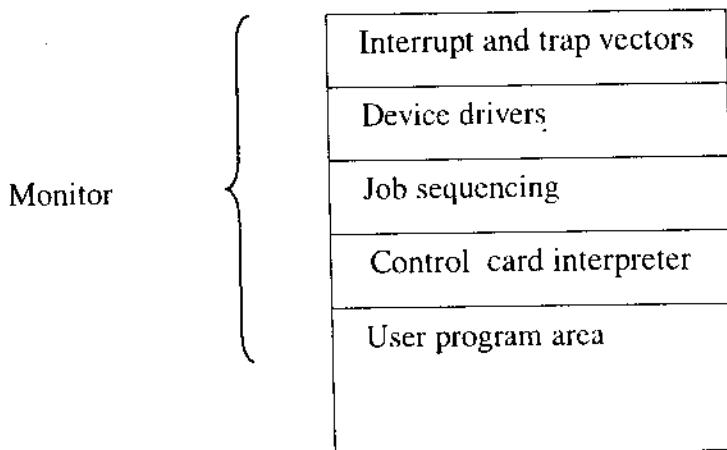
- Xét về phía người sử dụng thì hệ điều hành cần phải tạo được môi trường giao diện giữa người sử dụng và máy tính. Thông qua môi trường này, cho phép người sử dụng đưa ra các lệnh, chỉ thị điều khiển hoạt động của hệ thống.
- Xét về phía các chương trình ứng dụng thì hệ điều hành phải tạo môi trường để các chương trình ứng dụng hoạt động; cung cấp các cơ chế cho phép kích hoạt và loại bỏ các chương trình ứng dụng.
- Xét về phía phần cứng thì hệ điều hành phải quản lý các thiết bị một cách có hiệu quả, khai thác được hết khả năng của các thiết bị, cung cấp cho các chương trình và người sử dụng tài nguyên phần cứng khi có yêu cầu, thu hồi khi cần thiết.

Như vậy, có thể coi hệ điều hành là một tập hợp các chương trình hệ thống có chức năng tạo môi trường giao diện cho người sử dụng, tạo môi trường hoạt động cho các chương trình ứng dụng; quản lý và khai thác hiệu quả các thiết bị phần cứng.

3. Quá trình phát triển của hệ điều hành

Các hệ điều hành được phát triển song song với sự phát triển của máy tính điện tử. Ban đầu, các hệ điều hành làm việc theo phương pháp trọn gói, sau đó được bổ sung thêm các tính năng để có thể đáp ứng được nhu cầu công việc của

người sử dụng và sự phát triển của các hệ thống máy tính. Diễn hình là các giai đoạn sau:



Hình 1.2 - Cấu trúc monitor đơn giản

- Monitor đơn giản: Đây là hệ điều hành đầu tiên có thể tự động hoá, sắp xếp công việc cho máy tính thi hành. Monitor đơn giản là một chương trình nhỏ thường trú trong bộ nhớ. Các chương trình điều khiển thiết bị (Device Drivers) biết vùng đệm, các cờ nhớ, các thanh ghi và các bit kiểm tra của mình.

- Thao tác Off - Line: Mục đích của thao tác off - line là làm tăng hiệu quả của các thiết bị phân cứng. Thao tác off - line cho phép truy nhập các thiết bị một cách logic, không phụ thuộc vào tính chất vật lý của thiết bị dẫn đến loại trừ được hiện tượng các thiết bị vào/ra làm việc song hành với CPU.

- Thao tác Buffering: Buffering (thao tác tạo vùng đệm) nhằm làm tăng tốc các phép trao đổi ngoại vi, đảm bảo tốc độ chung của hệ thống. Thao tác buffering cho phép: giảm số lượng các thao tác vào/ra vật lý, thực hiện song song các thao tác vào/ra với các thao tác xử lý thông tin khác nhau, thực hiện trước các phép nhập dữ liệu...

- Thao tác SPOOL: SPOOL (Simultaneous Peripheral Operations On Line) là chế độ mà tất cả các trao đổi vào/ra, hệ điều hành chỉ làm việc với đĩa từ còn trao đổi giữa đĩa từ và các thiết bị được thực hiện theo các cơ chế riêng. Mục đích của SPOOL là cho phép hệ điều hành thao tác với các thiết bị một cách song song, làm tăng tốc độ của hệ thống một cách đáng kể.

- Đa chương trình và chia sẻ thời gian (Multi programming and Time sharing):

Trong giai đoạn này, các hệ điều hành cung cấp khả năng điều khiển hoạt động của nhiều chương trình tại cùng một thời điểm. Như vậy, các chương trình này đều có nhu cầu sử dụng tài nguyên trong cùng một thời điểm để thực hiện công việc, do đó tài nguyên hệ thống bị chia sẻ cho các chương trình. Trong khi đó, một số tài nguyên của hệ thống không thể cung cấp trong chế độ chia sẻ (ví dụ như CPU) dẫn đến hệ điều hành cần phải tổ chức phân bổ tài nguyên theo cơ chế hàng đợi. Khi một chương trình được thực hiện thì các chương trình còn lại phải ở trạng thái chờ được phân bổ tài nguyên nhưng vì thời gian tài nguyên phục vụ cho hoạt động của chương trình trong một chu kỳ là rất ngắn nên người sử dụng cảm nhận như chương trình của mình vẫn đang thực hiện và sở hữu toàn bộ tài nguyên hệ thống.

- Các chế độ bảo vệ: Để đảm bảo cho sự an toàn hệ thống và giúp cho hệ thống hoạt động ổn định, các hệ điều hành giai đoạn sau này còn bổ sung nhiều chế độ bảo vệ như: bảo vệ các thiết bị I/O, bảo vệ bộ nhớ, bảo vệ CPU... nhằm tránh các hiện tượng tranh chấp tài nguyên, sử dụng tài nguyên sai mục đích và khả năng gây lỗi tiềm ẩn của các thành phần hệ thống.

4. Phân loại hệ điều hành

Dựa trên phương thức hoạt động, điều khiển, quản lý tài nguyên... chúng ta có thể phân loại hệ điều hành như sau:

- **Hệ điều hành đơn nhiệm và hệ điều hành đa nhiệm:** hệ điều hành đơn nhiệm là hệ điều hành mà tại mỗi thời điểm chỉ có thể điều hành hoạt động của một chương trình. Khi một chương trình được nạp vào bộ nhớ thì nó sẽ chiếm dụng toàn bộ tài nguyên của hệ thống dẫn đến không thể thực thi một chương trình nào khác khi chương trình này chưa kết thúc. Hệ điều hành đa nhiệm là hệ điều hành cho phép thực hiện nhiều chương trình cùng một thời điểm. Tài nguyên trong chế độ hoạt động này được chia sẻ cho các chương trình dẫn đến cần phải đảm bảo tốt tính bình đẳng trong vấn đề phân phối tài nguyên.

- **Hệ điều hành đơn chương và hệ điều hành đa chương:** hệ điều hành đơn chương là tại mỗi thời điểm chỉ cho phép một người sử dụng làm việc. Hệ điều hành đa chương là tại mỗi thời điểm cho phép nhiều người sử dụng cùng làm việc.

- **Hệ điều hành chia sẻ thời gian và hệ điều hành thời gian thực:** Trong hệ điều hành chia sẻ thời gian, một CPU luôn luôn phục vụ các tiến trình và một tiến trình có thể rơi vào trạng thái chờ đợi khi chưa được phân bổ CPU, còn trong hệ điều hành thời gian thực, tiến trình được nạp vào hệ thống ở bất kỳ thời

điểm nào đều được phân bổ giờ CPU.

- Hệ điều hành tập trung và hệ điều hành phân tán: hệ điều hành tập trung được cài đặt trên hệ thống máy chủ của mạng, nó điều hành mọi thao tác, xử lý và tính toán tại các máy trạm. Hệ điều hành phân tán gồm hai thành phần được cài đặt trên máy chủ và máy trạm của mạng. Hệ điều hành tại máy chủ chịu trách nhiệm cung ứng dịch vụ, quản lý hệ thống và thực hiện các thao tác xử lý chung; hệ điều hành tại máy trạm có thể thực hiện các thao tác xử lý riêng.

5. Các tính chất cơ bản của hệ điều hành

- Tính tin cậy: Mọi hoạt động, mọi thông báo của hệ điều hành phải chuẩn xác tuyệt đối. Chỉ khi nào chắc chắn đúng thì hệ điều hành mới cung cấp thông tin cho người sử dụng. Ví dụ khi truy nhập đĩa, nếu gặp lỗi truy nhập hệ điều hành cố gắng lặp lại thao tác nhiều lần. Nếu vẫn không được, lúc đó mới đưa ra các thông báo lỗi.

- Tính an toàn: Hệ điều hành cần phải đảm bảo sao cho dữ liệu và các chương trình không bị thay đổi ngoài ý muốn trong mọi trường hợp và trong mọi chế độ hoạt động. Để đảm bảo được yếu tố an toàn, các hệ điều hành cần cung cấp các cơ chế bảo vệ dữ liệu và bảo vệ các tài nguyên sử dụng chung, tránh được sự vi phạm do vô tình hoặc cố ý của người sử dụng và các chương trình.

- Tính hiệu quả: Các tài nguyên của hệ thống phải được khai thác một cách triệt để sao cho ngay cả khi tài nguyên hạn chế vẫn có thể giải quyết được những yêu cầu phức tạp. Một khía cạnh khác của tính hiệu quả là phải duy trì hoạt động đồng bộ trong toàn hệ thống, không được để những thiết bị chậm trì hoãn hoạt động của hệ thống.

- Tính kế thừa: Hệ điều hành phải có tính kế thừa các ưu điểm, khắc phục các nhược điểm của phiên bản trước và khả năng thích nghi với những thay đổi trong tương lai. Tính kế thừa là rất quan trọng, ngay cả với các hệ điều hành thế hệ mới. Khi nâng cấp hệ điều hành thì tính kế thừa mang tính chất bắt buộc. Ví dụ như các thao tác, thông báo không được thay đổi hoặc nếu có thì cần hạn chế và phải được hướng dẫn cụ thể khi chuyển đổi từ phiên bản này sang phiên bản khác. Đảm bảo tính kế thừa sẽ duy trì và phát triển đội ngũ người sử dụng, giảm chi phí đào tạo khi tiếp cận tới các phiên bản hệ điều hành mới.

- Tính thuận lợi: Hệ điều hành phải sử dụng dễ dàng, có hiệu quả tuỳ theo kiến thức và kinh nghiệm của người dùng. Hệ điều hành phải có hệ thống trợ giúp, hướng dẫn phong phú, đầy đủ giúp người sử dụng có thể tự đào tạo mình

ngay trong quá trình khai thác.

Trong một khía cạnh nào đó, các tính chất trên có thể mâu thuẫn với nhau nhưng mỗi hệ điều hành cần có một giải pháp trung hoà, ưu tiên hợp lý ở tính chất này hoặc tính chất khác.

6. Các nguyên tắc thiết kế và xây dựng hệ điều hành

Khi xây dựng hệ điều hành, các modul chương trình phải được thiết kế dựa trên một số nguyên tắc sau để đảm bảo các tính chất của hệ điều hành.

- Nguyên tắc modul: Hệ điều hành được xây dựng từ những modul độc lập và giữa chúng có các quy tắc liên kết thành một hệ thống có tổ chức. Nguyên tắc modul thể hiện ở hai dạng: dạng chức năng và dạng chương trình. Các modul quan hệ với nhau thông qua dữ liệu vào và ra, quan hệ phân cấp giữa các modul được thiết lập khi liên kết chúng thành các modul lớn để giải quyết các vấn đề phức tạp. Nguyên tắc Modul cho phép tổ hợp những modul đã có theo nhiều cách khác nhau, đảm bảo tính đa dạng chức năng của hệ điều hành.

- Nguyên tắc tương đối trong định vị: Các modul chương trình được viết theo địa chỉ tương đối tính từ đầu bộ nhớ, khi thực hiện chúng mới được định vị vào một vùng nhớ cụ thể. Nguyên tắc này giúp cho hệ thống sử dụng bộ nhớ một cách linh hoạt và hệ điều hành không bị phụ thuộc vào cấu hình bộ nhớ cụ thể.

- Nguyên tắc Macro Processor: Theo nguyên tắc này, khi có nhiệm vụ cụ thể hệ thống sẽ xây dựng các thẻ yêu cầu, liệt kê các công việc phải thực hiện. Trên cơ sở đó thực hiện các chương trình tương ứng với công việc cần thực hiện. Mọi hệ điều hành đều sử dụng nguyên tắc này trong đối thoại giữa người và máy trên ngôn ngữ vận hành. Nguyên tắc này làm cho quá trình đối thoại linh hoạt hơn mà không cần tới một chương trình dịch phức tạp.

- Nguyên tắc lặp chức năng: Mỗi công việc phải có nhiều cách thực hiện khác nhau với những tổ hợp modul khác nhau. Điều này đảm bảo tính an toàn của hệ thống (vẫn có thể khai thác hệ thống khi thiếu hoặc hỏng một số thành phần nào đó) đồng thời người sử dụng sẽ thao tác dễ dàng hơn đối với hệ thống (cùng một công việc nhưng có thể thao tác theo nhiều cách khác nhau). Đôi khi trong hệ thống còn tồn tại nhiều modul khác nhau cùng giải quyết chung một vấn đề, sự đa dạng này cho phép người sử dụng lựa chọn được phương pháp thực hiện tối ưu cho công việc của mình.

- Nguyên tắc giá trị chuẩn: Mỗi modul có thể có rất nhiều tham số, việc nhớ hết các tham số và phạm vi sử dụng chúng là vấn đề rất phức tạp và modul sẽ

trở nên công kenne một cách không cần thiết. Để giải quyết vấn đề này, trong mỗi modul có một tập các tham số ứng với những trường hợp thường gặp nhất. Nếu trong câu lệnh gọi modul thiếu tham số nào thì hệ thống sẽ bổ sung từ tập tham số này. Nguyên tắc này thể hiện rất rõ trong các hệ thống cài đặt.

- Nguyên tắc khởi tạo khi cài đặt: Khi cài đặt hệ điều hành, chương trình cài đặt sẽ tạo những phiên bản làm việc thích hợp với những tham số kỹ thuật hiện có, loại bỏ những modul không cần thiết để có một phiên bản tối ưu cả về cấu trúc lẫn phương thức hoạt động.

- Nguyên tắc bảo vệ nhiều mức: Để đảm bảo an toàn hệ thống và dữ liệu, các chương trình và dữ liệu phải được bảo vệ ở nhiều mức khác nhau. Cơ chế bảo vệ nhiều mức đã làm giảm đáng kể các lỗi không cố ý của các tiến trình và người sử dụng.

II. CÁC CHỨC NĂNG CƠ BẢN CỦA HỆ ĐIỀU HÀNH

1. Quản lý tiến trình

Có thể coi tiến trình là một chương trình đang hoạt động, khi thực hiện, tiến trình đòi hỏi một số tài nguyên nhất định như: CPU, bộ nhớ, các thiết bị... Các tài nguyên này sẽ được cấp phát cho tiến trình vào những thời điểm cần thiết và được thu hồi khi tiến trình kết thúc. Ngoài ra, khi tiến trình hoạt động trong hệ thống có thể phát sinh các tiến trình con. Như vậy, nhiệm vụ của hệ điều hành trong quản lý tiến trình là:

- Đảm bảo những điều kiện tối thiểu để tiến trình có thể thực thi.
- Đảm bảo điều kiện cho sự hoạt động song song của nhiều tiến trình.
- Tạo và xoá các tiến trình của người sử dụng và hệ thống.
- Ngừng và bắt đầu lại các tiến trình.
- Tạo các cơ chế để đồng bộ hóa các tiến trình.
- Tạo các cơ chế để liên lạc giữa các tiến trình.
- Tạo các cơ chế để xử lý bối tắc.

2. Quản lý bộ nhớ

Bộ nhớ trong là thiết bị lưu trữ mà CPU có thể truy xuất một cách trực tiếp. Khi tổ chức một chương trình, sau biến dịch chương trình được chuyển sang ngôn ngữ máy tính, khi đó nó có các địa chỉ tương đối. Khi thực hiện, chương trình được nạp vào bộ nhớ, các địa chỉ tương đối sẽ được chuyển đổi thành các

địa chỉ vật lý xác định để CPU có thể truy xuất được trong quá trình xử lý, đó là quá trình sinh địa chỉ. Sau khi chương trình hoạt động xong, hệ thống cần phải giải phóng các địa chỉ vật lý đã cấp phát (giải phóng bộ nhớ). Để tăng hiệu xuất xử lý của hệ thống, tại cùng một thời điểm hệ thống có thể cho phép nhiều chương trình cùng tồn tại trong bộ nhớ. Do đó, hệ điều hành cần thực hiện các nhiệm vụ cơ bản trong quản lý bộ nhớ là:

- Cấp phát và thu hồi không gian nhớ cho các tiến trình.
- Lưu trữ dấu vết những thành phần của bộ nhớ hiện đang sử dụng và do tiến trình nào sử dụng.
- Quyết định tiến trình nào được nạp vào bộ nhớ khi có khả năng.
- Sắp xếp và giải phóng không gian nhớ khi cần thiết.

3. Quản lý bộ nhớ ngoài

Khi cần lưu trữ các chương trình hoặc dữ liệu, các hệ thống máy tính bắt buộc phải sử dụng bộ nhớ ngoài (đĩa từ, băng từ, compaq...). Nhiệm vụ chính của hệ điều hành phải đảm bảo được các chức năng sau:

- Quản lý không gian nhớ tự do trên bộ nhớ ngoài.
- Cấp phát không gian nhớ tự do.
- Cung cấp các khả năng định vị bộ nhớ ngoài.
- Lập lịch cho bộ nhớ ngoài.

4. Quản lý hệ thống vào/ra

Một trong những mục tiêu của hệ điều hành là che dấu các chi tiết phần cứng cụ thể đối với người sử dụng. Điều khiển hoạt động của các thiết bị bằng cách gửi các lệnh điều khiển tới thiết bị và tiếp nhận, xử lý các tín hiệu ngắn, xử lý lõi... Hơn nữa, hệ điều hành cần cung cấp một giao diện đơn giản, độc lập giữa các thiết bị và hệ thống. Do đó, chức năng của hệ điều hành trong quản lý hệ thống vào/ra là:

- Che dấu những đặc thù của các thiết bị vào/ra.
- Tạo lập những chương trình để quản lý, điều khiển thiết bị chung và các thiết bị đặc biệt.

5. Quản lý file

Khi lưu trữ thông tin trên bộ nhớ ngoài, mỗi thiết bị lưu trữ sẽ có những đặc tính vật lý khác nhau. Để tạo điều kiện thuận lợi trong công tác lưu trữ và quản lý thông tin, hệ điều hành sử dụng một đơn vị lưu trữ đồng nhất trên tất cả các

thiết bị lưu trữ gọi là tệp tin (file). Để có thể dễ dàng truy xuất, hệ điều hành còn tổ chức các file thành các thư mục và kiểm soát việc truy nhập đồng thời đến cùng một file. Như vậy, trong lĩnh vực quản lý file, hệ điều hành chịu trách nhiệm về các thao tác sau đây:

- Tạo và xoá file.
- Tạo và xoá thư mục.
- Hỗ trợ các nguyên lý thao tác file và thư mục.
- Ánh xạ các file lên bộ nhớ phụ.
- Ghi dự phòng các file lên bộ nhớ ổn định.

6. Hệ thống bảo vệ

Khi hệ thống cho phép nhiều người sử dụng đồng thời, các tiến trình song hành cần phải được bảo vệ để tránh sự xâm phạm vô tình hoặc cố ý có thể gây sai lệch toàn hệ thống. Hệ điều hành cần xây dựng các cơ chế bảo vệ, cho phép đặc tả sự kiểm soát và một phương thức để áp dụng các chiến lược bảo vệ thích hợp. Như vậy, mục đích của hệ thống bảo vệ là:

- Giúp cho hệ thống hoạt động bình thường.
- Bảo vệ các tài nguyên sử dụng chung.
- Phát triển và ngăn chặn các khả năng sai sót của các tiến trình.

7. Lập mạng

Mạng máy tính là một tập hợp các máy tính được kết nối với nhau bằng môi trường truyền tin nhằm mục đích cho phép người sử dụng dùng chung tài nguyên và phục vụ công tác truyền thông. Mỗi máy tính trong mạng có một bộ nhớ độc lập và các tiến trình có thể được kết nối, xử lý thông qua hệ thống mạng. Khi đó hệ điều hành phải hỗ trợ khả năng quản lý, chia sẻ tài nguyên, truyền thông trên mạng thông qua các thành phần điều khiển giao tiếp mạng.

8. Hệ thống giải thích lệnh (thông dịch lệnh)

Hệ thống giải thích lệnh là thành phần quan trọng nhất của hệ điều hành, đóng vai trò tạo giao diện giữa máy tính và người sử dụng. Nó giúp máy tính hiểu và xử lý được các chỉ thị, các lệnh của người sử dụng.

III. CÁC THÀNH PHẦN CỦA HỆ ĐIỀU HÀNH

1. Các phục vụ của hệ điều hành

Hệ điều hành tạo ra môi trường cho các chương trình hoạt động, do đó hệ

điều hành phải phục vụ chương trình và những người sử dụng chương trình đó. Với những hệ điều hành khác nhau thì sẽ có một số các phục vụ đặc biệt khác nhau nhưng về nguyên tắc chung, các hệ điều hành phải có một số kiểu phục vụ sau:

- Phục vụ thực hiện chương trình.
- Điều khiển thao tác vào/ra.
- Các thao tác file.
- Phát hiện lỗi sai sót.
- Phân phối tài nguyên.
- Thống kê, kế toán.
- Tổ chức các phục vụ.

2. Các gọi hệ thống

Các gọi hệ thống (system call) cung cấp một giao diện giữa chương trình đang hoạt động và hệ điều hành. Hệ điều hành cung cấp hai phương pháp để tổ chức thực hiện các gọi hệ thống:

- Tổ chức bằng những lệnh hợp ngữ.
- Tổ chức trực tiếp từ chương trình ngôn ngữ bậc cao bằng cách sử dụng chương trình con.
 - Các gọi hệ thống được chia thành ba loại chính:
 - + Các chương trình điều khiển tiến trình thực thi.
 - + Các chương trình thao tác với file và thiết bị.
 - + Các chương trình bảo trì thông tin hệ thống.

3. Các chương trình hệ thống

Các chương trình hệ thống cung cấp công cụ cho người sử dụng thực hiện các thao tác quản lý và điều khiển hệ thống. Điển hình là:

- Các chương trình thao tác với file và thư mục.
- Các chương trình thông tin trạng thái.
- Các chương trình hỗ trợ ngôn ngữ lập trình.
- Các chương trình điều khiển nạp và thực hiện chương trình.
- Chương trình giải thích lệnh (Command Interpreter).

4. Các chương trình ứng dụng

Các chương trình ứng dụng đi kèm hệ điều hành nhằm mục đích hỗ trợ cho người sử dụng thực hiện các thao tác ứng dụng cơ bản như: các chương trình soạn thảo văn bản đơn giản, các trình duyệt Web, các chương trình trò chơi giải trí...

Câu hỏi ôn tập

1. Trình bày khái niệm về tài nguyên hệ thống. Cho ví dụ minh họa.
2. Nêu các chức năng cơ bản của hệ điều hành. Cho ví dụ minh họa.
3. Trình bày mối quan hệ giữa hệ điều hành và các thành phần trong hệ thống, từ đó nêu khái niệm về hệ điều hành.
4. Phân biệt hệ điều hành đơn nhiệm, đa nhiệm, đơn sắc và đồ họa. Cho ví dụ thông qua hệ điều hành DOS và Windows.
5. Thông qua các hệ điều hành đã học, cho các ví dụ minh họa để làm rõ các tính chất cơ bản của hệ điều hành.
6. Trình bày các nguyên tắc thiết kế và xây dựng hệ điều hành. Cho ví dụ qua các hệ điều hành đã học.
7. So sánh cơ chế bảo vệ của hệ điều hành Windows 9x và Windows 2000 hoặc XP.
8. Nêu các chương trình hệ thống, chương trình ứng dụng trong hệ điều hành DOS và hệ điều hành Windows.

Chương 2

QUẢN LÝ TIẾN TRÌNH

Mục tiêu

Sau chương này, người học có thể hiểu được khái niệm tiến trình và các mối quan hệ của chúng trong hệ thống. Biết được các biện pháp quản lý tiến trình song hành với nội dung chính là giải quyết bài toán tranh chấp tài nguyên giữa các tiến trình. Đồng thời, người học có thể hình dung được mối nguy hại của bế tắc, biết được các mức phòng tránh bế tắc của hệ điều hành và có thể giải quyết được một số tình trạng bế tắc.

Nội dung

Trình bày các khái niệm cơ bản về tiến trình, bài toán đoạn tới hạn và các phương pháp giải quyết. Khái niệm về hiện tượng bế tắc và các biện pháp phòng tránh bế tắc.

I. CÁC KHÁI NIỆM CƠ BẢN

1. Khái niệm tiến trình

Để hỗ trợ hoạt động đa nhiệm, hệ thống máy tính cần phải có khả năng thực hiện nhiều tác vụ xử lý đồng thời nhưng việc điều khiển hoạt động song hành ở cấp độ phần cứng là rất khó khăn. Vì vậy, các nhà thiết kế hệ điều hành đã xuất một mô hình song hành giả lập bằng cách chuyển đổi bộ xử lý qua lại giữa các chương trình để duy trì hoạt động của nhiều chương trình tại cùng một thời điểm. Trong mô hình này, các chương trình trong hệ thống được tổ chức thành các tiến trình (process).

Như vậy, có thể coi tiến trình là một chương trình đang xử lý, nó sở hữu một con trỏ lệnh, tập các thanh ghi và các biến. Để hoàn thành nhiệm vụ của mình, các tiến trình có thể còn yêu cầu một số tài nguyên hệ thống như: CPU, bộ nhớ và các thiết bị.

Chúng ta cần phân biệt rõ hai khái niệm chương trình và tiến trình. Chương trình là một thực thể thụ động chưa đựng các chỉ thị điều khiển máy tính thi

hành một tác vụ cụ thể nào đó. Khi cho thực hiện các chỉ thị này, chương trình được chuyển thành tiến trình là một thực thể hoạt động, với con trỏ lệnh xác định chỉ thị kế tiếp sẽ thi hành kèm theo các tập tài nguyên phục vụ cho hoạt động của tiến trình.

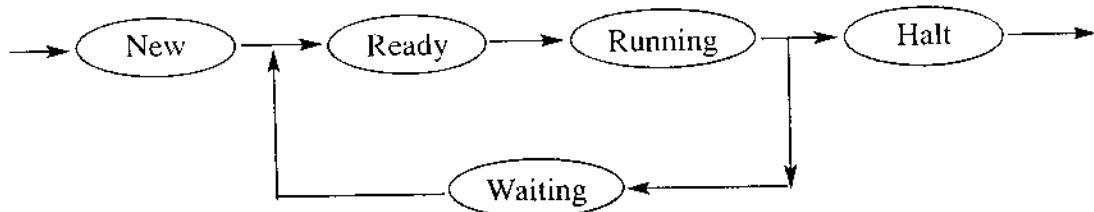
2. Các trạng thái của một tiến trình

Trạng thái của tiến trình tại mỗi thời điểm được xác định bởi hoạt động hiện thời của tiến trình tại thời điểm đó. Trong suốt khoảng thời gian tồn tại trong hệ thống, một tiến trình có thể thay đổi trạng thái do rất nhiều nguyên nhân như: chờ đợi sự kiện nào đó xảy ra, đợi một thao tác vào/ra hoàn tất, hết thời gian xử lý...

Tại mỗi thời điểm, tiến trình có thể nhận một trong các trạng thái sau:

- Khởi tạo (new): tiến trình đang được tạo lập.
- Sẵn sàng (ready): tiến trình chờ được cấp phát CPU để xử lý.
- Thực hiện (running): tiến trình được xử lý.
- Đợi (waiting): tiến trình phải dừng vì thiếu tài nguyên hoặc chờ một sự kiện nào đó.
- Kết thúc (halt): tiến trình đã hoàn tất công việc xử lý.

Các trạng thái của tiến trình có thể được biểu diễn qua sơ đồ sau:



Hình 2.1. Các trạng thái của một tiến trình

Hệ điều hành quản lý hoạt động của các tiến trình trong hệ thống thông qua khối mô tả tiến trình (Process Control Block – PCB). Khối mô tả tiến trình bao gồm các thành phần:

- Số thứ tự của tiến trình.
- Con trỏ trạng thái của tiến trình (cho biết trạng thái hiện tại của tiến trình).
- Vùng nhớ lưu trữ giá trị các thanh ghi mà tiến trình đang sử dụng.
- Thông tin về tài nguyên tiến trình đang sử dụng hoặc được phép sử dụng.

3. Quan hệ giữa các tiến trình

Các tiến trình hoạt động trong hệ thống tồn tại hai mối quan hệ: độc lập và hợp tác (song hành).

* *Quan hệ độc lập*: tiến trình được gọi là độc lập nếu hoạt động của nó không gây ảnh hưởng hoặc không bị ảnh hưởng bởi các tiến trình khác cũng đang hoạt động của hệ thống. Tiến trình độc lập có những đặc trưng sau:

- Trạng thái của nó không bị chia sẻ với bất kỳ tiến trình nào khác.
- Việc thực hiện tiến trình là đơn định (kết quả chỉ phụ thuộc vào đầu vào).
- Tiến trình có thể tái hiện (lặp lại).
- Tiến trình có thể dừng hoặc bắt đầu lại mà không gây ảnh hưởng tới các tiến trình khác trong hệ thống.

* *Quan hệ hợp tác*: tiến trình được gọi là hợp tác (song hành) nếu hoạt động của nó gây ảnh hưởng hoặc bị ảnh hưởng bởi các tiến trình khác cũng đang hoạt động trong hệ thống. Tiến trình hợp tác có những đặc trưng sau:

- Trạng thái của nó bị chia sẻ cho các tiến trình khác.
- Việc thực hiện tiến trình không đơn định (kết quả phụ thuộc dây thực hiện tương ứng và không dự báo trước).
- Tiến trình không thể tái hiện.

4. Ví dụ về tiến trình song hành - Bài toán nhà sản xuất và người tiêu dùng (Producer/ Consumer Problem)

* *Bài toán*: Giả sử có hai tiến trình P và C song hành. Tiến trình P cung cấp thông tin cho hoạt động của tiến trình C. Thông tin do P sản xuất được đặt trong vùng đệm và C lấy thông tin từ vùng đệm để sử dụng. Trong trường hợp vùng đệm có kích thước hạn chế, hãy xây dựng thuật toán điều khiển hoạt động của hai tiến trình trên.

* *Thuật toán*: Khi kích thước vùng đệm hạn chế sẽ xảy ra hai trường hợp:

- Vùng đệm đầy, khi đó tiến trình P phải ở trạng thái phải chờ cho tới khi có vùng đệm rỗng.

- Vùng đệm rỗng, khi đó tiến trình C phải ở trạng thái phải chờ thông tin.

Giả sử:

- Vùng đệm chứa được n phần tử
- Thông tin có kiểu *Item* nào đó
- Biến *In* chỉ số phân tử được sản xuất

- Biến *Out* chỉ số phân tử được tiêu thụ
 - Tiến trình P sản xuất thông tin chứa trong biến trung gian *NextP*
 - Tiến trình C tiêu thụ thông tin chứa trong biến trung gian *NextC*
- Khi đó:
- Vùng đệm rỗng khi *In* = *Out*
 - Vùng đệm đầy khi $(In + 1) \bmod n = Out$

* *Thuật toán:*

Type *Item* = ...;

Var *Buffer*: array [0..n - 1] of *Item*;

In, *Out* : 0..n - 1

Begin

{Tiến trình P}

Repeat

...

Sản xuất thông tin và chứa trong *NextP*;

...

While $(In + 1) \bmod n = Out$ **do Skip;**

Buffer[*In*] := *NextP*

In := *In* + 1 **mod** *n*;

Until *false*;

{Tiến trình C}

Repeat

While *In* = *Out* **do Skip;**

NextC := *Buffer*[*Out*];

Out := *Out* + 1 **mod** *n*

...

Lấy thông tin trong chứa trong *NextC*

...

Until *false*;

End;

5. Khái niệm về tài nguyên “gắn” và đoạn tối hạn

Trong thuật toán trên nếu ta dùng một biến đếm Counter (được khởi đầu = 0) dùng để đếm số lượng các phân tử trong vùng đệm. Khi đó giá trị của biến Counter sẽ tăng 1 khi P sản xuất thêm 1 phân tử và giảm đi 1 khi C lấy đi 1 phân

tử. Đồng thời vùng đệm đầy khi Counter = n và vùng đệm rỗng khi Counter = 0.

Thuật toán có thể viết lại như sau:

Type *Item* = ...;
Var *Buffer*: array [0..n - 1] of *Item*;
 In, *Out* : 0..n - 1
 Counter : 0..n

Begin

{Tiến trình P}

Repeat

...

Sản xuất thông tin và chứa trong *NextP*;

...

While *Counter* = n **do** *Skip*;

Buffer[*In*] := *NextP*;

In := *In* + 1 mod n;

Counter := *Counter* + 1;

Until false;

{Tiến trình C}

Repeat

While *Counter* = 0 **do** *Skip*;

NextC := *Buffer*[*Out*];

Out := *Out* + 1 mod n;

Counter := *Counter* - 1;

...

Lấy thông tin trong chứa trong *NextC*

...

Until false;

End;

Nhận xét: Vì hai tiến trình P và C song hành nên có thể xảy ra trường hợp trong cùng một thời điểm, tiến trình P tăng biến Counter lên 1 (*Counter* := *Counter* + 1) còn tiến trình C lại giảm biến Counter đi 1 (*Counter* := *Counter* - 1) dẫn tới kết quả sai. Như vậy, biến Counter trong trường hợp này được gọi là tài nguyên “găng”. Đoạn trình sử dụng biến Counter gọi là đoạn tối hạn.

Định nghĩa:

Các tài nguyên logic và vật lý phân bổ cho các tiến trình song hành là tài nguyên “găng”.

Các đoạn trình sử dụng tài nguyên găng gọi là đoạn tối hạn (*Critical Section*).

6. Bài toán đoạn tối hạn

* *Bài toán:* Giả sử có n tiến trình P_0, P_1, \dots, P_{n-1} song hành, mỗi tiến trình có một đoạn tối hạn. Tìm một phương thức sao cho các tiến trình vượt qua đoạn tối hạn của mình mà không ảnh hưởng tới hoạt động của hệ thống.

* *Nhận xét:* Việc giải quyết bài toán đoạn tối hạn là phải thiết kế một nghị thức sao cho các tiến trình có thể sử dụng để hợp tác với nhau và thỏa mãn ba điều kiện:

- Điều kiện loại trừ lẫn nhau: tại mỗi thời điểm, chỉ có một tiến trình được phép thực hiện trong đoạn tối hạn của mình.

- Điều kiện tiến triển: không tiến trình nào được phép ở lâu vô hạn trong đoạn tối hạn của mình.

- Điều kiện chờ đợi có giới hạn: các tiến trình không phải chờ đợi vô hạn trước khi đi vào đoạn tối hạn của mình.

Trên thực tế, hai xu hướng mà các hệ điều hành thường áp dụng để giải quyết bài toán đoạn tối hạn là:

- + Sử dụng các thuật toán cấp thấp: là các thuật toán nằm ngay trong tiến trình.

- + Sử dụng các thuật toán cấp cao: là các thuật toán nằm ngoài tiến trình.

II. CÁC PHƯƠNG PHÁP GIẢI QUYẾT BÀI TOÁN ĐOẠN TỐI HẠN

1. Phương pháp khóa trong

1.1. Nguyên tắc chung

Phương pháp này dựa trên cơ sở nếu hai hay nhiều tiến trình cùng định ghi vào một địa chỉ nào đó của bộ nhớ trong, thì giải thuật chỉ cho phép một tiến trình được thực hiện còn các tiến trình khác phải chờ.

Mỗi tiến trình sử dụng một byte trong bộ nhớ để làm khoá. Khi tiến trình vào đoạn tối hạn, byte khoá của nó được gán giá trị = 1 để thông báo cho các tiến trình còn lại biết tài nguyên găng đã được sử dụng. Khi ra khỏi đoạn tối

hạn, byte khoá được gán giá trị = 0 để thông báo tài nguyên găng đã được giải phóng.

Trước khi vào đoạn tối hạn, các tiến trình phải kiểm tra byte khoá của các tiến trình khác. Nếu có byte nào đó chứa giá trị = 1 thì tiến trình phải chờ cho tới khi byte đó nhận giá trị = 0.

1.2. Thuật toán (bài toán 2 tiến trình)

Var $K1, K2$: byte;

Begin

$K1 := 0; K2 := 0;$

{Tiến trình 1}

Repeat

While $K2 = 1$ **do Skip;**

$K1 := 1;$

Tiến trình 1 vào đoạn tối hạn;

$K1 := 0;$

Phần còn lại của tiến trình 1;

Until *false*;

{Tiến trình 2}

Repeat

While $K1 = 1$ **do Skip;**

$K2 := 1;$

Tiến trình 2 vào đoạn tối hạn;

$K2 := 0;$

Phần còn lại của tiến trình 2;

Until *false*;

End;

Nhận xét: Nhược điểm của giải thuật trên là không đảm bảo tính loại trừ lẫn nhau. Giả sử tiến trình 1 thực hiện rất nhanh so với tiến trình 2 và tiến trình 1 đang trong đoạn tối hạn còn tiến trình 2 đang chờ vào đoạn tối hạn. Sau khi tiến trình 1 ra khỏi đoạn tối hạn $\rightarrow K1 = 0$ và tiến trình 2 thoát khỏi trạng thái chờ nhưng chưa vào đoạn tối hạn $\rightarrow K2 \neq 1$. Trong khi đó tiến trình 1 đã thực hiện xong phần còn lại của mình và quay trở lại đầu tiến trình. Vì $K2 \neq 1$ nên tiến trình 1 có thể thực hiện phép gán $K1 = 1$ dẫn đến cả hai tiến trình cùng vào đoạn tối hạn.

1.3. Thuật toán Dekker

Để giải quyết nhược điểm trên, thuật toán của Dekker dùng thêm một biến khoá TT để xác định độ ưu tiên của các tiến trình khi cả hai tiến trình cùng muốn vào đoạn tới hạn (TT = 1 hoặc TT = 2).

Var $K1, K2, TT: byte;$

Begin

$K1 := 0; K2 := 0;$

$TT := 1;$

{Tiến trình 1}

Repeat

$K1 := 1;$

While $K2 = 1$ **do**

If $TT = 2$ **then**

begin

$K1 := 0;$

While $TT = 2$ **do Skip;**

$K1 = 1;$

end;

Tiến trình 1 vào đoạn tới hạn;

$K1 := 0; TT := 2;$

Phản còn lại của tiến trình 1;

Until *false*;

{Tiến trình 2}

Repeat

$K2 := 1;$

While $K1 = 1$ **do**

If $TT = 1$ **then**

begin

$K2 := 0;$

While $TT = 1$ **do Skip;**

$K2 = 1;$

End;

Tiến trình 2 vào đoạn tới hạn;

$K2 := 0; TT := 1;$

Phần còn lại của tiến trình 1;

Until false;

End;

Thuật toán Dekker giải quyết bài toán đoạn tối hạn hợp lý trong mọi trường hợp cho dù tốc độ thực hiện của các tiến trình khác nhau.

* *Ưu điểm và nhược điểm của phương pháp khoá trong*

- **Ưu điểm:** phương pháp khoá trong không đòi hỏi công cụ đặc biệt, do đó có thể tổ chức bằng một ngôn ngữ bất kỳ và thực hiện trên mọi hệ thống.

- **Nhược điểm:** độ phức tạp của thuật toán sẽ tăng khi số tiến trình nhiều hoặc số lượng đoạn tối hạn trong các tiến trình lớn; các tiến trình phải chờ đợi ở trạng thái tích cực.

2. Phương pháp kiểm tra và xác lập

2.1. Nguyên tắc chung

Phương pháp này dựa vào sự hỗ trợ của phần cứng, có một lệnh thực hiện hai phép xử lý liên tục không bị tách rời.

Giả sử ta gọi lệnh đó là TS (Test and Set) lệnh này có 2 tham số: biến chung G và biến riêng L (biến chung G thông thường là một bit trong từ trạng thái hoặc trong thanh ghi cờ). Dạng thức thực hiện của lệnh TS (L) như sau:

L := G (gán giá trị biến chung cho biến riêng)

G := 1 (gán giá trị 1 cho biến chung)

2.2. Thuật toán (bài toán 2 tiến trình)

Var L1, L2, G, TT : byte;

Begin

 G := 0; TT = 1;

{Tiến trình 1}

 Repeat

 L1 := 1;

 While L1 = 1 do TS (L1);

 Tiến trình 1 vào đoạn tối hạn;

 G := 0; TT = 2;

 Phần còn lại của tiến trình 1;

 Until false;

{Tiến trình 2}

 Repeat

```

L2 := 1;
While L2 = 1 do TS (L2);
    Tiến trình 2 vào đoạn tối hạn;
    G := 0; TT = 1;
    Phần còn lại của tiến trình 2;
    Until false;
End;

```

* *Ưu điểm và nhược điểm của phương pháp kiểm tra và xác lập*

- **Ưu điểm:** phương pháp này đơn giản, độ phức tạp không tăng khi số tiến trình và số đoạn tối hạn tăng. Nhiều máy tính được trang bị tới vài lệnh kiểu này để điều khiển tiến trình như IBM PC có tới 4 lệnh.

- **Nhược điểm:** tiến trình vẫn phải chờ đợi tích cực; khó xác định được tiến trình nào sẽ vào đoạn tối hạn khi có quá nhiều tiến trình cùng chờ.

3. Phương pháp đèn hiệu

3.1. Nguyên tắc chung

Đèn hiệu S là một biến nguyên mà chỉ có hai phép xử lý WAIT và SIGNAL mới thay đổi được giá trị của nó.

Định nghĩa phép WAIT(S):

S := S - 1

Nếu S >= 0 tiếp tục thực hiện tiến trình

Nếu S < 0 đưa tiến trình vào hàng đợi

Định nghĩa phép SIGNAL(S)

S := S + 1

Nếu S <= 0 đưa tiến trình trong hàng đợi vào đoạn tối hạn

Chú ý:

- Phép WAIT và SIGNAL không bị phân chia trong tiến trình thực hiện.

- Nếu ban đầu S=1 và cả hai tiến trình đều đưa ra phép WAIT(S) thì chỉ có một tiến trình được phép vào đoạn tối hạn, tiến trình còn lại sẽ được đưa vào hàng đợi.

3.2. Thuật toán (cho bài toán 2 tiến trình)

Var S : byte;

Begin

 S := 1;

{Tiến trình 1}

```

Repeat
    WAIT(S);
    Tiến trình 1 vào đoạn tối hạn;
    SIGNAL(S)
    Phân còn lại của tiến trình 1;
    Until false;
{Tiến trình 2}
Repeat
    WAIT(S);
    Tiến trình 2 vào đoạn tối hạn;
    SIGNAL(S)
    Phân còn lại của tiến trình 2;
    Until false;

```

End;

* *Ưu điểm và nhược điểm của phương pháp đèn hiệu*

- *Ưu điểm:* trong phương pháp này, mỗi tiến trình chỉ cần kiểm tra quyền vào đoạn tối hạn một lần, sau đó nó được vào đoạn tối hạn hoặc phải xếp hàng đợi; trong khi đợi, tiến trình không ở trạng thái tích cực vì khi một tiến trình nào đó ra khỏi hàng đợi, nó sẽ bật tín hiệu đưa tiến trình thoát khỏi trạng thái chờ để vào đoạn tối hạn.

- *Nhược điểm:* phép WAIT và SIGNAL phải tổ chức và xử lý hàng đợi, do đó phụ thuộc vào từng hệ điều hành cụ thể và thường được thể hiện dưới dạng các thủ tục. Khi các tiến trình nằm trong hàng đợi, cần phải áp dụng các thuật toán xử lý hàng đợi để đạt được kết quả tối ưu.

4. Phương pháp dùng trình thư ký

4.1. Nguyên tắc chung

Trình thư ký (monitor) là bộ các thủ tục và cấu trúc thông tin động, hoạt động trong chế độ phân chia thời gian, hỗ trợ việc thực hiện tiến trình. Mỗi thời điểm chỉ có một tiến trình làm việc được với monitor.

4.2. Cơ chế hoạt động của monitor

Khi tiến trình muốn sử dụng tài nguyên, hệ thống gắn monitor vào tiến trình. Nếu được phép sử dụng tài nguyên thì tiến trình sẽ được tiếp tục bình thường nếu không thì tiến trình được xếp vào hàng đợi.

Khi tài nguyên đang được giải phóng thì monitor nhận điều khiển và bật tín

hiệu giải phóng tiến trình khỏi trạng thái chờ.

Chú ý:

- Monitor không phải là một tiến trình mà chỉ là một đối tượng thụ động.
- Monitor nằm ngoài tiến trình.
- Monitor được kích hoạt khi tiến trình cần sử dụng (thông thường monitor được hệ thống kích hoạt khi phân phối lại tài nguyên).

* *Ưu điểm và nhược điểm của phương pháp dùng trình thư ký*

- **Ưu điểm:** monitor chứa nhiều cấu trúc thông tin động nên nó có thể giải quyết tốt vấn đề đoạn tới hạn; monitor có thể ghép vào lớp công cụ hệ thống nên có tính phổ dụng cao (nhiều tiến trình có thể sử dụng được).
- **Nhược điểm:** monitor cũng có thể trở thành tài nguyên găng.

5. Phương pháp tổ chức liên lạc giữa các tiến trình

Hệ điều hành xây dựng một hệ thống báo giữa các tiến trình dựa trên 3 thao tác cơ bản:

- Send message: Gửi thông báo.
- Receive message: Nhận thông báo.
- Communication link: Tạo móc nối liên lạc giữa các tiến trình.

Dựa vào hệ thống báo này, các tiến trình có thể phối hợp để vào đoạn tới hạn bằng cách trao đổi thông báo cho nhau. Hệ thống báo sử dụng hai cách thức liên lạc:

- Trực tiếp: Bằng cách sử dụng hai thao tác gửi và nhận.
- Gián tiếp: Dùng hộp thư trung gian.

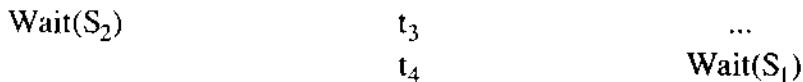
III. HIỆN TƯỢNG BẾ TẮC

1. Các khái niệm cơ bản

1.1. Khái niệm về bế tắc

Giả sử có hai tiến trình P_1 và P_2 song hành sử dụng các tài nguyên r_1 và r_2 được điều khiển bởi hai đèn hiệu S_1 và S_2 . Tại mỗi thời điểm, mỗi tài nguyên chỉ phục vụ cho sự hoạt động của một tiến trình. Xét trạng thái:

P_1	Thời điểm	P_2
Wait(S_1)	t_1	Wait(S_2)
...	t_2	...



Nhận xét: tại ví dụ trên, sau thời điểm t_3 , tiến trình P_1 rời vào trạng thái chờ tài nguyên r_2 đang được P_2 sử dụng; sau thời điểm t_4 , tiến trình P_2 rời vào trạng thái chờ tài nguyên r_1 đang được tiến trình P_1 sử dụng và bắt đầu từ đây, cả hai tiến trình rơi vào trạng thái chờ đợi vô hạn và hệ thống gặp bế tắc.

Như vậy, bế tắc là trạng thái khi hai hoặc nhiều tiến trình cùng chờ đợi một số sự kiện nào đó và nếu không có tác động đặc biệt từ bên ngoài thì sự chờ đợi đó là vô hạn.

1.2. Điều kiện xảy ra bế tắc trong hệ thống

Hiện tượng bế tắc xảy ra khi và chỉ khi trong hệ thống tồn tại bốn điều kiện:

- Có tài nguyên găng.
- Có hiện tượng giữ và đợi: có một tiến trình đang giữ một số tài nguyên và đợi tài nguyên bổ sung đang được giữ bởi các tiến trình khác.
- Không có hệ thống phân phối lại tài nguyên: việc sử dụng tài nguyên không bị ngắt.
- Có hiện tượng chờ đợi vòng tròn.

1.3. Các mức phòng tránh bế tắc

Để tránh hiện tượng bế tắc, thông thường hệ thống áp dụng ba mức:

- Ngăn ngừa: áp dụng các biện pháp để hệ thống không rơi vào trạng thái bế tắc.
- Dự báo và tránh bế tắc: áp dụng các biện pháp để kiểm tra các tiến trình xem có bị rơi vào trạng thái bế tắc hay không. Nếu có thì thông báo trước khi bế tắc xảy ra.
- Nhận biết và khắc phục: tìm cách phát hiện và giải quyết.

2. Các biện pháp phòng tránh bế tắc

2.1. Ngăn ngừa bế tắc

Để phòng ngừa bế tắc, cần phải đảm bảo sao cho 4 điều kiện gây bế tắc không xảy ra đồng thời.

2.1.1. Loại bỏ tài nguyên găng

Mô phỏng tài nguyên găng bằng các tài nguyên có thể dùng chung được (áp dụng kỹ thuật SPOOL).

2.1.2. Loại bỏ yếu tố giữ và đợi

Thực hiện phân bổ trước tài nguyên; tiến trình chỉ có thể thực hiện khi mọi tài nguyên mà nó yêu cầu đã được phân bổ đủ. Tiến trình chỉ được phép đòi tài

nguyên khi nó không giữ tài nguyên nào cả. Nếu tiến trình phải đợi thì mọi tài nguyên nó đang giữ phải tạm thời giải phóng.

2.1.3. Xây dựng hệ thống ngắt tài nguyên

Hệ thống ngắt tài nguyên có thể được xây dựng theo hai phương pháp:

- Phương pháp 1: Nếu tiến trình đang giữ một số tài nguyên và yêu cầu tài nguyên bổ sung nhưng hệ thống không thể phân bổ ngay thì mọi tài nguyên mà tiến trình đang giữ sẽ bị ngắt và được bổ sung vào danh sách các tài nguyên tự do. Tiến trình sẽ được bắt đầu lại khi nó được phân bổ đủ các tài nguyên cần thiết.

- Phương pháp 2: Nếu tiến trình đang giữ một số tài nguyên và yêu cầu tài nguyên bổ sung nhưng hệ thống không thể phân bổ ngay, khi đó hệ thống sẽ kiểm tra tài nguyên mà tiến trình yêu cầu có bị giữ bởi các tiến trình khác cũng đang đợi hay không. Nếu có thì ngắt các tiến trình này, thu hồi lại tài nguyên để phân bổ cho tiến trình yêu cầu; ngược lại tiến trình yêu cầu phải đợi và trong khi đợi, tài nguyên hiện có của nó cũng có thể bị ngắt khi có tiến trình khác yêu cầu. Tiến trình sẽ được bắt đầu lại khi nó được phân bổ đủ tài nguyên yêu cầu và tái tạo lại các tài nguyên bị ngắt.

2.1.4. Loại bỏ yếu tố chờ đợi vòng tròn

Yếu tố chờ đợi vòng tròn có thể được loại bỏ bằng cách sắp xếp thứ tự các tài nguyên. Mỗi tài nguyên r được gán một số thứ tự $f(r)$.

- Phương pháp 1: Tiến trình giữ tài nguyên r_i , chỉ được phép đòi tài nguyên r_j khi $f(r_i) < f(r_j)$.

- Phương pháp 2: Tiến trình đang giữ tài nguyên r_i , muốn đòi tài nguyên r_j thì phải giải phóng các tài nguyên r_i thoả mãn điều kiện $f(r_i) > f(r_j)$

2.2. Dự báo và tránh bế tắc

Nguyên tắc chung của dự báo và tránh bế tắc là mỗi lần phân bổ tài nguyên cho các tiến trình thì hệ thống sẽ kiểm tra xem việc phân bổ đó có đẩy hệ thống vào tình trạng bế tắc hay không. Nếu có thì tìm cách giải quyết trước khi bế tắc xảy ra.

2.2.1. Khái niệm về dãy tiến trình an toàn

Cho dãy tiến trình P_1, P_2, \dots, P_n song hành. Dãy tiến trình được gọi là an toàn (safe process) nếu với mọi tiến trình P_i , tài nguyên mà P_i cần có thể được thoả mãn bởi các tài nguyên khả dụng của hệ thống và tài nguyên do các tiến trình P_i đang giữ với điều kiện $i' < i$.

Hệ thống ở trạng thái an toàn tại một thời điểm nếu dãy tiến trình song hành tại thời điểm đó có thể được sắp xếp thành một dãy an toàn.

2.2.2. Thuật toán chuyển hệ sang trạng thái an toàn

Giả sử hệ có n tiến trình và m kiểu tài nguyên. Các cấu trúc dữ liệu sử dụng trong thuật toán được xây dựng như sau:

- Available: mảng $1 \times m$ thể hiện số tài nguyên có thể sử dụng của mỗi kiểu. Nếu $\text{Available}(j) = k$ suy ra có k tài nguyên kiểu r_j có thể sử dụng.

- Max: mảng $n \times m$ thể hiện số tài nguyên cực đại mà mỗi tiến trình yêu cầu. Nếu $\text{Max}(i,j) = k$ suy ra tiến trình P_i chỉ có thể yêu cầu cực đại k tài nguyên kiểu r_j .

- Allocation: mảng $n \times m$ thể hiện số tài nguyên mỗi kiểu hiện đã phân bổ cho các tiến trình. Nếu $\text{Allocation}(i,j) = k$ suy ra tiến trình P_i đang sử dụng k tài nguyên kiểu r_j .

- Need: mảng $n \times m$ thể hiện số tài nguyên còn cần của mỗi tiến trình. Nếu $\text{Need}(i,j) = k$ suy ra tiến trình P_i còn cần k tài nguyên kiểu r_j .

Chú ý: $\text{Need}(i,j) = \text{Max}(i,j) - \text{Allocation}(i,j)$

- Request: mảng $n \times m$ thể hiện yêu cầu tài nguyên của các tiến trình tại mỗi thời điểm. Nếu $\text{Request}(i,j) = k$ suy ra tiến trình P_i đang yêu cầu k tài nguyên kiểu r_j .

* Thuật toán:

Step1: **If** $\text{Request}(i) \leq \text{Need}(i)$ **then** goto Step2 **else** error;
(tiến trình yêu cầu tài nguyên vượt quá quy định)

Step2: **If** $\text{Request}(i) \leq \text{Available}$ **then** goto Step3 **else** P_i wait;
(không đủ tài nguyên để phân bổ cho P_i)

Step3: Hệ thống dự định phân bổ tài nguyên như sau:
 $\text{Available} := \text{Available} - \text{Request}(i)$
 $\text{Allocation}(i) := \text{Allocation}(i) + \text{Request}(i)$
 $\text{Need}(i) := \text{Need}(i) - \text{Request}(i)$

Step4: Kiểm tra tính an toàn của hệ:

Nếu hệ ở trạng thái an toàn thì phân bổ tài nguyên theo dự định, ngược lại tiến trình P_i phải đợi cùng với yêu cầu tài nguyên $\text{Request}(i)$.

2.2.3. Thuật toán kiểm tra tính an toàn của hệ thống

Trong thuật toán sử dụng thêm hai cấu trúc dữ liệu sau:

Work: mảng $1 \times m$ thể hiện số tài nguyên khả dụng của hệ thống và số tài nguyên do các tiến trình P_i đang sử dụng với điều kiện $i' < i$.

Finish: mảng 1 x n đánh dấu các tiến trình đã xét.

* Thuật toán:

Step 1: Khởi tạo:

Work := Available;

Finish(i) := false; (với mọi $i = 1..n$)

Step 2: Tìm i sao cho $Finish(i) = false$ và $Need(i) \leq Work$

Nếu không tìm thấy, goto Step 4

Step 3: $Work := Work + Allocation(i);$

$Finish(i) := true;$

goto Step 2;

Step 4: Nếu $Finish(i) = true$ với mọi i thì hệ thống ở trạng thái an toàn; ngược lại hệ thống gặp bế tắc.

Ví dụ: Giả sử có 5 tiến trình và 3 kiểu tài nguyên A, B, C ở trạng thái như sau:

Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P ₀	3	0	2	7	0	3	3	3	2
P ₁	2	1	0	9	2	2			
P ₂	0	1	1	2	5	3			
P ₃	0	0	2	4	3	2			
P ₄	2	0	0	3	2	2			

a - Hệ có ở trạng thái an toàn hay không?

b - Giả sử tiến trình p₄ có yêu cầu tài nguyên (1, 2, 2). Hỏi có thể phân bổ cho p₄ được hay không?

Bài giải

a - Để xét tính an toàn của hệ, ta cần tính ma trận Need theo công thức:

$$\text{Need} = \text{Max} - \text{Allocation}$$

Process	Need		
	A	B	C
P ₀	4	0	1
P ₁	7	1	2
P ₂	2	4	2
P ₃	4	3	0
P ₄	1	2	2

Theo thuật toán kiểm tra tính an toàn của hệ ta có:

Work := Available := (3,3,2)

Finish[i] = false với i = 0,1,2,3,4

Xét:

Need[0] >= Work \Rightarrow Finish[0] = false

Need[1] >= Work \Rightarrow Finish[1] = false

Need[2] >= Work \Rightarrow Finish[2] = false

Need[3] >= Work \Rightarrow Finish[3] = false

Need[4] <= Work \Rightarrow Finish[4] = true và Work = Work + Allocation[4] = (5,3,2)

Need[0] <= Work \Rightarrow Finish[0] = true và Work = Work + Allocation[0] = (8,3,4)

Need[1] <= Work \Rightarrow Finish[1] = true và Work = Work + Allocation[1] = (10,4,4)

Need[2] <= Work \Rightarrow Finish[2] = true và Work = Work + Allocation[2] = (10,5,5)

Need[3] <= Work \Rightarrow Finish[3] = true và Work = Work + Allocation[3] = (12,5,5)

Như vậy, Finish[i] = true với i = 4,0,1,2,3 dẫn đến dãy tiến trình p₄, p₀, p₁, p₂, p₃ là dãy an toàn suy ra hệ ở trạng thái an toàn.

b - Giả sử tiến trình p₄ có yêu cầu tài nguyên (1, 2, 2). Theo thuật toán chuyển hệ sang trạng thái an toàn, ta có:

Request[4] <= Need[4] \Rightarrow thoả mãn

Request[4] <= Available \Rightarrow thoả mãn

Do đó, hệ dự định phân bổ:

Available = Available - Request[4] = (2,1,0)

Allocation[4] = Allocation[4] + Request[4] = (3,2,2)

Need[4] = Need[4] - Request[4] = (0,0,0)

và có trạng thái như sau:

Process	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
p ₀	3	0	2	4	0	1	2	1	0
p ₁	2	1	0	7	1	2			
p ₂	0	1	1	2	4	2			
p ₃	0	0	2	4	3	0			
p ₄	3	2	2	0	0	0			

Áp dụng thuật toán kiểm tra tính an toàn của hệ ta có:

Work := Available = (2,1,0)

$\text{Finish}[i] = \text{false}$ với $i = 0, 1, 2, 3, 4$

Xét

$\text{Need}[0] \geq \text{Work} \Rightarrow \text{Finish}[0] = \text{false}$

$\text{Need}[1] \geq \text{Work} \Rightarrow \text{Finish}[1] = \text{false}$

$\text{Need}[2] \geq \text{Work} \Rightarrow \text{Finish}[2] = \text{false}$

$\text{Need}[3] \geq \text{Work} \Rightarrow \text{Finish}[3] = \text{false}$

$\text{Need}[4] \leq \text{Work} \Rightarrow \text{Finish}[4] = \text{true}$ và $\text{Work} = \text{Work} + \text{Allocation}[4] = (5, 3, 2)$

$\text{Need}[0] \leq \text{Work} \Rightarrow \text{Finish}[0] = \text{true}$ và $\text{Work} = \text{Work} + \text{Allocation}[0] = (8, 3, 4)$

$\text{Need}[1] \leq \text{Work} \Rightarrow \text{Finish}[1] = \text{true}$ và $\text{Work} = \text{Work} + \text{Allocation}[1] = (10, 4, 4)$

$\text{Need}[2] \leq \text{Work} \Rightarrow \text{Finish}[2] = \text{true}$ và $\text{Work} = \text{Work} + \text{Allocation}[2] = (10, 5, 5)$

$\text{Need}[3] \leq \text{Work} \Rightarrow \text{Finish}[3] = \text{true}$ và $\text{Work} = \text{Work} + \text{Allocation}[3] = (12, 5, 5)$

Như vậy, $\text{Finish}[i] = \text{true}$ với $i = 4, 0, 1, 2, 3$ dẫn đến dãy tiến trình p_4, p_0, p_1, p_2, p_3 ở trạng thái an toàn, suy ra hệ ở trạng thái an toàn. Do đó, có thể phân bổ tài nguyên cho p_4 .

2.3. Phát hiện bế tắc

Dữ liệu như phần 2.2.3

Thuật toán:

Step 1: Tìm i sao cho $\text{Finish}(i) = \text{false}$ và $\text{Request}(i) \leq \text{Work}$

Nếu không tìm thấy goto Step 3

Step 2: $\text{Work} := \text{Work} + \text{Allocation}(i)$

$\text{Finish}(i) := \text{true}$

Goto Step 1

Step 3: Nếu tồn tại i sao cho $\text{Finish}(i) = \text{false}$ thì hệ thống gặp bế tắc.

2.4. Xử lý bế tắc

Khi hệ thống gặp bế tắc, hệ điều hành có thể áp dụng các phương pháp sau giải quyết:

- Thông báo cho operator biết để tự xử lý

- Định chỉ hoạt động của tiến trình: phương pháp này dựa trên việc thu hồi lại các tài nguyên của những tiến trình bị kết thúc. Có thể sử dụng một trong hai cách định chỉ sau:

+ Định chỉ hoạt động của mọi tiến trình trong tình trạng bế tắc.

+ Định chỉ hoạt động lần lượt của từng tiến trình cho tới khi thoát khỏi tình trạng bế tắc (khi định chỉ tiến trình nào thì thu hồi lại tài nguyên của tiến trình đó).

Chú ý: Khi định chỉ hoạt động của các tiến trình cần chú ý tới các yếu tố sau:

- + Độ ưu tiên của tiến trình
- + Tiến trình đã diễn ra bao lâu và còn bao lâu sẽ hoàn thành?
- + Có bao nhiêu kiểu tài nguyên và số lượng tài nguyên mà tiến trình đã dùng?
- + Tiến trình còn cần bao nhiêu tài nguyên để hoàn thành một công việc?

- Thu hồi tài nguyên: áp dụng biện pháp ngắt tài nguyên từ một số tiến trình để cấp phát cho các tiến trình đang có nhu cầu, sau đó kiểm tra lại tình trạng bế tắc. Phương pháp này cần phải:

- + Xem xét và lựa chọn tiến trình nào để ngắt tài nguyên và ngắt tài nguyên nào?
- + Khả năng phục hồi lại trạng thái ban đầu của tiến trình có thực hiện được hay không?
- + Có thể xảy ra khả năng một số tiến trình không bao giờ được cấp đủ tài nguyên không?

2.5. Kết luận chung về phòng tránh bế tắc

Không thể dùng một phương pháp xử lý bế tắc chung cho toàn bộ các tình huống. Cần phải chia các tình huống thành các lớp khác nhau và mỗi lớp thì áp dụng một phương pháp xử lý thích hợp.

Ví dụ:

- Tài nguyên hệ thống (Internal Resource): dùng biện pháp ngăn ngừa bằng cách sắp thứ tự tài nguyên.
- Bộ nhớ trong (RAM): dùng biện pháp ngăn ngừa bằng cách ngắt tài nguyên.
- Các thiết bị có thể phân bổ được: dùng các thuật toán tránh bế tắc.
- Bộ nhớ đệm: dùng phương pháp phân bổ trước.

Câu hỏi và bài tập

1. Nêu khái niệm về tiến trình và mối quan hệ giữa các tiến trình trong hệ thống. Trình bày đặc trưng của các mối quan hệ đó.
2. Nêu ý nghĩa các trạng thái của một tiến trình. Trình bày nội dung của khối điều khiển tiến trình.
3. Trình bày khái niệm về tài nguyên găng và đoạn tới hạn (critical section), từ đó nêu mục tiêu của quản lý tiến trình.

4. Nêu khái niệm bài toán đoạn tới hạn và trình bày nguyên tắc chung của các phương pháp giải quyết bài toán đoạn tới hạn.

5. Trình bày nguyên tắc của các phương pháp giải quyết bài toán đoạn tới hạn.

6. Xây dựng thuật toán giải quyết bài toán đoạn tới hạn theo phương pháp khoá trong, kiểm tra và xác lập, dùng đèn hiệu trong trường hợp có ba hoặc nhiều tiến trình.

7. Nêu khái niệm về bế tắc và các điều kiện để xảy ra bế tắc trong hệ.

8. Trình bày nguyên tắc chung và các biện pháp phòng ngừa bế tắc,

9. Nêu khái niệm về dây tiến trình an toàn và các thuật toán dự báo, tránh bế tắc

10. Trình bày các biện pháp giải quyết khi hệ thống gặp bế tắc. Cho ví dụ đối với một hệ điều hành cụ thể.

11. Cho một dây tiến trình và 4 kiểu tài nguyên được phân bổ như sau:

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P ₀	0	0	1	2	0	0	1	2	1	5	2	0
P ₁	1	0	0	0	1	7	5	0				
P ₂	1	3	5	4	2	3	5	6				
P ₃	0	6	3	2	0	6	5	2				
P ₄	0	0	1	4	0	6	5	6				

a - Tính số tài nguyên mà các tiến trình còn cần.

b - Hệ có ở trạng thái an toàn hay không?

c - Giả sử tiến trình p₁ có yêu cầu tài nguyên (0,4,2,0), có thể phân bổ cho tiến trình p₁ hay không?

12. Giả sử cho dây 5 tiến trình, 3 kiểu tài nguyên A, B, C và yêu cầu tài nguyên tương ứng của các tiến trình như sau:

Process	Allocation			Max			Available			Request		
	A	B	C	A	B	C	A	B	C	A	B	C
P ₀	0	1	0	7	5	3	9	6	5	1	2	0
P ₁	2	0	0	3	2	2				1	0	2
P ₂	3	0	2	9	0	2				2	0	0
P ₃	2	1	1	2	2	2				0	0	1
P ₄	0	0	2	4	3	3				2	2	0

Hỏi có thể phân bổ tài nguyên cho các tiến trình được không?

Chương 3

LẬP LỊCH CHO CPU

Mục tiêu

Sau chương này, người học có thể hình dung được tầm quan trọng của "giờ CPU" và các phương pháp, các thuật toán lập lịch cho CPU của hệ điều hành nhằm đáp ứng yêu cầu được phục vụ của các tiến trình. Đồng thời, người học cũng biết được một công cụ để CPU có thể điều khiển hoạt động của các tiến trình một cách chính xác, đó là ngắt.

Nội dung

Trình bày các khái niệm cơ bản về giờ CPU, các thuật toán lập lịch cho CPU, khái niệm về ngắt và phương pháp xử lý ngắt của hệ điều hành.

I. CÁC KHÁI NIỆM CƠ BẢN

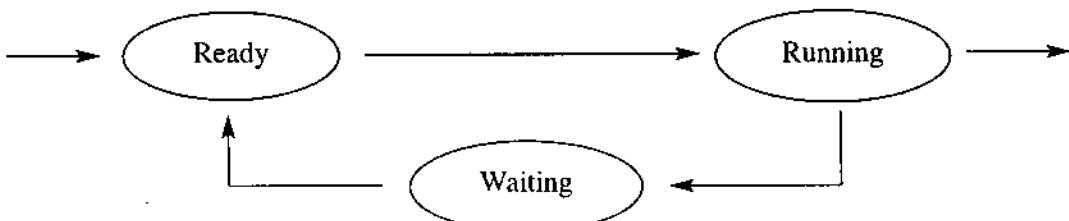
1. Khái niệm giờ CPU

CPU là một loại tài nguyên quan trọng của máy tính. Mọi tiến trình muốn hoạt động được đều phải có sự phục vụ của CPU (để xử lý, tính toán...). Thời gian mà CPU phục vụ cho tiến trình hoạt động được gọi là giờ CPU.

Tại mỗi thời điểm nhất định, chỉ có một tiến trình được phân phối giờ CPU để hoạt động (thực hiện các lệnh của mình).

2. Các trạng thái của tiến trình liên quan đến giờ CPU

Trong chế độ đa chương trình, có ba trạng thái của tiến trình liên quan mật thiết đến giờ CPU bao gồm:



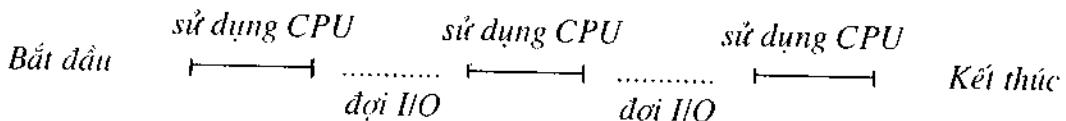
Hình 3.1. Các trạng thái của tiến trình liên quan đến giờ CPU

- Sẵn sàng (ready): là trạng thái mà tiến trình được phân phối đầy đủ mọi tài nguyên cần thiết và đang chờ giờ CPU.

- Thực hiện (running): là trạng thái mà tiến trình được phân phối đầy đủ mọi tài nguyên cần thiết và giờ CPU.

- Đợi (waiting): là trạng thái tiến trình không thực hiện được vì thiếu một vài điều kiện nào đó (đợi dữ liệu vào/ra, đợi tài nguyên bổ sung...). Khi sự kiện mà nó chờ đợi xuất hiện, tiến trình sẽ quay trở lại trạng thái sẵn sàng.

Như vậy, trong suốt thời gian tồn tại của mình, các tiến trình sẽ tuân thủ theo sơ đồ thực hiện sau:



Hình 3.2. Sơ đồ thực hiện tiến trình

Một tiến trình đang trong trạng thái thực hiện, nó có thể rời khỏi trạng thái bởi một trong ba lý do:

- Tiến trình đã hoàn thành công việc, khi đó nó trả lại giờ CPU và chuyển sang chờ xử lý kết thúc.

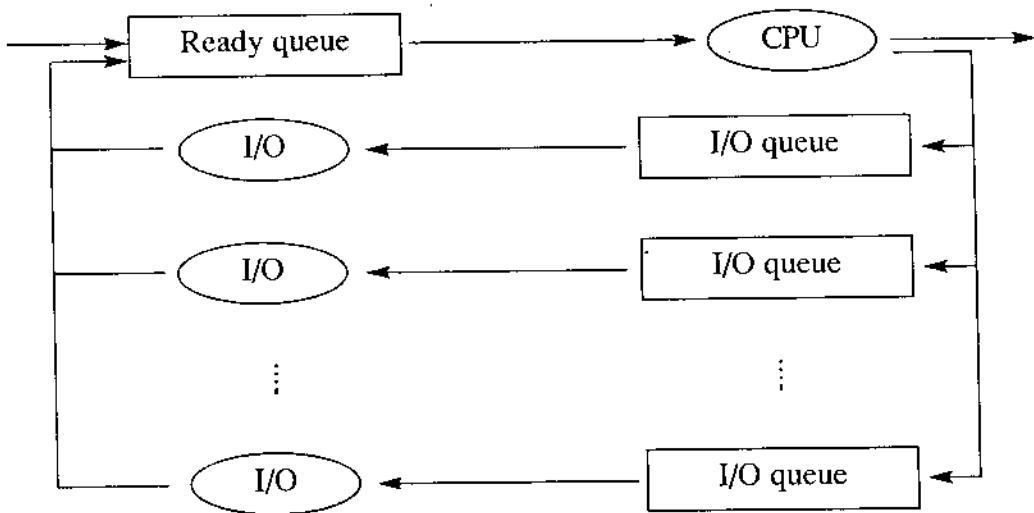
- Tiến trình tự ngắt: Khi tiến trình chờ đợi một sự kiện nào đó, tiến trình sẽ được chuyển sang trạng thái thực hiện khi xuất hiện sự kiện nó đang chờ.

- Tiến trình sử dụng hết giờ CPU dành cho nó, khi đó nó sẽ được chuyển sang trạng thái sẵn sàng.

Việc chuyển tiến trình sang trạng thái sẵn sàng về bản chất là thực hiện việc phân phối lại giờ CPU.

3. Khái niệm lập lịch cho CPU

Để điều khiển tiến trình ở nhiều trạng thái khác nhau, hệ thống thường tổ chức các từ trạng thái (thực chất là các khối điều khiển tiến trình) để ghi nhận tình trạng sử dụng tài nguyên và trạng thái tiến trình. Các từ trạng thái được tổ chức theo kiểu hàng đợi như sau:



Hình 3.3 - Sơ đồ tổ chức hàng đợi các tiến trình

Như vậy, lập lịch cho CPU có nghĩa là tổ chức một hàng đợi các tiến trình sẵn sàng để phân phối giờ CPU cho chúng dựa trên độ ưu tiên của các tiến trình; sao cho hiệu suất sử dụng CPU là tối ưu nhất.

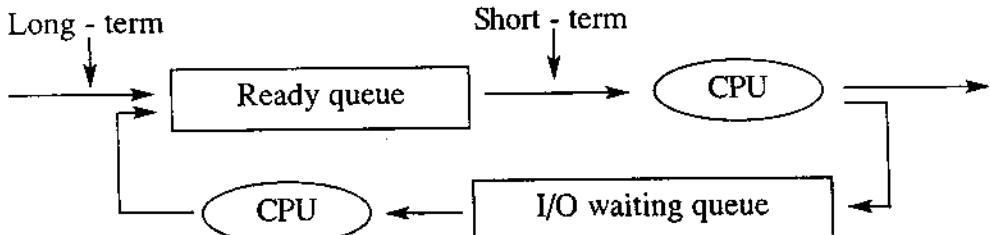
Mỗi tiến trình ở trạng thái sẵn sàng được gắn với một thứ tự ưu tiên. Thứ tự ưu tiên này được xác định dựa vào các yếu tố như: thời điểm hình thành tiến trình, thời gian thực hiện tiến trình, thời gian kết thúc tiến trình...

4. Các phương pháp lập lịch và yếu tố đánh giá

Vì các tiến trình diễn ra trong hệ thống là ngẫu nhiên, do đó không có thuật toán lập lịch tối ưu mà chỉ có những thuật toán tốt đối với những dạng tiến trình nào đó. Các phương pháp lập lịch thường được áp dụng là:

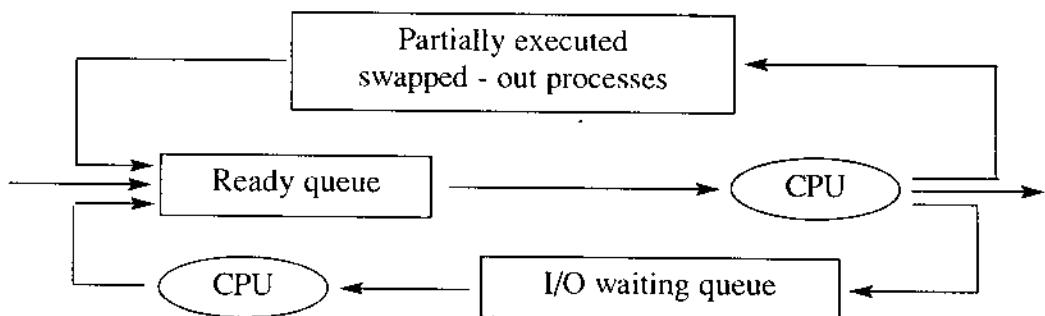
- Long-term scheduler: áp dụng với những tiến trình đã được lập danh sách và SPOOL.

- Short-term scheduler: áp dụng với những tiến trình mà mã nguồn của nó đã được đẩy vào bộ nhớ trong (tiến trình đã sẵn sàng nhận giờ phân bổ của CPU).



Hình 3.4 - Phạm vi áp dụng của Long - term scheduler và Short - term scheduler

- Medium-term scheduler: áp dụng với những tiến trình có thao tác swaping (những tiến trình đã được thực hiện một phần sau đó đưa ra ngoài...).



Hình 3.5 - Phạm vi áp dụng của Medium - term scheduler

Yếu tố để đánh giá các phương pháp lập lịch:

- Sự công bằng: mỗi tiến trình dù sớm hay muộn cũng phải được phân phối giờ CPU.

- Tận dụng giờ CPU: thời gian vô ích của CPU càng ít càng tốt. Khi đó hệ số throughput của hệ thống cao (số lượng tiến trình được phục vụ trong một đơn vị thời gian nhiều).

- Tổng thời gian thực hiện tiến trình (Turn Around Time): được tính từ khi tiến trình bắt đầu cho tới khi tiến trình kết thúc.

- Thời gian tiến trình chờ được xử lý trong hàng đợi (Wait Time).

- Thời gian đáp ứng (Response Time): khi tiến trình hoạt động trong hệ thống, nó cần dùng giờ CPU nhiều lần. Mỗi lần cần dùng giờ CPU tiến trình sẽ đưa ra một yêu cầu, như vậy thời gian tính từ khi tiến trình có yêu cầu giờ CPU tới khi nó được hệ thống phân bổ gọi là thời gian đáp ứng.

Chú ý: Việc lập lịch cho CPU liên quan đến việc tổ chức hàng đợi các khối điều khiển tiến trình. Do đó, cần tổ chức hàng đợi sao cho các yếu tố trên là tối ưu nhất.

II. CÁC THUẬT TOÁN LẬP LỊCH

1. First Come First Served (FCFS)

Trong thuật toán này, độ ưu tiên phục vụ tiến trình căn cứ vào thời điểm hình thành tiến trình. Hàng đợi các tiến trình được tổ chức theo kiểu FIFO. Mọi tiến trình đều được phục vụ theo trình tự xuất hiện cho đến khi kết thúc hoặc bị ngắt.

Ưu điểm của thuật toán này là giờ CPU không bị phân phối lại (không bị ngắt) và chi phí tổ chức thực hiện thấp nhất (vì không phải thay đổi thứ tự ưu tiên phục vụ, thứ tự ưu tiên là thứ tự của tiến trình trong hàng đợi).

Nhược điểm của thuật toán là thời gian trung bình chờ phục vụ của các tiến trình là như nhau (không kể tiến trình ngắn hay dài), do đó dẫn tới ba nhược điểm sau:

- Thời gian chờ đợi trung bình sẽ tăng vô hạn khi hệ thống tiếp cận tới hạn khả năng phục vụ của mình.
- Nếu độ phát tán thời gian thực hiện tiến trình tăng thì thời gian chờ đợi trung bình cũng tăng theo.
- Khi có tiến trình dài, ít bị ngắt thì các tiến trình khác phải chờ đợi lâu hơn.

Ví dụ: Cho dãy tiến trình và thời gian phục vụ tương ứng:

Processes	T thực hiện
P ₁	24
P ₂	3
P ₃	3

Sơ đồ Gantt biểu thị thứ tự thực hiện các tiến trình như sau:



Theo sơ đồ này, chúng ta thấy thời gian chờ đợi của các tiến trình là:

Processes	T thực hiện	T đợi
P ₁	24	0
P ₂	3	24
P ₃	3	27
Σ	30	51

dẫn đến thời gian chờ đợi trung bình của các tiến trình là $51/3=17$

2. Shortest Job First (SJF)

Thuật toán SJF xác định thứ tự ưu tiên thực hiện tiến trình dựa vào tổng thời gian thực hiện tiến trình. Tiến trình nào có tổng thời gian thực hiện ngắn sẽ được

ưu tiên phục vụ trước.

Ví dụ: Cho dãy tiến trình và thời gian phục vụ tương ứng:

Processes	T thực hiện
P ₁	24
P ₂	3
P ₃	3

Sơ đồ Gantt biểu thị thứ tự thực hiện các tiến trình như sau:



Theo sơ đồ này, chúng ta thấy thời gian chờ đợi của các tiến trình là:

Processes	T thực hiện	T đợi
P ₂	3	0
P ₃	3	3
P ₁	24	6
Σ	30	9

dẫn đến thời gian chờ đợi trung bình của các tiến trình là $9/3=3$

Ưu điểm của thuật toán là thời gian chờ đợi trung bình của các tiến trình ngắn hơn so với FCFS. SJF nhanh chóng loại bỏ các tiến trình ngắn, giảm số lượng các tiến trình trong hàng đợi.

Nhược điểm chính của thuật toán là chế độ phân phối lại giờ CPU cũng được áp dụng trong trường hợp ngắt các tiến trình dài đang thực hiện để phục vụ các tiến trình ngắn hơn mới xuất hiện trong hàng đợi. Nếu tiến trình mới xuất hiện có tổng thời gian thực hiện ngắn nhưng vẫn lớn hơn thời gian cần thiết để thực hiện nốt tiến trình đang thực hiện thì việc ngắt tiến trình là không hợp lý.

3. Shortest Remain Time (SRT)

Tương tự như SJF nhưng trong thuật toán này, độ ưu tiên thực hiện các tiến trình dựa vào thời gian cần thiết để thực hiện nốt tiến trình (bằng tổng thời gian trừ đi thời gian đã thực hiện). Như vậy, trong thuật toán này cần phải thường xuyên cập nhật thông tin về thời gian đã thực hiện tiến trình. Đồng thời, chế độ phân bổ lại giờ CPU cũng phải được áp dụng nếu không sẽ làm mất tính ưu việt của thuật toán.

4. Round Robin (RR)

Trong thuật toán này, hệ thống quy định một lượng tử thời gian (time quantum) khoảng từ 10 – 100 mili giây (ms).

Mỗi tiến trình trong hàng đợi lần lượt được phân phối một lượng tử thời gian để thực hiện. Sau khoảng thời gian đó, nếu tiến trình chưa kết thúc hoặc không rời vào trạng thái đợi thì nó được chuyển về cuối hàng đợi.

Hàng đợi các tiến trình được tổ chức theo kiểu vòng tròn và các tiến trình luôn luôn đảm bảo được phục vụ. Khi có tiến trình mới phát sinh, nó sẽ được đưa vào hàng đợi vòng tròn và được đặt ở vị trí phục vụ ngay. Các tiến trình dù ngắn hay dài đều có độ ưu tiên phục vụ như nhau.

Trên thực tế, để đảm bảo độ ưu tiên cho các tiến trình dài, hệ thống sẽ phân chia các tiến trình thành m lớp. Số lần được phục vụ và thời gian một lần phục vụ tiến trình tại mỗi lớp khác nhau (giả sử ở lớp thứ i, tiến trình được phục vụ k_i lần và mỗi lần với thời gian q_i).

Nếu sau khoảng thời gian đã được phân phối mà tiến trình chưa kết thúc hoặc không bị ngắt thì nó được chuyển sang lớp thứ $i + 1$ (với k_{i+1} và q_{i+1} lớn hơn). Lượng tử thời gian sẽ tăng dần cho đến khi tiến trình rời vào lớp ngoài cùng (lớp m). Ở đó nó sẽ được phục vụ với lượng tử q_m không đổi. Như vậy thứ tự ưu tiên của các tiến trình sẽ tăng dần theo thời gian xếp hàng đợi.

Ưu điểm của phương pháp phục vụ đồng mức theo lớp sẽ cho phép hệ thống ưu tiên những tiến trình ngắn (vì nó kết thúc sớm) nhưng không gây tổn hại lớn cho các tiến trình dài.

Nhược điểm là do phải thường xuyên phân phối lại giờ CPU nên thời gian chờ đợi trung bình của Round Robin có thể lớn hơn so với FCFS.

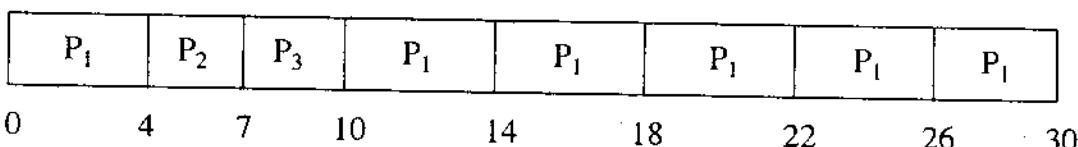
Chú ý: Trong thuật toán, cần chọn giá trị lượng tử thời gian (time quantum) thích hợp. Nếu chọn giá trị time quantum lớn thì việc bổ sung tiến trình mới hoặc kích hoạt tiến trình bị ngắt sẽ làm tăng thời gian chờ đợi trung bình, nhưng ngược lại nếu chọn giá trị time quantum nhỏ thì nó sẽ làm cho các tiến trình phải liên tục chuyển trạng thái dẫn đến giảm hiệu số hữu ích của CPU.

Thông thường giá trị time quantum được chọn theo công thức: $q = t/n$ hoặc $q = t/n - s$. Trong đó: t là thời gian không chế trước; n là số tiến trình; s là thời gian chuyển từ tiến trình này sang tiến trình khác.

Ví dụ: Cho dãy tiến trình với thời gian thực hiện tương ứng theo bảng dưới và time quantum có giá trị $q = 4$.

Processes	T thực hiện
P ₁	24
P ₂	3
P ₃	3

Sơ đồ Gantt biểu thị thứ tự thực hiện các tiến trình như sau:



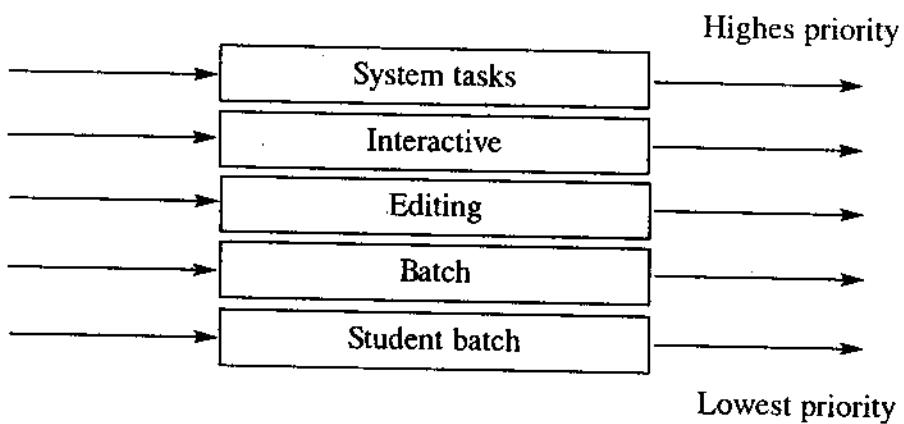
Theo sơ đồ này, chúng ta thấy thời gian chờ đợi của các tiến trình như sau:

Processes	T thực hiện	T đợi
P ₁	24	6
P ₃	3	4
P ₁	3	7
Σ	30	17

dẫn đến thời gian chờ đợi trung bình của các tiến trình là $17/3 = 5.66$.

5. Multi Level Queue (MLQ)

Thuật toán dựa vào thông tin do người sử dụng cung cấp và kết quả phân tích của hệ thống để phân chia các tiến trình sẵn sàng thành các mức hàng đợi có độ ưu tiên khác nhau.



Hình 3.6. Tổ chức hàng đợi các tiến trình trong MLQ

Trong mỗi mức có cách tổ chức và thực hiện thích hợp. Khi có một tiến trình ở mức cao hơn cần thực hiện thì hệ thống phải ngắt tiến trình ở các mức thấp hơn để phục vụ tiến trình mức cao.

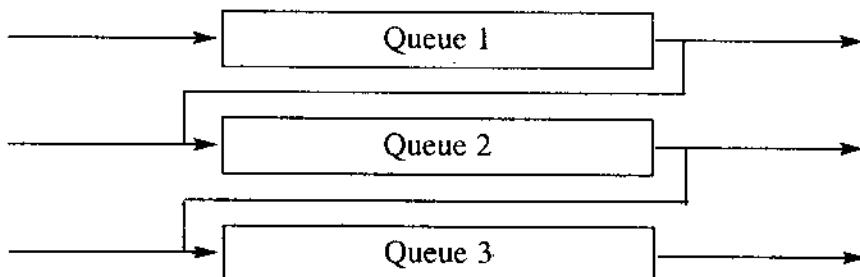
Chú ý: Trên thực tế, cách tổ chức hai hàng đợi được áp dụng phổ biến nhất, trong đó:

- Một hàng đợi cho các tiến trình trao đổi vào/ra nhiều.
- Một hàng đợi cho các tiến trình thực hiện tính toán nhiều.

Các tiến trình ở mức hàng đợi thứ nhất có độ ưu tiên cao hơn nhưng vì phải chờ kết quả vào/ra nên không sử dụng hết hiệu suất giờ CPU được phân bổ. Các tiến trình ở hàng đợi thứ hai hoạt động tính toán dựa trên nền của các tiến trình ở mức thứ nhất. Sau khi thực hiện, tiến trình ở mức nào phải quay trở về đúng mức đó nếu chưa kết thúc.

6. Multi Level Feedback Queues (MLFQ)

MLFQ là thuật toán tổng quát nhất để lập lịch cho CPU. Trong thuật toán này, hệ thống tổ chức nhiều hàng đợi. Mỗi hàng đợi có độ ưu tiên và cách thức tổ chức thực hiện khác nhau. Các tiến trình có thể chuyển đổi từ hàng đợi này sang hàng đợi khác.



Hình 3.7. Tổ chức hàng đợi các tiến trình trong MLFQ

Các yếu tố cần chú ý:

- Số lượng hàng đợi sao cho hợp lý.
- Chọn thuật toán thích hợp cho mỗi hàng đợi.
- Tiêu chuẩn và cách thức hạ mức một tiến trình.
- Tiêu chuẩn và cách thức nâng mức một tiến trình.
- Tiêu chuẩn và cách thức để chỉ định một tiến trình mới phát sinh đi vào hàng đợi nào.

III. NGẮT

1. Khái niệm ngắt (Interrupt)

Để tiến trình có thể thực hiện chính xác, cần phải có sự phối hợp nhịp nhàng giữa hoạt động của CPU và các thiết bị. Ngắt là phương tiện để các thiết bị thông báo cho CPU biết việc thay đổi trạng thái của mình.

Từ góc độ CPU, ta có thể coi ngắt là việc ngừng đột xuất việc thực hiện một tiến trình để chuyển sang thực hiện một tiến trình khác khi có một sự kiện nào đó xảy ra.

Như vậy, ngắt là công cụ để chuyển điều khiển tới một tiến trình khác mà tiến trình hiện tại không biết.

2. Phân loại ngắt

Ngắt được chia thành hai loại: ngắt trong và ngắt ngoài.

Ngắt trong là ngắt gây ra bởi các sự kiện liên quan đến hoạt động của CPU. Ví dụ như các sự kiện tràn ô nhớ, thực hiện phép chia cho 0, vi phạm địa chỉ bộ nhớ, mã lệnh sai...

Ngắt ngoài là ngắt gây ra bởi các sự kiện nằm ngoài tiến trình đang thực hiện như: tín hiệu đồng hồ, sự cố kỹ thuật, ngắt vào/ra...

3. Quy trình xử lý ngắt

Vấn đề quan trọng trong xử lý ngắt là ghi nhận thời điểm xảy ra ngắt. Nếu xử lý ngắt ngay lập tức (không chờ thực hiện xong câu lệnh hiện thời) cho phép hệ thống giải quyết được các yêu cầu cấp bách tự nhiên nhưng sẽ làm mất câu lệnh hiện thời. Thuận lợi hơn là chờ câu lệnh hiện thời kết thúc rồi mới xử lý ngắt, như vậy hệ thống cần định kỳ kiểm tra xem có tín hiệu ngắt xuất hiện hay không? Cơ chế ghi nhận ngắt này nằm ngoài các chương trình xử lý ngắt và phải được lưu ý khi xây dựng các chương trình hệ thống.

Có rất nhiều phương pháp liên quan đến xử lý ngắt nhưng quy trình chung có thể mô tả gồm năm bước:

- Ghi nhận đặc trưng của sự kiện gây ra ngắt vào ô nhớ quy định.
- Ghi nhận trạng thái của tiến trình bị ngắt (bộ đếm chương trình, nội dung các thanh ghi, chế độ làm việc...).
- Chuyển địa chỉ chương trình xử lý ngắt vào thanh ghi địa chỉ lệnh của CPU.

- Thực hiện chương trình xử lý sự kiện.
- Khôi phục lại tiến trình bị ngắt.

Ba bước đầu của quy trình xử lý ngắt do các thành phần kỹ thuật của máy tính đảm nhận, hai bước còn lại do hệ điều hành đảm nhận. Cụ thể như sau:

Hệ điều hành xử lý sự kiện bằng các chương trình xử lý ngắt, mỗi loại sự kiện có một chương trình xử lý riêng. Việc đầu tiên của chương trình xử lý ngắt là lưu lại các thông tin cụ thể về trạng thái của tiến trình bị ngắt (các thông tin lưu lại ở bước hai do phần cứng đảm nhận chỉ là các thông tin chung). Phần tiếp theo là đoạn chương trình xử lý sự kiện, mỗi sự kiện đòi hỏi một cách xử lý khác nhau. Nếu sự kiện không đòi hỏi xử lý gấp thì hệ thống có thể đưa tiến trình xử lý vào hàng đợi.

Khi khôi phục trạng thái tiến trình bị ngắt, nếu tiến trình ngắt không thể tiếp tục thực hiện vì sự kiện xảy ra thì sau khi thông báo nguyên nhân, chương trình xử lý ngắt chuyển tiến trình sang bộ phận xử lý kết thúc.

Chú ý: Một số hoạt động trong xử lý ngắt phải được thực hiện ngay lập tức như nhớ trạng thái tiến trình, xử lý lỗi,... Chương trình thực hiện các thao tác này phải thường trú trong bộ nhớ và là một thành phần của hệ điều hành. Những hoạt động xử lý ít cấp thiết hơn do các chương trình không thường trú đảm nhận. Như vậy, bản thân xử lý ngắt thường được chia thành hai mức: mức một và mức hai, tuy nhiên không phải sự kiện nào cũng đòi hỏi cả hai mức xử lý.

Những ngắt liên quan đến điều khiển và hoạt động của CPU thường được chuyển tới chương trình ghi nhận ngắt ngay lập tức để kích hoạt các chương trình xử lý ngắt tương ứng. Vì vậy trong hàng đợi luôn luôn có các tiến trình hệ thống đảm bảo điều phối, thu thập thông tin thống kê, phục vụ xử lý ngắt mức hai... Khi có một CPU thì bản thân các tiến trình hệ thống nhiều khi cũng phải xếp hàng chờ xử lý.

4. Ngắt kép

Ở phần trên, chúng ta mới chỉ xét trường hợp các sự kiện gây ngắt xảy ra một cách riêng biệt. Trong trường hợp các sự kiện xảy ra đồng thời hoặc sự kiện gây ngắt xuất hiện ngay trong tiến trình xử lý ngắt, khi đó hệ thống sẽ gấp một ngắt kép.

Để xử lý ngắt kép, cần gán cho mỗi ngắt một thứ tự ưu tiên, ngắt nào có thứ tự ưu tiên cao sẽ được xử lý trước.

Một phương pháp xử lý ngắt kép khác hay được áp dụng là phương pháp tổ

chức Stack. Trong trường hợp này, ngắt có thể được mở hay bị che. Khi bị che thì tín hiệu ngắt không thể tác động tới CPU. Ngắt bị che có thể bị mất hoặc được xếp hàng chờ xử lý. Nếu được xếp hàng thì sau khi có lệnh mở che, hệ thống sẽ xử lý các ngắt xếp hàng.

Nếu ngắt mới xuất hiện ngay trong lúc xử lý ngắt thì nó có thể bị che hay được xử lý ngay tùy theo quan hệ của hai ngắt. Nếu được xử lý ngay thì chương trình xử lý ngắt cũ trở thành chương trình bị ngắt, còn nếu bị che (chưa được xử lý) thì ngắt mới có thể bị mất hoặc xếp vào hàng đợi. Khi chương trình xử lý ngắt rơi vào trạng thái đợi (bị ngắt) thì nó sẽ phải ở đó cho đến khi chương trình điều khiển chuyển nó sang trạng thái sẵn sàng (điều này xảy ra khi CPU gặp lệnh mở chắn).

Để hệ thống có thể hoạt động được bình thường, một số tín hiệu ngắt phải tới được CPU bất kỳ thời điểm nào (ví dụ như tín hiệu khi có sự cố nghiêm trọng hoặc tín hiệu nhịp đồng hồ). Việc che ngắt hoặc mở che có thể thực hiện bằng các công cụ phần cứng hoặc các chương trình phần mềm do các chương trình xử lý ngắt quyết định. Khi đang tiến hành xử lý ngắt nào đó, nếu xuất hiện tín hiệu ngắt cùng loại thì tín hiệu đó phải bị che để tránh ghi đè thông tin vào cùng một bộ nhớ dành cho loại ngắt đó. Sau khi xử lý xong ngắt thứ nhất, hệ thống phải mở che để xử lý tiếp ngắt đang bị che.

Nếu có thiết bị kỹ thuật hỗ trợ phân phối bộ nhớ khác nhau cho từng mức ưu tiên thì việc xử lý ngắt kép bằng phương pháp gán mức ưu tiên sẽ đơn giản hơn rất nhiều.

Xử lý ngắt không nhất thiết phải tiến hành liên tục. Ví dụ xử lý ngắt vào/ra có thể chia thành hai giai đoạn: giai đoạn một xử lý kết quả các phép kiểm tra ở mức kênh và thiết bị; giai đoạn hai xử lý kết quả của chính phép vào/ra. Trong khoảng thời gian giữa hai giai đoạn, hệ thống có thể xử lý các ngắt khác nếu có.

Câu hỏi và bài tập

1. Các trạng thái nào của tiến trình liên quan đến giờ CPU.
2. Khái niệm về lập lịch cho CPU, các phương pháp lập lịch và các tiêu chuẩn đánh giá.
3. Trình bày các thuật toán lập lịch cho CPU.
4. Khái niệm, phân loại ngắt và quy trình xử lý ngắt.

5. Cho dãy tiến trình với thời gian thực hiện tương ứng như sau:

Process	T thực hiện
p ₁	10
p ₂	2
p ₃	7
p ₄	1
p ₅	5

a - Vẽ sơ đồ Grant theo các thuật toán: FCFS, SJF, RR ($q = 2$)

b - Cho biết thời gian tồn tại trong hệ thống của các tiến trình trong ở mỗi thuật toán.

c - Tính thời gian chờ đợi trung bình của các tiến trình trong các thuật toán. Thuật toán nào có thời gian chờ đợi trung bình ngắn nhất.

Chương 4

QUẢN LÝ BỘ NHỚ TRONG

Mục tiêu

Sau chương này, người học có thể hiểu được các khái niệm về địa chỉ logic, địa chỉ vật lý khi thực hiện chương trình. Nắm bắt được các cấu trúc cơ bản của chương trình và các sơ đồ quản lý bộ nhớ của hệ điều hành. Ngoài ra, còn có thể biết được khái niệm về bộ nhớ ảo và nguyên tắc hoạt động của nó.

Nội dung

Trình bày các yêu cầu của quản lý bộ nhớ, các cấu trúc cơ bản của chương trình và các sơ đồ quản lý bộ nhớ của hệ điều hành.

Khái niệm bộ nhớ ảo và phương pháp cài đặt bộ nhớ ảo.

I. CÁC KHÁI NIỆM CƠ BẢN

1. Yêu cầu của quản lý bộ nhớ trong

Bộ nhớ là thiết bị lưu trữ duy nhất mà thông qua đó CPU có thể trao đổi thông tin với môi trường bên ngoài. Do vậy, nhu cầu tổ chức, quản lý bộ nhớ là một trong những nhiệm vụ cơ bản hàng đầu của hệ điều hành. Bộ nhớ được tổ chức như mảng một chiều các từ nhớ (word), mỗi từ nhớ có một địa chỉ. Việc trao đổi thông tin với môi trường bên ngoài được thực hiện thông qua các thao tác đọc/ghi dữ liệu vào một địa chỉ cụ thể nào đó trong bộ nhớ.

Hệ điều hành chịu trách nhiệm cấp phát không gian nhớ cho các tiến trình khi có yêu cầu. Để thực hiện tốt nhiệm vụ này, hệ điều hành cần phải xem xét một số khía cạnh sau:

- Sự tương ứng giữa địa chỉ logic và địa chỉ vật lý: làm thế nào để chuyển đổi một địa chỉ logic thành một địa chỉ vật lý?

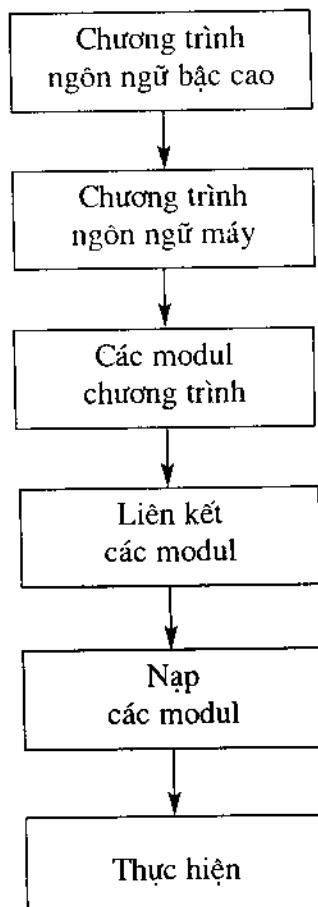
- Quản lý bộ nhớ vật lý bao gồm: phân bổ không gian nhớ cho các tiến trình hoạt động; thu hồi không gian nhớ khi tiến trình kết thúc; quản lý được không gian nhớ tự do.

- Chia sẻ thông tin: cho phép các tiến trình đang hoạt động trong bộ nhớ có thể chia sẻ thông tin với nhau.

- Bảo vệ bộ nhớ: ngăn chặn các tiến trình xâm phạm đến vùng nhớ được cấp phát cho các tiến trình khác.

2. Quá trình sinh địa chỉ (gán địa chỉ)

Thông thường, một chương trình được lưu trữ trên các thiết bị nhớ ngoài như một tệp tin nhị phân có thể xử lý. Để thực hiện chương trình, chúng ta cần nạp nó vào bộ nhớ để tạo lập các tiến trình tương ứng và xử lý. Các địa chỉ trong chương trình nguồn là các địa chỉ tương đối, vì thế một chương trình khi thực hiện sẽ phải trải qua nhiều giai đoạn xử lý để chuyển đổi các địa chỉ tương đối thành các địa chỉ tuyệt đối trong bộ nhớ. Làm rõ hơn về vấn đề này, chúng ta xét quy trình thực hiện một chương trình trên máy tính qua sơ đồ sau:



Hình 4.1 - Quá trình sinh địa chỉ

Trong quá trình dịch, các biến ngoài được chuyển đổi thành các biến trong. Hệ thống sẽ phân phối không gian nhớ và gán địa chỉ cho các biến (địa chỉ xác định mối quan hệ giữa không gian nhớ và các biến). Các địa chỉ ở đây được gọi là địa chỉ tương đối - địa chỉ logic (vì nó chưa được gắn với một bộ nhớ cụ thể nào).

Chương trình Link sẽ mốc nối các modul thành một chương trình hoàn chỉnh, có thể thực hiện được theo một nguyên tắc chung.

Khi thực hiện, chương trình được nạp vào một bộ nhớ cụ thể. Các địa chỉ tương đối sẽ được ánh xạ tới các địa chỉ tuyệt đối - địa chỉ vật lý xác định sao cho phù hợp với môi trường. Thao tác này được gọi là định vị chương trình.

Từ đây, chúng ta có các khái niệm:

- Địa chỉ logic - địa chỉ tương đối: là địa chỉ do hệ thống tạo ra và được cấp phát cho các biến khi dịch chương trình.

- Địa chỉ vật lý - địa chỉ tuyệt đối: là địa chỉ cụ thể trong bộ nhớ, được cấp phát cho các biến khi thực hiện chương trình.

- Bộ nhớ logic: là tập hợp tất cả các địa chỉ logic phát sinh khi dịch chương trình.

- Bộ nhớ vật lý: là tập hợp tất cả các địa chỉ vật lý tương ứng với các địa chỉ logic khi thực hiện chương trình.

- Chế độ lập trình trong bộ nhớ thực: cần xác định trước kích thước bộ nhớ logic không vượt quá kích thước bộ nhớ vật lý.

- Chế độ lập trình trong bộ nhớ ảo: kích thước bộ nhớ logic không bị phụ thuộc kích thước bộ nhớ vật lý.

Chú ý:

- Để thực hiện việc chuyển đổi địa chỉ logic thành địa chỉ vật lý vào thời điểm xử lý, các hệ điều hành sử dụng một cơ chế phần cứng MMU (Memory Management Unit).

- Chương trình của người sử dụng chỉ thao tác trên các địa chỉ logic chứ không thao tác với các địa chỉ vật lý. Địa chỉ vật lý chỉ được xác định khi thực hiện truy xuất dữ liệu.

II. CÁC CẤU TRÚC CƠ BẢN CỦA CHƯƠNG TRÌNH

Có nhiều phương pháp tổ chức chương trình ở bộ nhớ trong để thực hiện. Các phương pháp này khác nhau ở kiểu định vị chương trình trong bộ nhớ và

thời điểm thực hiện phép ánh xạ địa chỉ tương đối thành địa chỉ tuyệt đối.

Ví dụ:

- Nếu tại thời điểm biên dịch có thể biết vị trí mà chương trình sẽ được định vị vào bộ nhớ thì trình biên dịch có thể phát sinh ngay mã với các địa chỉ tuyệt đối. Tuy nhiên, nếu sau này có sự thay đổi vị trí định vị thì sẽ phải biên dịch lại chương trình.

- Nếu tại thời điểm biên dịch chưa thể biết vị trí định vị, trình biên dịch sẽ phát sinh mã tương đối. Khi nạp chương trình, các địa chỉ tương đối mới được ánh xạ tới các địa chỉ tuyệt đối. Như vậy, nếu thay đổi vị trí định vị, hệ thống chỉ cần nạp lại chương trình và tính toán lại các địa chỉ tuyệt đối mà không cần phải biên dịch lại.

- Nếu có nhu cầu di chuyển chương trình từ vùng nhớ này sang vùng nhớ khác trong quá trình xử lý thì thời điểm ánh xạ địa chỉ tương đối thành địa chỉ tuyệt đối bị trì hoãn đến tận thời điểm xử lý và cần phải sử dụng cơ chế phân cứng đặc biệt.

Cấu trúc một chương trình thể hiện cách quản lý bộ nhớ logic và cho ta thấy hình ảnh của chương trình ở bộ nhớ vật lý khi thực hiện. Mỗi chương trình có thể có các dạng cấu trúc sau: cấu trúc tuyến tính, cấu trúc động, cấu trúc Overlay, cấu trúc phân đoạn, cấu trúc phân trang.

Chú ý: mỗi chương trình có thể được chia thành nhiều modul. Các modul có thể giống hoặc khác nhau về cấu trúc.

1. Cấu trúc tuyến tính

Là cấu trúc mà sau khi biên dịch, các modul được tập hợp thành một chương trình hoàn thiện, chứa đầy đủ mọi thông tin để có thể thực hiện (trừ dữ liệu vào); Mọi biến ngoài đều được gán địa chỉ cụ thể. Khi thực hiện chỉ cần định vị chương trình một lần vào bộ nhớ.

M0	M1	M2	M3	M4	M5
----	----	----	----	----	----

Hình 4.2 - Cấu trúc tuyến tính

- **Ưu điểm:** đơn giản, dễ tổ chức biên dịch và định vị, thời gian thực hiện nhanh vì mọi công việc chuẩn bị đều được thực hiện trước, hệ thống không cần phải biên tập thêm, không mất thời gian tìm kiếm và nạp modul chương trình

(trừ dữ liệu vào). Người ta thường lấy thời gian thực hiện chương trình trong cấu trúc này làm chuẩn để đánh giá, so sánh các phương pháp tổ chức khác nhau. Ngoài ra, chương trình cấu trúc tuyến tính có tính lưu động cao, dễ dàng sao chép chương trình tới các hệ thống khác có cùng tập mã lệnh mà vẫn duy trì khả năng thực hiện.

- Nhược điểm: cấu trúc tuyến tính lãng phí bộ nhớ, mức lãng phí tỉ lệ với kích thước chương trình. Điều này gây khó khăn trong thực tế là chương trình càng lớn, người sử dụng càng phải tiết kiệm bộ nhớ.

2. Cấu trúc động

Trong cấu trúc động, các modul chương trình được biên tập một cách riêng biệt. Khi thực hiện chương trình, hệ thống chỉ cần định vị modul gốc. Trong quá trình thực hiện, cần tới modul nào (đã đăng ký với hệ thống để thực hiện) thì hệ thống cấp phát không gian nhớ và nạp tiếp modul đó. Khi hoạt động xong thì giải phóng modul khỏi bộ nhớ, thu hồi không gian nhớ.

M0		
M0	M1	M2
M0	M3	M4
M0	M5	

Hình 4.3 - Cấu trúc động

- Ưu điểm: nếu quản lý bộ nhớ và tổ chức tốt chương trình sẽ tiết kiệm bộ nhớ; kích thước bộ nhớ không phụ thuộc kích thước chương trình.

- Nhược điểm: trong cấu trúc này, trách nhiệm nạp và xoá các modul do người sử dụng đảm nhiệm, do đó các câu lệnh nạp, xoá phải được nêu ngay trong chương trình nguồn. Dẫn đến kích thước chương trình nguồn lớn và người sử dụng cần phải nắm vững cấu trúc chương trình và các công cụ điều khiển bộ nhớ của hệ điều hành. Ngoài ra, do phải xác định trình tự định vị modul ngay trong chương trình nguồn nên chương trình có cấu trúc động sẽ bị phụ thuộc vào version của hệ điều hành.

3. Cấu trúc Overlay

Trong cấu trúc Overlay, các modul chương trình sau khi biên dịch được chia

thành các mức:

Mức 0: mức chứa modul gốc dùng để nạp chương trình.

Mức 1: chứa các modul được gọi bởi mức 0.

Mức 2: chứa các modul được gọi bởi mức 1.

.....

Mức i: chứa các modul được gọi bởi mức i-1.

Bộ nhớ dành cho chương trình cũng được chia thành các mức tương ứng với các mức chương trình. Kích thước mỗi mức trong bộ nhớ bằng kích thước modul lớn nhất của mức chương trình tương ứng.

Mức 0: 80Kb	M0 (80Kb)	
Mức 1: 90Kb	M1 (50Kb)	M2 (90Kb)
Mức 2: 100Kb	M3 (50Kb)	M4 (100Kb)
		M5 (70 Kb)

Hình 4.4 - Cấu trúc Overlay

Chương trình nguồn được viết và biên dịch như các cấu trúc khác tạo thành các modul chương trình (các modul này không chứa các lệnh giao tiếp với hệ điều hành). Để tạo thành chương trình cấu trúc Overlay, người sử dụng cần cung cấp thông tin về các mức cho trình biên dịch thông qua sơ đồ Overlay (trong hệ điều hành DOS, các file này có phần mở rộng là OVL). Modul gốc được lưu trữ trong một file chương trình riêng.

Khi thực hiện chương trình, modul gốc được định vị vào bộ nhớ như chương trình có cấu trúc tuyến tính. Cần tới modul nào, hệ thống sẽ tìm kiếm trong sơ đồ Overlay và nạp vào bộ nhớ ở mức tương ứng.

Thành phần hệ thống duy trì hoạt động của chương trình Overlay là Supervisor Overlay. Khi nạp modul mới vào một mức bộ nhớ đã được sử dụng, các modul đang tồn tại trong đó sẽ bị xoá. Để có thể sử dụng lại các modul cũ sau này, hệ thống phải ghi tạm nó vào một file tạm thời ở bộ nhớ ngoài. Khi cần nạp một modul nào, trước hết Supervisor Overlay sẽ tìm kiếm trong file tạm

thời. Việc tạo file tạm thời là không cần thiết đối với các modul vào/ra nhiều lần hoặc các modul không thay đổi. Vì không thể dự báo trước khi cần thì gọi modul nào nên Supervisor Overlay phải nằm thường trú trong bộ nhớ. Điều này đương nhiên sẽ làm giảm hiệu quả của các biện pháp tối ưu bộ nhớ.

Trong cấu trúc Overlay, thông thường người ta không xét tới các lời gọi đồng mức (tức là một modul có lời gọi một modul khác cùng mức). Việc gọi đồng mức gây khó khăn vì modul gọi có thể bị phủ trước khi modul được gọi định vị xong trong bộ nhớ. Một sơ đồ Overlay hợp lý nói chung không thể xuất hiện các lời gọi đồng mức, để xử lý các lời gọi đồng mức, chương trình biên dịch phải tạo ra thêm một bảng thông tin điều khiển và dựa vào đó Supervisor Overlay mới biết được các modul gọi và được gọi.

- **Ưu điểm:** cấu trúc Overlay có tính chất định vị động, do đó cho phép sử dụng bộ nhớ nhiều hơn phần bộ nhớ mà hệ thống dành cho chương trình. Mặc dù quá trình định vị được tiến hành trong thời điểm thực hiện chương trình nhưng nhìn chung cấu trúc chương trình vẫn mang tính chất tĩnh, nó không thay đổi trong tất cả các lần thực hiện chương trình.

+ So với cấu trúc động, cấu trúc Overlay chỉ đòi hỏi người sử dụng cung cấp những thông tin đơn giản và quan trọng nhất là không gắn cố định cấu trúc vào chương trình nguồn.

+ Nếu người sử dụng xây dựng được sơ đồ Overlay tốt và các modul có độ dài không quá lớn thì hiệu quả của cấu trúc này không kém gì so với cấu trúc động. Chính vì vậy, các chương trình ứng dụng phần lớn đều có cấu trúc Overlay (ví dụ như FOXPRO, QUATRO...)

- **Nhược điểm:** cấu trúc Overlay vẫn yêu cầu người sử dụng cung cấp thông tin phụ (mặc dù chỉ là các thông tin đơn giản). Hiệu quả tiết kiệm bộ nhớ vẫn phụ thuộc cách tổ chức, bố trí các modul của chương trình. Chỉ cần một vài modul có kích thước lớn ở các mức Overlay khác nhau thì hiệu quả sử dụng bộ nhớ sẽ giảm hẳn.

+ Do cấu trúc chương trình là tĩnh nên nếu có bổ sung thêm bộ nhớ tự do thì hiệu quả sử dụng chương trình của hệ thống cũng không thay đổi.

4. Cấu trúc phân đoạn

Chương trình của người sử dụng được biên dịch thành từng modul độc lập. Thông tin về các modul được chứa trong một bảng điều khiển gọi là bảng quản lý đoạn (Segment Control Block - SCB). Trong bảng quản lý đoạn còn chứa các

thông tin trợ giúp việc định vị các modul vào bộ nhớ.

Khi thực hiện chương trình, hệ thống sẽ dựa vào bảng quản lý đoạn để nạp các modul cần thiết vào trong bộ nhớ cho tới khi hết khả năng. Nếu cần nạp các modul mới nhưng thiếu bộ nhớ thì hệ thống sẽ đưa bớt ra ngoài những modul có khả năng không sử dụng nữa.

- **Ưu điểm:** cấu trúc này không yêu cầu người sử dụng phải khai báo thêm thông tin, mọi công việc đều do hệ thống đảm nhận và khi dung lượng bộ nhớ tăng thì tốc độ thực hiện chương trình cũng tăng.

- **Nhược điểm:** hiệu quả sử dụng bộ nhớ phụ thuộc vào cách phân chia chương trình thành các modul độc lập. Cũng như cấu trúc Overlay, cấu trúc phân đoạn chỉ cần tồn tại một vài modul có độ dài lớn thì hiệu quả sử dụng bộ nhớ sẽ bị giảm một cách đáng kể. Mặt khác, chương trình có cấu trúc phân đoạn chỉ áp dụng được khi bộ nhớ quản lý theo kiểu phân đoạn.

5. Cấu trúc phân trang

Chương trình được biên dịch như cấu trúc tuyến tính, sau đó được phân chia thành các phần bằng nhau gọi là trang. Thông tin về các trang được chứa trong một bảng điều khiển gọi là bảng quản lý trang (Page Control Block- PCB). Mỗi phần tử trong bảng quản lý trang tương ứng với một trang trong chương trình của người sử dụng.

Khi thực hiện, hệ thống sẽ dựa vào bảng quản lý trang để nạp các trang cần thiết vào bộ nhớ.

- **Ưu điểm:** cấu trúc phân trang phát huy được hiệu quả sử dụng của bộ nhớ.
- **Nhược điểm:** chương trình chỉ áp dụng đối với bộ nhớ được quản lý theo kiểu phân trang.

III. CÁC SƠ ĐỒ QUẢN LÝ BỘ NHỚ

1. Sơ đồ phân hoạch cố định

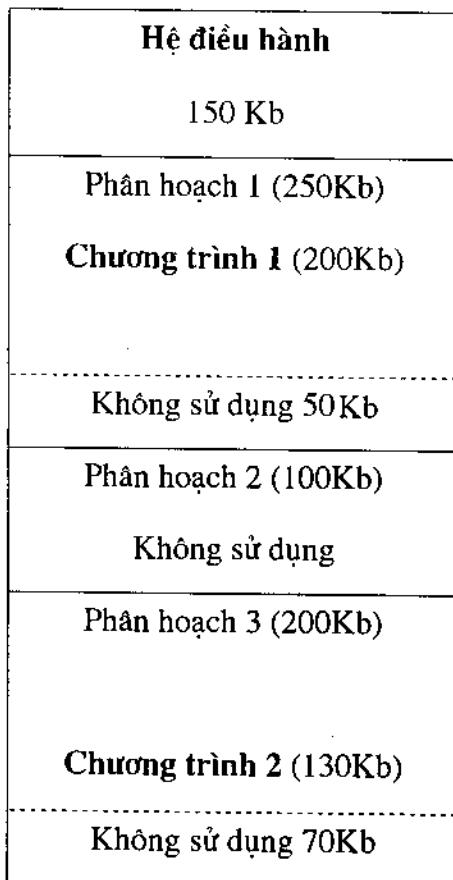
Bộ nhớ được chia thành n phần, không nhất thiết phải bằng nhau và mỗi phần được sử dụng như một bộ nhớ độc lập gọi là một phân hoạch. Mỗi phân hoạch có thể nạp được một chương trình và tổ chức thực hiện một cách đồng thời. Như vậy, trên lý thuyết nếu có n phân hoạch thì sẽ có thể nạp được n chương trình và thực hiện một cách đồng thời (n được gọi là hệ số song song của hệ thống).

Vì mỗi phân hoạch được coi như một bộ nhớ độc lập, nên mỗi chương trình sẽ có một danh sách quản lý không gian nhớ tự do riêng. Chương trình được nạp

vào phân hoạch nào thì sẽ ở đó cho đến khi kết thúc.

Mỗi phân hoạch sẽ được gắn với một số lớp phục vụ, chương trình khi định vị vào bộ nhớ cũng được phân lớp theo khai báo của người sử dụng. Mỗi phân hoạch chỉ phục vụ các chương trình thuộc lớp mình quản lý. Như vậy, chúng ta có thể tránh được trường hợp định vị chương trình nhỏ vào vùng nhớ lớn, tránh lãng phí bộ nhớ.

Để sửa đổi cấu trúc các phân hoạch cần phải nạp lại hệ điều hành nhưng để tránh mất thông tin chúng ta phải chờ cho tới khi các chương trình kết thúc. Cũng có một số công cụ cho phép kết hợp một số phân hoạch liền kề thành một phân hoạch có cấu trúc lớn hơn mà thông tin ở các phân hoạch khác vẫn được bảo toàn.



Hình 4.5 - Bộ nhớ phân hoạch cố định

Ưu điểm: sơ đồ phân hoạch cố định đơn giản, dễ tổ chức, giảm thời gian tìm kiếm.

- Nhược điểm:

+ Nếu kích thước chương trình nhỏ hơn kích thước phân hoạch chia nó thì phần bộ nhớ còn thừa sẽ không sử dụng được. Hiện tượng này được gọi là phân đoạn nội vi (internal fragmentation), nó dẫn tới trường hợp tổng bộ nhớ tự do còn lớn nhưng không sử dụng được, do đó không tận dụng được hết khả năng bộ nhớ, gây lãng phí bộ nhớ.

+ Chương trình sẽ không thực hiện được nếu kích thước của nó lớn hơn kích thước của phân hoạch lớn nhất. Khi đó, cần phải phân hoạch lại hoặc kết hợp các phân hoạch kề nhau thành một phân hoạch có kích thước lớn hơn.

Chú ý: trong sơ đồ phân hoạch cố định, chúng ta cần phải chọn được phân hoạch có kích thước phù hợp với kích thước chương trình để có thể đạt được hiệu quả tối ưu. Các thuật toán thường được áp dụng là:

- First Fit: chọn phân hoạch đầu tiên đủ lớn để cấp phát.
- Best Fit: chọn phân hoạch có kích thước nhỏ nhất nhưng đủ để cấp phát.
- Worst Fit: chọn phân hoạch có kích thước lớn nhất để cấp phát.

2. Sơ đồ phân hoạch động

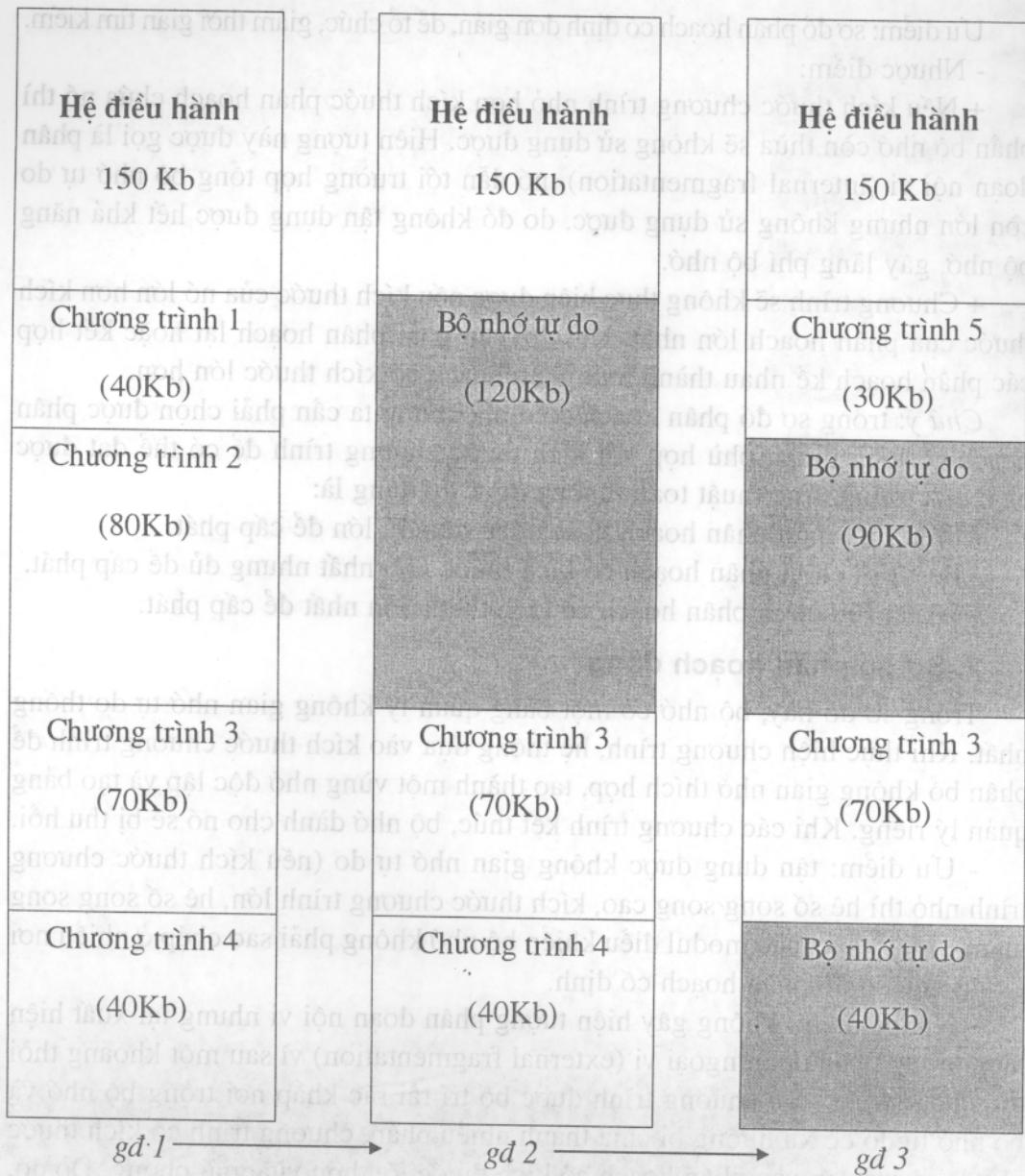
Trong sơ đồ này, bộ nhớ có một bảng quản lý không gian nhớ tự do thống nhất. Khi thực hiện chương trình, hệ thống dựa vào kích thước chương trình để phân bổ không gian nhớ thích hợp, tạo thành một vùng nhớ độc lập và tạo bảng quản lý riêng. Khi các chương trình kết thúc, bộ nhớ dành cho nó sẽ bị thu hồi.

- **Ưu điểm:** tận dụng được không gian nhớ tự do (nếu kích thước chương trình nhỏ thì hệ số song song cao, kích thước chương trình lớn, hệ số song song giảm). Mặt khác, các modul điều khiển bộ nhớ không phải sao chép ở nhiều nơi giống như sơ đồ phân hoạch cố định.

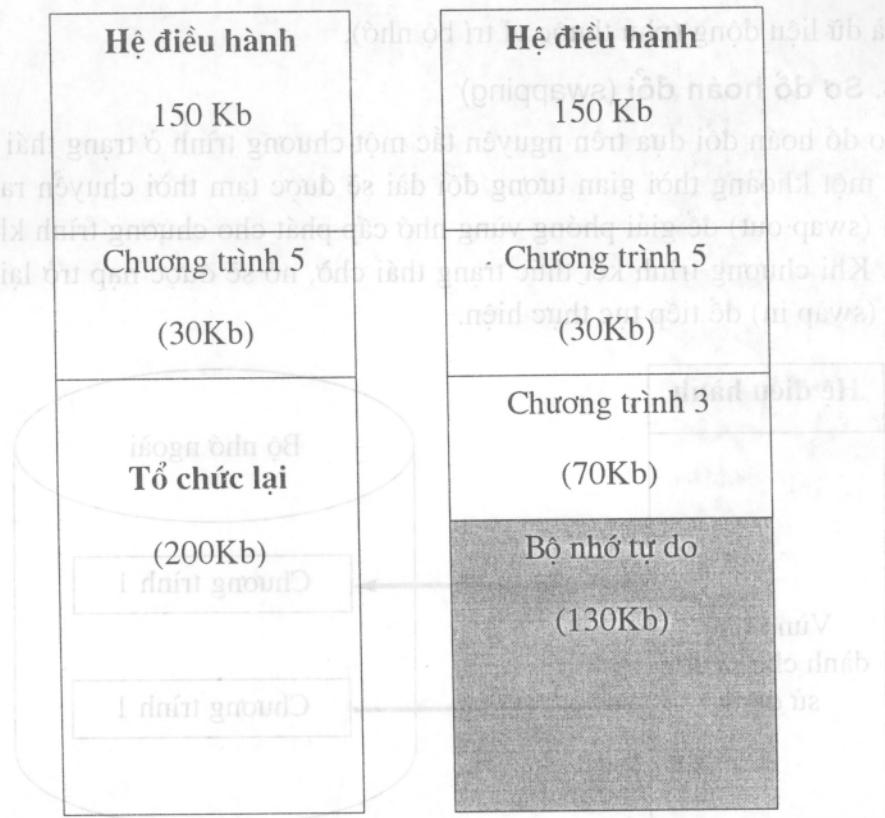
- **Nhược điểm:** không gây hiện tượng phân đoạn nội vi nhưng lại xuất hiện hiện tượng phân đoạn ngoại vi (external fragmentation) vì sau một khoảng thời gian hoạt động, các chương trình được bố trí rải rác khắp nơi trong bộ nhớ và bộ nhớ tự do có xu hướng bị chia thành nhiều phần, chương trình có kích thước nhỏ lại được nạp vào phân hoạch có kích thước lớn hơn vừa giải phóng. Do đó, hệ thống vẫn quản lý đầy đủ nhưng không sử dụng được. Để khắc phục hiện tượng này cần phải bố trí lại bộ nhớ.

+ Tìm thời điểm thích hợp để dừng các chương trình đang hoạt động.
+ Dưa một số hoặc toàn bộ các chương trình đang hoạt động cùng trạng thái của nó ra bộ nhớ ngoài, trả lại không gian nhớ cho hệ thống.

+ Tái định vị các chương trình và khôi phục trạng thái hoạt động.



Hình 4.6 - Bộ nhớ phân hoạch động



Hình 4.7 - Tổ chức lại bộ nhớ

Chú ý:

Thời điểm thích hợp để dừng chương trình là thời điểm mà các giá trị đã được tính toán xong và lưu vào bộ nhớ hoặc thanh ghi. Chỉ có thời điểm này, chúng ta mới có thể thực hiện công việc lưu và phục hồi trạng thái chương trình được.

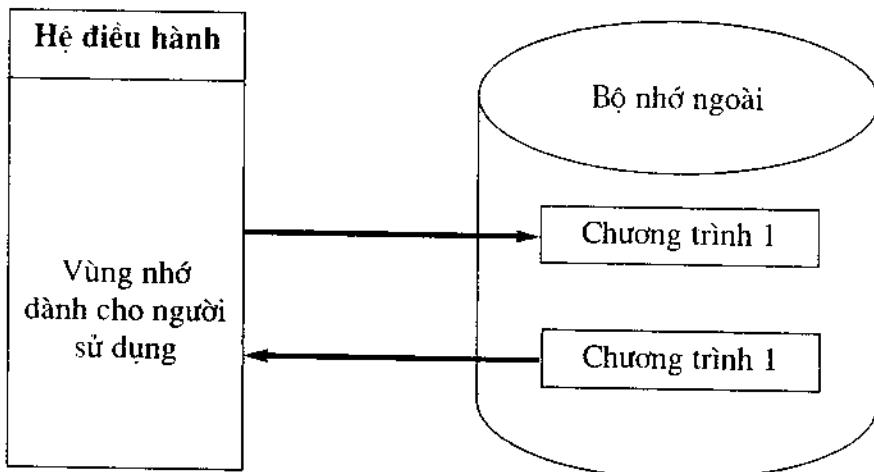
Do việc đưa chương trình ra, định vị lại và phục hồi trạng thái phức tạp nên có thể không cần đưa hết tất cả các chương trình ra bộ nhớ ngoài. Khi một chương trình được đưa ra, bộ nhớ cấp phát cho nó được giải phóng và có khả năng mức độ phân đoạn bộ nhớ giảm. Quá trình này sẽ dừng khi hệ số phân đoạn bộ nhớ đạt mức an toàn.

Việc tái định vị chương trình phức tạp hơn nhiều so với định vị ban đầu vì trong quá trình thực hiện, chương trình có thể tạo ra các câu lệnh mới và dữ liệu. Hệ thống cần phải phân biệt đâu là dữ liệu tĩnh (không phụ thuộc vị trí bộ nhớ),

đó là dữ liệu động (phụ thuộc vị trí bộ nhớ).

3. Sơ đồ hoán đổi (swapping)

Sơ đồ hoán đổi dựa trên nguyên tắc một chương trình ở trạng thái chờ đợi trong một khoảng thời gian tương đối dài sẽ được tạm thời chuyển ra bộ nhớ ngoài (swap out) để giải phóng vùng nhớ cấp phát cho chương trình khác hoạt động. Khi chương trình kết thúc trạng thái chờ, nó sẽ được nạp trở lại bộ nhớ trong (swap in) để tiếp tục thực hiện.



Hình 4.8 - Sơ đồ swapping

Chú ý: khi chương trình được nạp trở lại bộ nhớ trong có thể xảy ra hai trường hợp: Nếu quá trình ánh xạ địa chỉ logic thành địa chỉ vật lý được thực hiện vào thời điểm nạp chương trình thì chương trình cần được cấp phát lại đúng vùng nhớ mà nó đã chiếm giữ trước đó. Nếu quá trình ánh xạ được thực hiện vào thời điểm xử lý thì có thể nạp lại chương trình vào một vùng nhớ bất kỳ.

Sơ đồ swapping cần sử dụng một bộ nhớ ngoài (thường là đĩa từ), bộ nhớ này phải đủ lớn để lưu trữ các chương trình bị hoán đổi và phải cho phép hệ thống truy nhập trực tiếp đến các chương trình này.

Trong các hệ swapping, cần phải quan tâm tới thời gian chuyển đổi giữa các tác vụ. Mỗi chương trình cần được cấp phát giờ CPU đủ lớn để không thấy rõ sự chậm trễ do các thao tác swap gây ra. Nếu không, hệ thống sẽ dùng phần lớn thời gian để chuyển đổi các chương trình vào/ra bộ nhớ trong, như vậy giờ CPU được sử dụng không hiệu quả.

Trong sơ đồ tổ chức này cũng gây hiện tượng phân đoạn bộ nhớ, do đó phải sử dụng các kỹ thuật dồn bộ nhớ để loại bỏ hiện tượng phân mảnh ngoại vi.

4. Sơ đồ phân đoạn

Các sơ đồ phân hoạch cố định và phân hoạch động không áp dụng được khi kích thước chương trình lớn hơn kích thước bộ nhớ vật lý. Ngoài ra, hệ số tích cực của các byte trong bộ nhớ cũng không được đồng đều (các byte được gọi là tích cực nếu nội dung của nó được sử dụng để thực hiện các câu lệnh trong thời điểm quan sát). Sơ đồ quản lý bộ nhớ tốt là sơ đồ có số byte tích cực lớn.

Vì người sử dụng không cần quan tâm tới chương trình của họ được bố trí trong bộ nhớ như thế nào (liên tục hay không liên tục) nên trong sơ đồ phân đoạn, các modul chương trình được biên dịch một cách riêng biệt. Thông tin về các modul chương trình được chứa trong bảng quản lý đoạn - SCB (Segment Control Block). Mỗi phần tử trong SCB tương ứng với một modul của chương trình và được đặc trưng bởi 3 trường tin:

- Dấu hiệu D: cho biết modul đã được nạp vào bộ nhớ hay chưa ($D = 0$ nếu modul chưa được nạp, $D = 1$ ngược lại).
- Địa chỉ A: địa chỉ của vùng nhớ sẽ định vị modul.
- Độ dài L: cho biết kích thước của modul.

Ban đầu, chỉ có trường L và D có giá trị. SCB được xây dựng ngay từ khi biên dịch chương trình.

Khi thực hiện, SCB được nạp vào bộ nhớ, địa chỉ đầu được đưa vào thanh ghi đoạn Rs.

Địa chỉ truy nhập dữ liệu được biểu diễn bởi cặp (s,d) trong đó s là số hiệu modul cần truy nhập, d là địa chỉ tương đối tính từ đầu segment.

Để truy nhập tới một dữ liệu cần phải qua hai bước:

- Bước 1: hệ thống lấy nội dung thanh ghi Rs cộng với s để tìm được phần tử thứ s trong SCB. Nếu trường dấu hiệu $D = 0$ (modul chưa được nạp vào bộ nhớ) thì hệ thống làm thủ tục nạp modul vào bộ nhớ, xin cấp phát không gian nhớ theo kích thước L, tìm modul ở bộ nhớ ngoài và định vị vào vùng nhớ được cấp phát, sửa lại nội dung trường địa chỉ A để nó chỉ tới modul thứ s. Nếu không tìm được modul, hệ thống sẽ báo lỗi và ngừng thực hiện. Nếu trường dấu hiệu $D = 1$ (modul đã nạp vào bộ nhớ), hệ thống sẽ thực hiện bước tiếp theo.

- Bước 2: hệ thống lấy nội dung trường địa chỉ A cộng với d và truy nhập tới bộ nhớ theo địa chỉ vừa tính được để đọc/ghi dữ liệu.

Ví dụ: Giả sử modul đầu tiên của chương trình ($s = 1$) có địa chỉ tương đối $d = 03026$, độ dài modul $L = 5000$, địa chỉ đầu $A = 400$, modul đã được nạp vào bộ nhớ ($D = 1$), nội dung thanh ghi Rs là 3. Để truy nhập tới modul, hệ thống cộng 3 với 1 để tìm ra phần tử thứ 4 trong SCB; Lấy $400 + 03026$ để tìm được địa chỉ 03426 truy nhập dữ liệu.

- **Ưu điểm:** sơ đồ này không đòi hỏi công cụ tổ chức đặc biệt, do đó có thể áp dụng trên mọi hệ thống.

- **Nhược điểm:** hiệu quả sử dụng bộ nhớ phụ thuộc vào cấu trúc chương trình của người sử dụng. Ngoài ra, sau một thời gian hoạt động, bộ nhớ bị phân đoạn, do đó cần phải tổ chức lại bộ nhớ bằng cách đưa bớt một số modul ra ngoài. Việc bố trí lại sẽ đơn giản hơn các sơ đồ trên vì có sự hỗ trợ của SCB. Phần tử bị đưa ra thì trường D sẽ bị gán giá trị bằng 0 và bộ nhớ cấp phát cho modul (xác định bằng trường A và L) sẽ được trả lại cho hệ thống.

Nếu xuất hiện nhu cầu cần phải tổ chức lại bộ nhớ thì vấn đề đầu tiên cần giải quyết là chọn modul nào để đưa ra ngoài. Thông thường, hệ thống hay áp dụng các giải pháp sau:

- Đưa modul tồn tại lâu nhất trong bộ nhớ.
- Đưa modul có lần sử dụng cuối cùng cách thời điểm hiện tại lâu nhất.
- Đưa modul có tần suất sử dụng thấp nhất.

Mỗi giải pháp đều có ưu điểm và nhược điểm riêng, phụ thuộc vào từng tình huống cụ thể. Vì vậy, người ta thường đưa ra dưới dạng tuỳ chọn.

Sơ đồ phân đoạn chỉ được áp dụng đồng bộ với chương trình có cấu trúc phân đoạn.

M0	M1	M2	M3	M4
----	----	----	----	----

Hình 4.9 - Các modul (segment) chương trình

	M0	M2		M1		M3		M4
--	----	----	--	----	--	----	--	----

Hình 4.10 - Ánh xạ chương trình sang bộ nhớ vật lý

5. Sơ đồ phân trang

Sơ đồ phân trang là một trường hợp đặc biệt của sơ đồ phân đoạn. Trong sơ đồ phân trang, bộ nhớ chương trình và bộ nhớ vật lý được chia thành các phần bằng nhau gọi là trang. Ở bộ nhớ vật lý, các trang được đánh số thứ tự từ 0, 1, 2, ... gọi là địa chỉ trang, như vậy trang là đơn vị dùng để phân phối bộ nhớ. Số trang của bộ nhớ vật lý phụ thuộc vào kích thước trang và kích thước bộ nhớ (thường là xác định) còn số trang của bộ nhớ chương trình phụ thuộc vào kích thước chương trình (có thể tùy ý). Thông thường, để tạo điều kiện thuận lợi trong xử lý, người ta thường đặt kích thước trang là lũy thừa của 2.

Page 0	Page 1	Page 2	Page 3	Page 4
--------	--------	--------	--------	--------

Hình 4.11 - Các trang chương trình

Page 0	Page 1	Page 2	Page 3	Page 4
--------	--------	--------	--------	--------

Hình 4.12 - Bộ nhớ vật lý và ánh xạ các trang chương trình

Mỗi trang được biểu diễn bởi một cặp (p, d) trong đó: p là số hiệu trang và d là địa chỉ tương đối tính từ đầu trang.

Khi thực hiện chương trình, hệ thống xây dựng một bảng quản lý trang (Page Control Block - PCB) để xác lập mối quan hệ giữa trang vật lý và trang logic. Mỗi phần tử trong bảng quản lý trang tương ứng với một trang logic và được đặc trưng bởi hai trường tin:

- Dấu hiệu D: cho biết trang đã được nạp vào bộ nhớ hay chưa ($D = 1$ đã nạp; $D = 0$ chưa nạp).
- Địa chỉ Ap: là địa chỉ trang vật lý chứa trang logic p đang xét. Nếu $D = 0$ thì Ap có thể chứa thông tin cần thiết để tìm trang ở bộ nhớ ngoài.

Địa chỉ của bảng quản lý trang được chứa trong thanh ghi quản lý trang Rp. Để truy nhập tới dữ liệu cần qua hai bước:

- Bước 1: Hệ thống lấy nội dung Rp cộng với p để truy nhập tới phần tử thứ p trong bảng quản lý trang (tương ứng với trang p). Nếu $D = 0$ (trang chưa được nạp vào bộ nhớ) thì hệ thống sẽ nạp trang vào bộ nhớ. Khi đó $D = 1$ và trường

địa chỉ Ap sẽ chứa địa chỉ trang trong bộ nhớ vật lý.

- Bước 2: Hệ thống lấy địa chỉ trang Ap ghép với d tạo ra địa chỉ vật lý của dữ liệu đã đưa vào đó và truy nhập tới địa chỉ vừa tính được để đọc/ ghi dữ liệu.

Ưu điểm: sơ đồ phân trang làm tăng tốc độ truy nhập bộ nhớ (so với sơ đồ phân đoạn thì ở bước 2, phép cộng được thay thế bởi phép ghép). Mặt khác, sơ đồ không bị hiện tượng phân đoạn bộ nhớ (vì kích thước trang logic và vật lý bằng nhau). Nếu còn bộ nhớ tự do thì kích thước phải chẵn trang, do đó bao giờ cũng đủ chỗ để đưa các trang mới vào. Tình trạng thiếu bộ nhớ chỉ thực sự xảy ra khi tất cả các trang vật lý đã được sử dụng hết.

Nhược điểm: sơ đồ phân trang cần có thiết bị vật lý hỗ trợ công việc định vị trang vì mỗi trang chương trình không phải là một modul hoàn chỉnh nên không thể biến đổi địa chỉ và tự định vị theo địa chỉ đầu. Do đó, việc định vị trang phải được sự hỗ trợ của các công cụ ngoài.

Chú ý:

Sơ đồ phân trang đảm bảo hệ số song song cao cho hệ thống, kích thước chương trình và dữ liệu nói chung không hạn chế nhưng khi bộ nhớ logic quá lớn thì kích thước bảng quản lý trang cũng tăng theo và hệ thống sẽ phải tốn nhiều không gian nhớ vật lý để lưu trữ nó. Trên thực tế, cũng như chương trình, chỉ một phần bảng quản lý trang là có tác dụng tích cực trong quá trình thực hiện, do đó để nâng cao hiệu quả của sơ đồ phân trang, cần phải cải tiến bảng quản lý trang.

Một vấn đề khác cũng cần lưu ý là phải chọn kích thước trang sao cho phù hợp vì hiệu quả của sơ đồ phụ thuộc nhiều vào kích thước trang. Nếu kích thước trang quá nhỏ thì kích thước bảng quản lý trang sẽ lớn và khả năng phai thường xuyên nạp lại trang cao. Ngược lại, nếu kích thước trang lớn thì số trang được nạp để xử lý sẽ giảm, gây tốn động đáng kể đến hiệu quả sử dụng bộ nhớ. Vì vậy, kích thước trang thường được chọn giao động từ 28 đến 210. Trong các hệ thống máy tính của IBM, kích thước các trang theo truyền thống là 512 bytes nhưng trong các hệ thống hiện nay, dung lượng bộ nhớ trong tương đối lớn thì kích thước trang có thể tăng lên đến 4Kb.

Qua cách tổ chức và thực hiện trên, ngoài bảng quản lý trang thì hệ thống còn cần bộ nhớ để chứa các trang dữ liệu và chương trình. Số trang dành cho chương trình càng nhiều thì hệ số song song của hệ thống càng giảm, ngược lại số trang dành cho chương trình ít thì hệ số song song tăng nhưng tốc độ thực

hiện chương trình giảm vì chương trình luôn rơi vào tình trạng thiếu trang vật lý để thực hiện. Từ đó, hiệu quả chung của phương pháp này phụ thuộc vào cách nạp trang và thay thế trang tích cực.

* *Các giải pháp nạp trang:*

- Biện pháp đơn giản nhất là nạp tất cả các trang của chương trình vào bộ nhớ ngay từ đầu, như vậy không thể xảy ra tình trạng thiếu trang tích cực khi đang thực hiện. Trong giải pháp này, bộ nhớ ngoài không được sử dụng để mở rộng bộ nhớ trong. Ưu điểm của giải pháp là đơn giản nhưng không phát huy được hiệu quả điểm đặc thù của sơ đồ phân trang.

- Một giải pháp khác cũng cho phép giảm khả năng thiếu trang tích cực mà không cần dùng tới không gian bộ nhớ vật lý quá lớn là giải pháp nạp trước (nạp trước các trang sắp sử dụng). Giải pháp cho phép người sử dụng tạo chương trình ở bộ nhớ logic với kích thước tùy ý, đồng thời hệ thống có thể duy trì hệ số song song cần thiết cho hoạt động có hiệu quả của toàn hệ thống. Điểm mấu chốt của giải pháp nạp trang trước là phải dự báo được các trang tích cực chuẩn bị sử dụng trong quá trình thực hiện. Nếu xác định được, ta có thể nạp trước các trang này vào bộ nhớ nhưng việc dự báo trang nào sắp sử dụng là cực kỳ khó khăn. Do đó, chi phí nạp trước các trang không dùng đến có thể sẽ lớn hơn rất nhiều chi phí nạp các trang thực sự được sử dụng.

- Thông thường, các hệ thống thường áp dụng giải pháp nạp trang theo yêu cầu. Theo giải pháp này thì trang chỉ được nạp khi xuất hiện yêu cầu truy nhập dữ liệu của trang, như vậy mọi lần nạp trang đều thực sự là cần thiết. Giải pháp này đảm bảo hiệu quả cao khi chúng ta có cách phân bố các trang ở bộ nhớ ngoài hợp lý và có cơ chế tìm kiếm tốt.

* *Các giải pháp thay thế trang:*

- Nếu còn nhiều không gian nhớ tự do thì không cần thiết phải thay thế trang nhưng khi thiếu không gian nhớ thì cần đưa một số trang ra ngoài và nạp vào các trang khác cần thiết cho việc thực hiện chương trình. Nguyên tắc chung là phải thay thế các trang có lần sử dụng kế tiếp, cách thời điểm đổi trang càng xa càng tốt. Trong trường hợp lý tưởng là các trang đó không còn cần sử dụng nữa nhưng trên thực tế, không thể dự đoán trước được các diễn biến của chương trình. Do đó, tồn tại một số giải pháp đổi trang cụ thể như sau:

- Giải pháp đổi vòng tròn hoặc đổi ngẫu nhiên, tổ chức đơn giản nhưng có thể dẫn đến khả năng đổi các trang vẫn còn đang sử dụng.

- Giải pháp FIFO (First In First Out) - trang nào nạp trước sẽ bị thay thế trước, như vậy trang có thời gian tồn tại trong bộ nhớ lâu nhất sẽ bị thay thế. Nếu chương trình được xây dựng theo cấu trúc tuyến tính thì giải pháp này rất tốt còn với các chương trình xây dựng theo nguyên tắc cấu trúc thì có thể xảy ra trường hợp đưa một trang quan trọng, đang sử dụng ra ngoài. Tuy vậy, giải pháp này cũng tương đối đơn giản vì chỉ cần theo dõi quá trình nạp trang để xác định thứ tự ưu tiên của các trang cần thay thế.

- Giải pháp LRU (Last Recently Used) - thay thế trang có lần sử dụng cuối, cách thời điểm đổi trang lâu nhất. Giải pháp này dựa trên giả thiết chương trình có tính cục bộ hoá trang sử dụng, không có các lệnh chuyển điều khiển đi xa câu lệnh đang thực hiện, dẫn đến giải pháp không đảm bảo tối ưu trong mọi trường hợp, đồng thời chi phí thực hiện cao, vì để xác định được cần thay thế trang nào, hệ thống cần phải định kỳ kiểm tra, thống kê các trang vào đã được sử dụng để tìm ra trang cần thay thế.

- Giải pháp LFU (Last Frequently Used) - thay thế trang có tần suất sử dụng thấp nhất. Đối với giải pháp này cần phải thống kê số lần truy nhập trang để tính được tần suất truy nhập.

Các giải pháp thay thế trang nói trên có thể áp dụng chung cho toàn bộ hệ thống hoặc áp dụng cục bộ cho từng chương trình. Trong trường hợp áp dụng cục bộ cho một chương trình đang thực hiện, hệ thống cần phải lưu trữ kích thước bộ nhớ đã cấp phát cho chương trình và tránh trường hợp cấp phát bộ nhớ không đều giữa các chương trình.

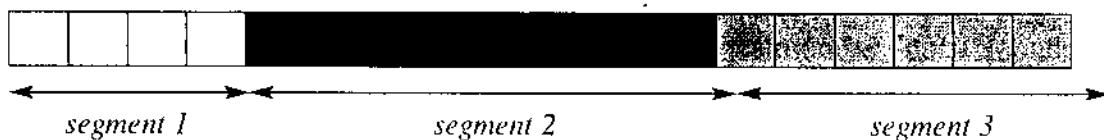
6. Sơ đồ kết hợp phân trang và phân đoạn

Sơ đồ phân trang đảm bảo hiệu quả sử dụng bộ nhớ không phụ thuộc vào cấu trúc chương trình của người sử dụng, điều khiển trang thuận tiện, đơn giản. Tuy nhiên, khi chương trình có kích thước lớn thì kích thước bảng quản lý trang cũng lớn theo, dẫn đến lãng phí bộ nhớ. Mặt khác, nếu kích thước trang quá nhỏ thì kích thước bảng quản lý trang sẽ lớn và khả năng phải thường xuyên nạp lại trang cao. Ngược lại, nếu kích thước trang lớn thì số trang được nạp để xử lý sẽ giảm gây tác động đáng kể đến hiệu quả sử dụng bộ nhớ.

Sơ đồ phân đoạn linh hoạt hơn về độ dài của các đoạn nhưng cũng chính vì độ dài của các đoạn khác nhau nên phức tạp trong thực hiện và cấp phát bộ nhớ.

Để phát huy được các ưu điểm và hạn chế nhược điểm của các sơ đồ trên, người ta thường sử dụng sơ đồ kết hợp phân trang và phân đoạn.

Trong sơ đồ này, chương trình được biên dịch theo sơ đồ phân đoạn và có một bảng quản lý đoạn chung (SCB). Mỗi đoạn trong chương trình lại được biên tập theo sơ đồ phân trang và tạo ra từng bảng quản lý trang (PCB) riêng cho mỗi đoạn. Khi chương trình được nạp vào hệ thống, hệ điều hành sẽ cấp phát cho chương trình các trang cần thiết để chứa đủ các đoạn của chương trình.



Hình 4.13 - Sơ đồ kết hợp phân trang - phân đoạn

Để hỗ trợ kỹ thuật phân đoạn cần có một bảng quản lý đoạn chung cho toàn bộ chương trình nhưng trong sơ đồ kết hợp này, mỗi đoạn cần có một bảng phân trang riêng biệt. Như vậy, trường địa chỉ A của phần tử thứ i trong bảng quản lý đoạn - SCB sẽ là nơi chứa bảng quản lý trang thứ i - PCB_i, trường độ dài L chứa độ dài của PCB_i.

Khi thực hiện, bảng quản lý trang sẽ được nạp vào bộ nhớ và địa chỉ đầu của nó được đưa vào thanh ghi quản lý đoạn Rs. Địa chỉ truy nhập dữ liệu được biểu diễn bởi một bộ ba phần tử (s,p,d) trong đó:

s - số hiệu đoạn cần truy nhập trong bảng quản lý đoạn.

p - số hiệu trang cần truy nhập trong bảng quản lý trang.

d - địa chỉ tương đối tính từ đầu trang.

Để truy nhập tới dữ liệu, hệ thống cần thực hiện ba bước:

- Bước 1: Lấy nội dung thanh ghi Rs cộng với s và truy nhập tới phần tử thứ s trong bảng quản lý đoạn.

- Bước 2: Nếu D = 0 thì thực hiện thủ tục nạp PCB tương ứng vào bộ nhớ và cập nhật nội dung trường A. Khi nạp xong PCB, hệ thống cộng nội dung trường A với p để truy nhập tới phần tử thứ p trong PCB.

- Bước 3: Khi tìm được phần tử thứ p trong PCB, hệ thống sẽ ghép nội dung của Ap (tương ứng phần tử thứ p) với d để tìm ra địa chỉ đọc/ghi dữ liệu.

Chú ý: ở sơ đồ này, bộ nhớ thường được chia thành ba phần chứa SCB, PCB và các trang. Bản thân bộ nhớ dành cho SCB và PCB cũng được quản lý theo chế độ phân trang.

IV. BỘ NHỚ ẢO

1. Khái niệm bộ nhớ ảo (Virtual Memory)

Nếu đặt toàn bộ chương trình vào bộ nhớ vật lý thì kích thước chương trình sẽ bị hạn chế bởi kích thước bộ nhớ vật lý.

Trên thực tế, trong nhiều trường hợp chúng ta không cần phải nạp toàn bộ chương trình vào bộ nhớ vật lý cùng một lúc vì tại mỗi thời điểm, chỉ có một lệnh của chương trình được thực hiện. Ví dụ các chương trình đều có một đoạn mã xử lý lỗi nhưng đoạn mã này rất ít khi được sử dụng vì hiếm khi lỗi xảy ra, trong trường hợp này không cần thiết phải nạp đoạn mã xử lý lỗi ngay từ đầu.

Từ nhận xét trên, một giải pháp được đề xuất là cho phép chương trình thực hiện được nạp từng phần vào bộ nhớ vật lý. Nguyên tắc của giải pháp này là tại mỗi thời điểm, trong bộ nhớ vật lý chỉ lưu trữ các lệnh và dữ liệu phục vụ cho hoạt động của chương trình tại thời điểm đó. Khi cần tới các lệnh hoặc dữ liệu mới, hệ thống sẽ nạp chúng vào bộ nhớ tại vị trí trước đó bị chiếm giữ bởi các lệnh không còn cần tới nữa. Với giải pháp này, một chương trình có thể có kích thước lớn hơn kích thước vùng nhớ cấp phát cho nó.

Một cách để thực hiện ý tưởng của giải pháp trên là sử dụng kỹ thuật overlay. Kỹ thuật overlay không đòi hỏi bất kỳ sự trợ giúp đặc biệt nào của hệ điều hành nhưng trái lại lập trình viên phải biết cách lập trình theo cấu trúc overlay và điều này đòi hỏi khá nhiều công sức.

Để giải phóng lập trình viên khỏi các rủi ro giới hạn về bộ nhớ, đồng thời không làm tăng thêm mức độ khó khăn trong công việc lập trình, người ta nghĩ tới hướng phát triển các kỹ thuật tự động, cho phép xử lý một chương trình có kích thước lớn trong một vùng nhớ nhỏ. Một kỹ thuật được áp dụng khá phổ biến đó là kỹ thuật bộ nhớ ảo.

Bộ nhớ ảo (virtual memory) là một kỹ thuật cho phép xử lý một chương trình không được nạp toàn bộ vào bộ nhớ vật lý. Bộ nhớ ảo mô hình hóa bộ nhớ như một bảng lưu trữ rất lớn và đồng nhất, tách biệt hẳn khái niệm bộ nhớ logic và bộ nhớ vật lý. Người sử dụng chỉ nhìn thấy và làm việc trong không gian địa chỉ ảo, việc chuyển đổi sang bộ nhớ vật lý do hệ điều hành thực hiện với sự trợ giúp của các cơ chế phân cứng cụ thể.

Chú ý: cần kết hợp kỹ thuật swapping để chuyển các phần của chương trình vào/ra giữa bộ nhớ chính và bộ nhớ phụ khi cần thiết.

Nhờ có sự tách biệt giữa bộ nhớ ảo và bộ nhớ vật lý nên có thể tổ chức một bộ nhớ ảo có kích thước lớn hơn rất nhiều lần bộ nhớ vật lý.

Bộ nhớ ảo cho phép giảm nhẹ công việc của lập trình viên vì họ không cần quan tâm đến giới hạn của bộ nhớ vật lý cũng như không cần tổ chức chương trình theo cấu trúc Overlay.

2. Cài đặt bộ nhớ ảo

Bộ nhớ ảo có thể được cài đặt dựa vào hai kỹ thuật: phân trang theo yêu cầu (demand paging) hoặc phân đoạn theo yêu cầu (demand segmentation). Tuy nhiên, việc cấp phát và thay thế các đoạn phức tạp hơn các trang vì các đoạn có kích thước không bằng nhau, do đó kỹ thuật phân trang theo yêu cầu được áp dụng phổ biến hơn.

Một hệ thống phân trang theo yêu cầu là hệ thống sử dụng kỹ thuật phân trang kết hợp với kỹ thuật swapping. Một chương trình được xem như một tập hợp các trang thường trú trên bộ nhớ ngoài (thường là đĩa từ). Khi cần xử lý, chương trình sẽ được nạp vào bộ nhớ trong nhưng thay vì nạp toàn bộ chương trình, hệ thống chỉ nạp các trang cần thiết trong thời điểm hiện tại. Như vậy, một trang chỉ được nạp vào bộ nhớ trong khi có yêu cầu. Với mô hình này, cần phải có một cơ chế phân cứng phân biệt các trang đang ở bộ nhớ trong và các trang ở bộ nhớ ngoài.

Cơ chế phân cứng hỗ trợ kỹ thuật phân trang theo yêu cầu là sự kết hợp giữa cơ chế hỗ trợ phân trang và kỹ thuật swapping. Cơ chế bao gồm:

- **Bảng trang:** cấu trúc bảng trang phải phản ánh được tình trạng của một trang đang ở bộ nhớ trong hay bộ nhớ ngoài.
- **Bộ nhớ ngoài:** bộ nhớ lưu trữ các trang không được nạp vào bộ nhớ trong. Bộ nhớ ngoài thường là đĩa từ và phần không gian dùng để lưu trữ tạm các trang, trong kỹ thuật swapping được gọi là không gian swapping.

3. Hiện tượng lỗi trang

Khi hệ thống truy xuất tới một trang nhưng trang này chưa được nạp vào bộ

nhớ trong sẽ làm phát sinh một lỗi trang (page fault). Hệ điều hành sẽ xử lý lỗi trang theo các bước sau:

Step 1: Kiểm tra việc truy xuất tới bộ nhớ có hợp lệ hay không.

Nếu có, goto *Step 2*

Ngược lại, kết thúc chương trình.

Step 2: Tìm vị trí chưa trang cần truy xuất trên đĩa từ

Step 3: Tìm một trang vật lý trống trong bộ nhớ chính

Nếu tìm thấy, goto *Step 4*

Nếu không, chọn một trang đang sử dụng và chuyển nội dung trang này ra bộ nhớ ngoài (lưu nội dung trang này vào đĩa từ), cập nhật bảng quản lý trang tương ứng.

Step 4: Chuyển trang muốn truy xuất từ bộ nhớ ngoài vào bộ nhớ trong: nạp trang cần truy xuất vào trang vật lý trống, cập nhật nội dung bảng quản lý trang.

Step 5: Tái kích hoạt chương trình.

4. Thay thế trang

Khi xảy ra lỗi trang, hệ thống cần phải nạp trang thiếu vào bộ nhớ. Nếu không còn trang trống trong bộ nhớ trong, hệ thống cần thực hiện việc thay thế trang tức là chọn một trang đang tồn tại ở bộ nhớ trong (không được sử dụng tại thời điểm hiện tại) và đưa nó ra không gian swapping trên đĩa từ để giải phóng một trang vật lý, dành chỗ nạp trang cần truy xuất vào bộ nhớ.

Các thuật toán thay thế trang được áp dụng như trong sơ đồ quản lý bộ nhớ theo kiểu phân trang, bao gồm:

- Thay thế trang có thời gian tồn tại trong bộ nhớ lâu nhất (FIFO).
- Thay thế trang có lần sử dụng cuối cùng, cách thời điểm hiện tại lâu nhất (LRU).
- Thay thế trang có tần suất sử dụng thấp nhất (LFU).
- Thay thế trang có tần suất sử dụng nhiều nhất (MFU - Most Frequently Used).

Câu hỏi và bài tập

1. Giải thích sự khác biệt giữa địa chỉ vật lý và địa chỉ logic.
2. Trình bày các cấu trúc cơ bản của chương trình.
3. Nêu mục đích của quản lý bộ nhớ và trình bày nguyên tắc hoạt động của các sơ đồ quản lý bộ nhớ.
4. Phân biệt giữa phân mảnh nội vi và phân mảnh ngoại vi.
5. Giả sử bộ nhớ chính được chia thành các phân hoạch có kích thước 600Kb, 500Kb, 200Kb, 300Kb (theo thứ tự), cho biết các tiến trình có kích thước 212Kb, 417Kb, 112Kb và 426Kb (theo thứ tự) sẽ được cấp phát bộ nhớ như thế nào nếu sử dụng các thuật toán: First - Fit, Best - Fit, Worst Fit. Thuật toán nào cho phép sử dụng bộ nhớ hiệu quả nhất trong trường hợp trên.
6. Trình bày khái niệm và mục đích của bộ nhớ ảo.
7. Thế nào là hiện tượng lỗi trang? Trình bày phương pháp giải quyết của hệ điều hành khi gặp hiện tượng lỗi trang.

Chương 5

QUẢN LÝ BỘ NHỚ NGOÀI

Mục tiêu

Sau chương này, người học có thể hiểu rõ các biện pháp quản lý và cấp phát không gian nhớ tự do trên đĩa từ, các thuật toán lập lịch cho đĩa từ và nguyên tắc quản lý thông tin trên bộ nhớ ngoài - Hệ File.

Nội dung

Trình bày các yêu cầu của quản lý bộ nhớ ngoài, cấu trúc vật lý của đĩa từ, các phương pháp quản lý và cấp phát không gian nhớ tự do, các thuật toán lập lịch cho đĩa từ và một số vấn đề về hệ file.

I. CÁC KHÁI NIỆM CƠ BẢN

1. Yêu cầu của quản lý bộ nhớ ngoài

Khi cần lưu trữ các chương trình hoặc dữ liệu, các hệ thống máy tính bắt buộc phải sử dụng bộ nhớ ngoài (đĩa từ, băng từ, compaq...). Nhiệm vụ chính của hệ điều hành phải đảm bảo được các chức năng sau:

- Quản lý không gian nhớ tự do trên bộ nhớ ngoài (Free Space Manage).
- Cấp phát không gian nhớ tự do (Allocation Methods).
- Cung cấp các khả năng định vị bộ nhớ ngoài.
- Lập lịch cho bộ nhớ ngoài (Disk Scheduling).

2. Cấu trúc vật lý

Xét cấu trúc vật lý của đĩa từ: đĩa từ bao gồm một hoặc nhiều lá đĩa đặt đồng trục. Mỗi mặt đĩa chia thành các rãnh tròn đồng tâm gọi là track, mỗi track được chia thành các cung gọi là sector, tập hợp các track cùng thứ tự trên các mặt đĩa gọi là cylinder (tử trụ).

Trên mỗi mặt đĩa có một đầu từ đọc/ghi dữ liệu (Read/Write Heads), để điều khiển đầu từ đọc/ghi dữ liệu cần có một trình điều khiển đĩa (Disk Controller).

Thông tin trên đĩa được tham chiếu bởi các thành phần: ổ đĩa, mặt đĩa, track, sector.

Hệ điều hành xem đĩa như mảng một chiều mà thành phần là các khối đĩa (Disk Block). Mỗi khối đĩa ghi các thông tin về mặt đĩa, track, sector mà hệ điều hành có thể định vị trên đó.

3. Thư mục thiết bị

Trên mỗi đĩa thông thường có một thư mục thiết bị (Device Directory) cho biết đĩa gồm những thông tin gì, độ dài, kiểu, người sở hữu, thời điểm khởi tạo, vị trí, được phân bổ không gian như thế nào?... Thư mục thiết bị được tạo ngay ở trên đĩa tại một vùng nhớ đặc biệt.

II. CÁC PHƯƠNG PHÁP QUẢN LÝ KHÔNG GIAN NHỚ TỰ DO

1. Phương pháp dùng bit vector (bitmap)

Không gian đĩa được chia thành các khối (block) và được đánh số từ 0... max.

Ví dụ: đĩa mềm 1.44Mb, 2 mặt, 80 track/1 mặt, 18 sector/1 track được đánh số như sau:

Head 0, track 0, sector 1	Block 0
.....	...
Head 0, track 0, sector 18	Block 18
Head 1, track 0, sector 1	Block 19
.....	...
Head 1, track 0, sector 18	Block 36
.....	...
Head 0, track 1, sector 1	Block 37
.....	...
.....	...
Head 1, track 79, sector 18	Block 2879

Mỗi khối đĩa sử dụng một bit để đánh dấu trạng thái. Khối đĩa nào đã sử dụng thì bit trạng thái có giá trị bằng 1, chưa sử dụng thì có giá trị bằng 0. Tập hợp các ký hiệu 0,1 tạo thành một bitvector (bitmap). Đọc thông tin trong bitmap hệ điều hành có thể xác định được không gian tự do trên đĩa.

Ví dụ: cho không gian đĩa từ như hình 5.1, các khối 2, 3, 4, 5, 8, 9, 10, 11,

12, 13, 17, 18, 25, 26, 27 là các khối đĩa tự do. Khi đó bitmap quản lý không gian nhớ tự do như sau:

1100001100000011100111110001111...

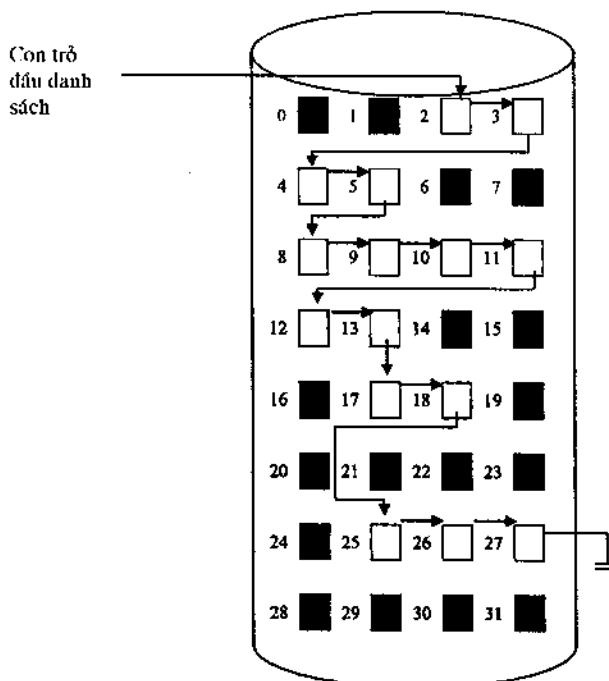
Phương pháp bitmap có ưu điểm là cài đặt đơn giản, dễ quản lý, dễ tìm kiếm những khối đĩa tự do liên tục trên đĩa nhưng tốn không gian lưu trữ dành cho bitmap (mỗi khối đĩa sẽ tốn một bit).

2. Phương pháp liệt kê (Free List)

Trong phương pháp này, hệ thống sử dụng một danh sách mốc nối để liệt kê các khối đĩa tự do. Con trỏ đầu trong danh sách chỉ tới khối đĩa tự do đầu tiên, mỗi khối có một con trỏ để trỏ tới khối kế tiếp (hình 5.1). Ưu điểm của phương pháp là tiết kiệm không gian nhớ nhưng làm tăng thời gian truy nhập dữ liệu.

3. Phương pháp lập nhóm (Grouping)

Trong phương pháp này, hệ thống cho phép nhóm các khối đĩa tự do liên tiếp thành một nhóm. Khối đĩa tự do đầu tiên trong nhóm lưu trữ địa chỉ của các khối đĩa tự do trong nhóm. Khối đĩa tự do cuối cùng trong nhóm lưu trữ địa chỉ của khối đĩa tự do đầu tiên của nhóm tiếp theo.



Hình 5.1 - Mô tả không gian đĩa tự

Ví dụ: Theo hình 5.1, ta có bảng quản lý không gian nhớ tự do như sau:

Nhóm	Khối đầu	Khối cuối
I	2 (2,3,4,5)	5 (8)
II	8 (8,9,10,11,12,13)	13 (17)
III	17 (17,18)	18 (25)
IV	25 (25,26,27)	27 (...)

4. Phương pháp đếm (Counting)

Phương pháp đếm là sự biến đổi của phương pháp lập nhóm. Trong phương pháp này, hệ thống lập danh sách quản lý địa chỉ của các khối đĩa tự do đầu tiên và số lượng các khối đĩa tự do liên tục kế tiếp các khối đĩa đó.

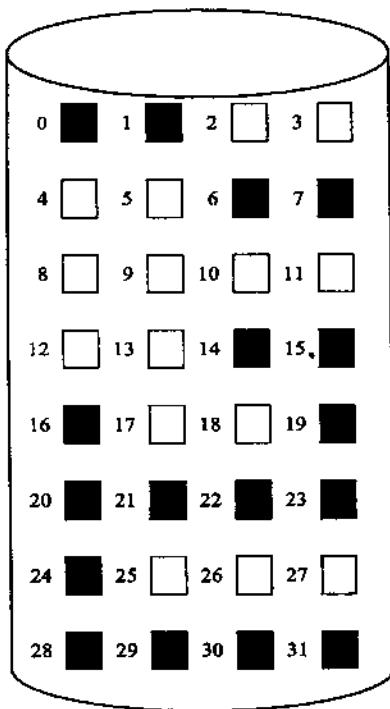
Ví dụ: Theo hình 5.1, ta có danh sách quản lý không gian nhớ tự do như sau:

Danh sách	Số lượng
2	4
8	6
17	2
25	3

III. CÁC PHƯƠNG PHÁP CẤP PHÁT KHÔNG GIAN NHỚ TỰ DO

1. Cấp phát liên tục (Contiguous)

Để phân bổ không gian nhớ cho một file, hệ thống chọn một đoạn liên tục các khối đĩa tự do để cấp phát cho file đó. Với phương pháp này, để định vị file hệ thống chỉ cần biết địa chỉ của khối đĩa tự do đầu tiên và số lượng block đã dùng.



Directory

File	Start	Length
f1	0	2
f2	14	3
f3	19	6
f4	28	4
f5	6	2

Hình 5.2 - Sơ đồ cấp phát liên tục

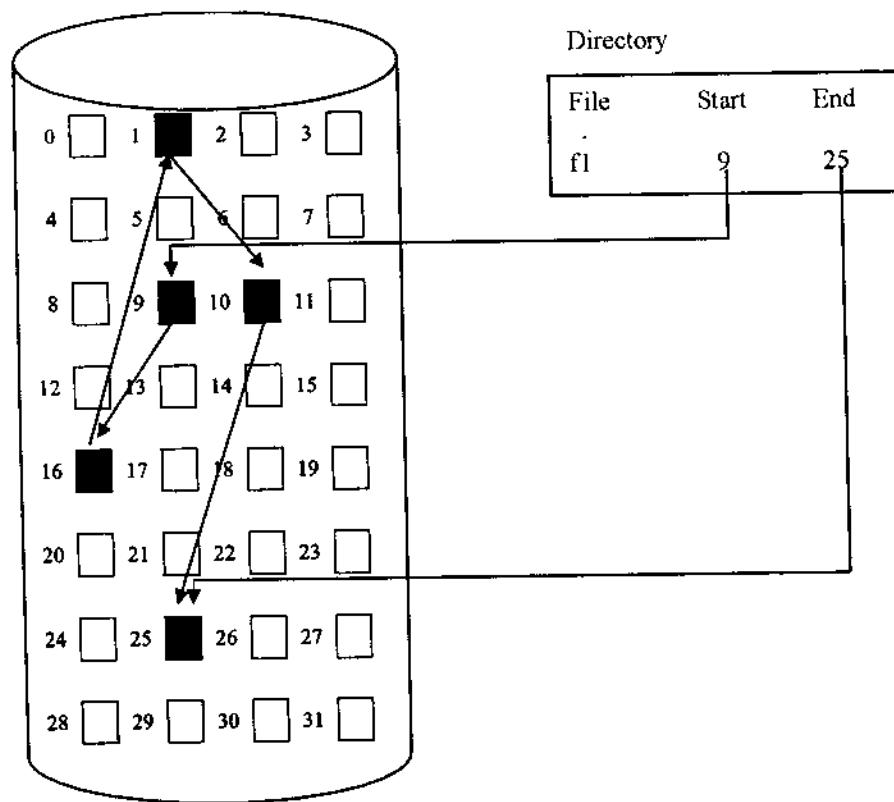
Ưu điểm của cấp phát liên tục là hỗ trợ cho phương pháp truy nhập tuần tự và truy nhập trực tiếp nhưng tồn tại ba nhược điểm chính:

- Phải chọn được thuật toán tối ưu để tìm các vùng không gian tự do cấp phát cho file (First Fit, Best Fit hoặc Worst Fit).
- Có thể xảy ra trường hợp không đủ số khối đĩa tự do liên tiếp cần thiết để cấp phát cho file (kích thước file lớn hơn vùng các khối đĩa liên tục lớn nhất).
- Trong trường hợp các khối đĩa tự do nằm rắn mạn sẽ không sử dụng được, gây lãng phí không gian nhớ.

2. Cấp phát liên kết (Linked)

Trong phương pháp này, mỗi file được định vị trong thư mục thiết bị bằng hai con trỏ, một cái trỏ tới khối đĩa đầu tiên, một cái trỏ tới khối đĩa cuối cùng đã cấp phát cho file. Trong mỗi khối đĩa đã cấp phát cũng có một con trỏ để trỏ tới khối đĩa kế tiếp.

Ví dụ: file f1 được cấp phát 5 khối đĩa có số hiệu 9, 16, 1, 11, 25; khối đầu là 9, khối cuối là 25.



Hình 5.3 - Sơ đồ cấp phát liên kết

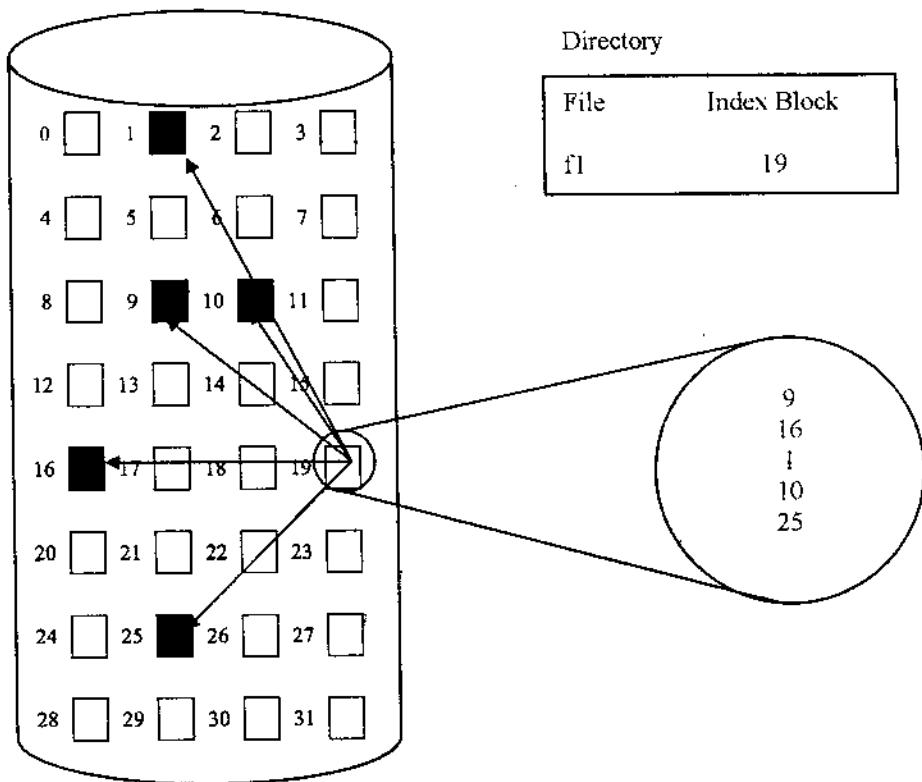
Cấp phát liên kết có ưu điểm là sử dụng được các khối đĩa tự do nằm rải mạn nhưng chỉ hỗ trợ truy nhập tuần tự, không hỗ trợ truy nhập trực tiếp; độ tin cậy không đảm bảo nếu bị mất các con trỏ liên kết. Mặt khác, phương pháp này tốn không gian nhớ để lưu trữ các con trỏ (khoảng 0,38%).

3. Cấp phát theo chỉ số (Index)

Trong phương pháp này, để cấp phát không gian nhớ cho một file, hệ thống sử dụng một khối đĩa đặc biệt gọi là khối đĩa chỉ số (index block) cho mỗi file. Trong khối đĩa chỉ số chứa địa chỉ của các khối đĩa đã cấp phát cho file, trong thư mục thiết bị địa chỉ của các khối đĩa chỉ số. Khi một khối đĩa được cấp phát cho file thì hệ thống loại bỏ địa chỉ của khối đĩa này khỏi danh sách các khối

đĩa tự do và cập nhật vào khối chỉ số của file.

Phương pháp cấp phát theo chỉ số hỗ trợ truy nhập trực tiếp nhưng lãng phí không gian nhớ dành cho khối đĩa chỉ số.



Hình 5.4 - Sơ đồ cấp phát theo chỉ số

IV. LẬP LỊCH CHO ĐĨA

1. Khái niệm về lập lịch cho đĩa (Disk Scheduling)

Thời gian truy nhập đĩa phụ thuộc ba yếu tố: thời gian di chuyển đầu từ đọc/ghi đến track hoặc cylinder cần thiết (seek-time), thời gian định vị đầu từ đọc/ghi tại khối đĩa cần truy nhập (latency-time) và thời gian truy nhập dữ liệu (transfer-time). Thời gian định vị đầu từ đọc/ghi và thời gian truy nhập dữ liệu thông thường cố định và phụ thuộc cấu trúc kỹ thuật của ổ đĩa. Do đó, để tăng tốc độ truy nhập đĩa, các hệ điều hành thường quan tâm tới thời gian di chuyển đầu từ đọc/ghi.

Như vậy, lập lịch cho đĩa là xây dựng các thuật toán dịch chuyển đầu từ đọc ghi sao cho thời gian truy nhập đĩa là tối ưu nhất.

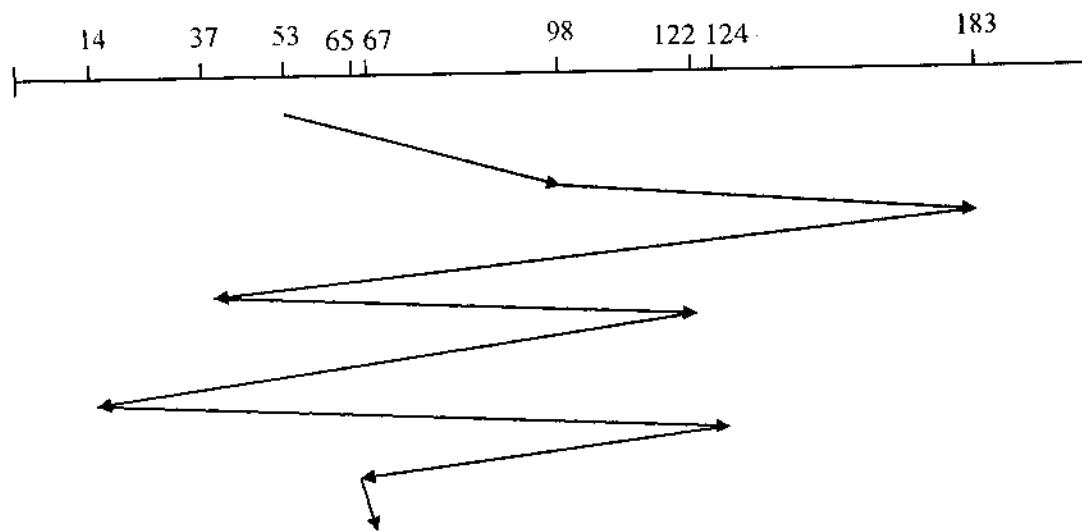
2. Một số phương pháp lập lịch

2.1. First Come First Served (FCFS)

Để truy nhập tới một file, hệ thống sẽ tổ chức một hàng đợi các yêu cầu phục vụ của các track (lưu trữ dữ liệu của file cần truy nhập). Track nào có yêu cầu phục vụ trước thì đầu từ đọc/ghi sẽ dịch chuyển tới đó trước.

Ví dụ: File F1 được phân bổ lần lượt tại các track có số thứ tự sau đây: 98, 183, 37, 122, 14, 124, 65, 67. Đầu từ đọc/ghi đang định vị tại track có số thứ tự 53

Sơ đồ dịch chuyển đầu từ đọc/ghi theo thuật toán FCFS được thể hiện như sau:

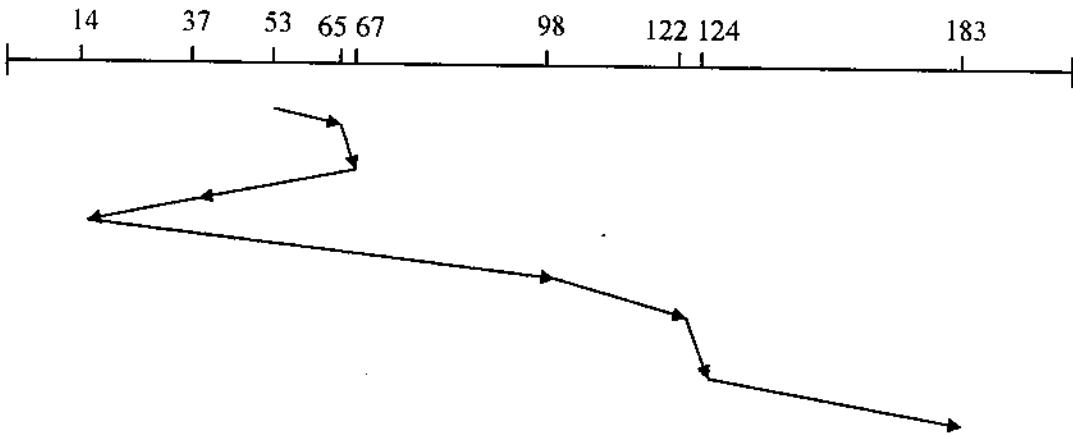


Hình 5.5 - Thuật toán lập lịch FCFS

2.2. Shortest Seek Time First (SSTF)

SSTF chọn track nào có thời gian di chuyển đầu từ đọc/ghi ngắn nhất thì phục vụ trước.

Theo ví dụ trên, sơ đồ dịch chuyển đầu từ đọc/ghi theo thuật toán SSTF được thể hiện như sau:

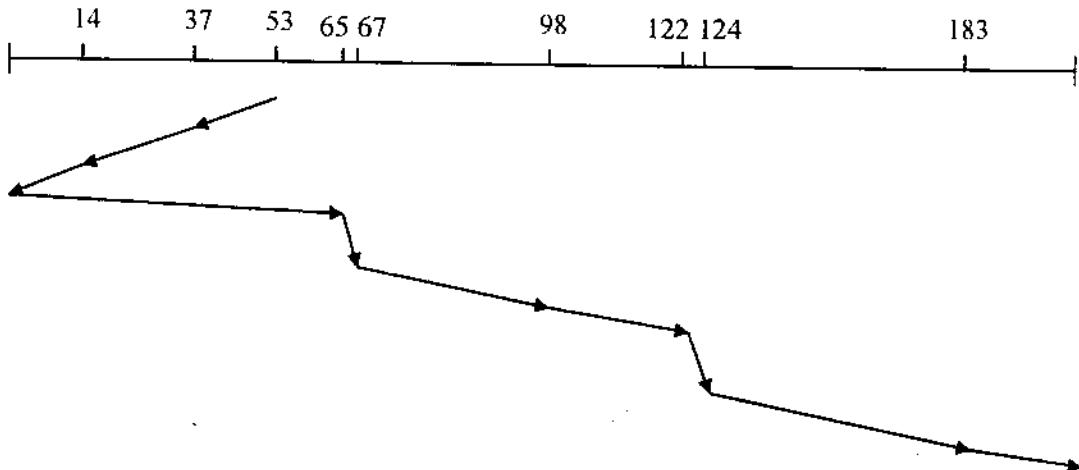


Hình 5.6 - Thuật toán lập lịch SSTF

2.3. Scan

Trong thuật toán này, đầu từ đọc/ghi quét từ track nhỏ nhất đến track lớn nhất, sau đó quét ngược lại, track nào có nhu cầu thì sẽ phục vụ.

Theo ví dụ trên, sơ đồ dịch chuyển đầu từ đọc/ghi theo thuật toán SSTF được thể hiện như sau:

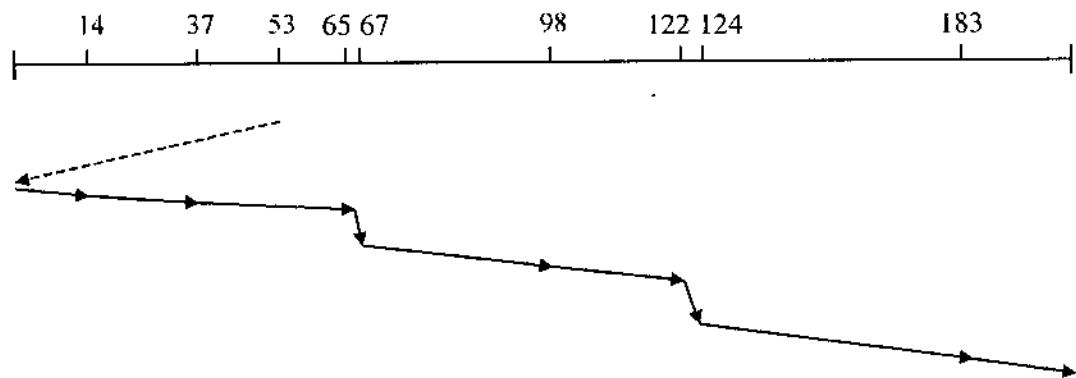


Hình 5.7 - Thuật toán lập lịch Scan

2.4. C - Scan

Thuật toán này tương tự như Scan nhưng đầu từ đọc/ghi không phục vụ đường về (không quét ngược lại).

Theo ví dụ trên, sơ đồ dịch chuyển đầu từ đọc/ghi theo thuật toán SSTF được thể hiện như sau:

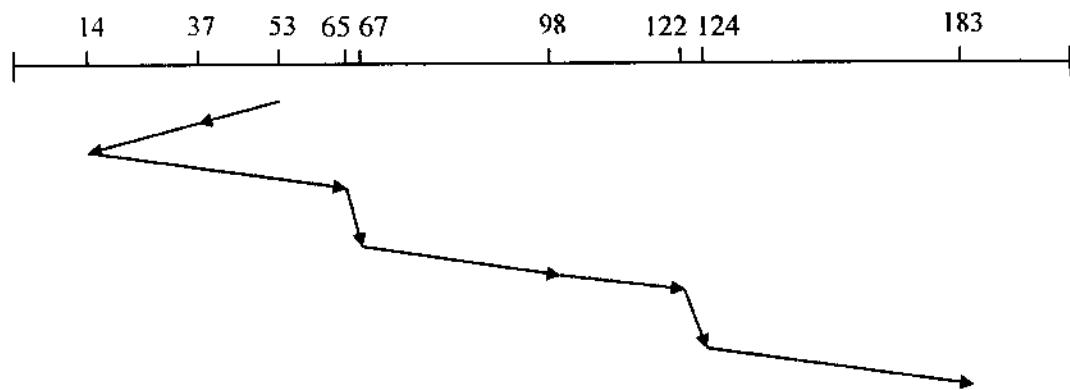


Hình 5.8 - Thuật toán lập lịch C-Scan

2.5. Look

Tương tự như Scan nhưng trong thuật toán này, đầu từ đọc/ghi chỉ quét trong phạm vi các track có nhu cầu phục vụ, không quét tới track đầu tiên hoặc cuối cùng (nếu các track này không có yêu cầu phục vụ).

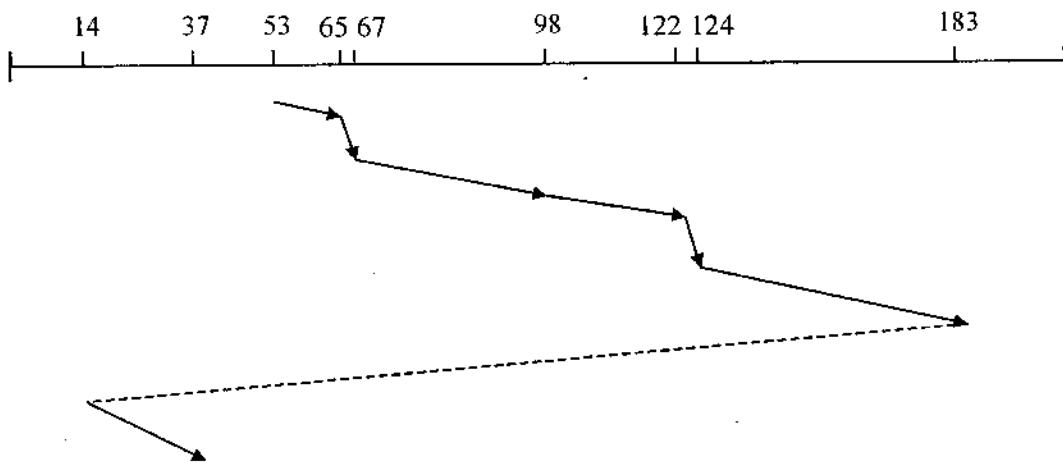
Theo ví dụ trên, sơ đồ dịch chuyển đầu từ đọc/ghi theo thuật toán SSTF được thể hiện như sau:



Hình 5.9 - Thuật toán lập lịch Look

2.6. C - Look

Tương tự như Look nhưng đầu từ đọc/ghi không phục vụ đường về. Theo ví dụ trên, sơ đồ dịch chuyển đầu từ đọc/ghi theo thuật toán SSTF được thể hiện như sau:



Hình 5.10 - Thuật toán lập lịch C - Look

Chú ý: Thuật toán FCFS, SSTF được áp dụng phổ biến, các thuật toán kiểu Scan, Look chỉ được áp dụng cho những đĩa chịu tải lớn.

V. HỆ FILE

1. Khái niệm hệ file (File System)

Dữ liệu máy tính được lưu trữ trên các thiết bị nhớ ngoài như: băng từ, đĩa từ, đĩa quang... và được tập hợp một cách có tổ chức theo đơn vị lưu trữ gọi là file. Như vậy, file là đơn vị logic để hệ điều hành quản lý thông tin trên đĩa. File có thể là một chương trình của người sử dụng, một chương trình của hệ thống hoặc một tập hợp dữ liệu của người sử dụng.

Trên phương diện người sử dụng, dữ liệu trong file được tổ chức thành các bản ghi logic mà mỗi bản ghi logic có thể là một byte hoặc một cấu trúc dữ liệu nào đó. Bản ghi logic chính là đơn vị dữ liệu mà các chương trình cần xử lý trong quá trình hoạt động của mình.

Để quản lý dữ liệu trên các phương tiện lưu trữ ngoài một cách có hiệu quả,

hệ điều hành cần phải tổ chức các file theo một nguyên tắc nhất định. Như vậy, hệ file là nguyên tắc mà hệ điều hành tổ chức và quản lý các file trên các phương tiện lưu trữ.

2. Các yêu cầu của hệ file

Mặc dù các hệ file có thể được tổ chức theo các nguyên tắc khác nhau nhưng cần phải đảm bảo các yêu cầu chung như sau:

- Hệ file phải được tổ chức sao cho dễ tìm kiếm, dễ lưu trữ, cập nhật, tiết kiệm không gian nhớ.
- Phải đảm bảo tính độc lập của hệ file với hệ thống và các thiết bị ngoại vi.
- Hệ file phải đảm bảo tính an toàn dữ liệu khi có sự cố chương trình hoặc kỹ thuật.
- Hệ file phải đảm bảo tính an toàn trong vấn đề truy nhập thông tin của người sử dụng.

3. Các thao tác của hệ file

Một hệ file dù phức tạp hay đơn giản cũng đều phải cung cấp cho người sử dụng những công cụ đơn giản để có thể thao tác với file. Trong các hệ file, thường có các thao tác sau:

- Tạo file: cho phép người sử dụng trực tiếp xây dựng file hoặc cung cấp dữ liệu.
- Đọc file: cho phép người sử dụng đọc các dữ liệu trong file, tạo các bản sao nhưng không được phép sửa đổi nội dung file.
- Bổ sung, cập nhật dữ liệu vào file: cho phép người sử dụng sửa đổi nội dung file, cập nhật thêm dữ liệu vào file.
- Thay đổi thuộc tính file: cho phép thay đổi các thuộc tính như: chỉ đọc, ẩn, hệ thống, lưu trữ và gán các quyền truy nhập file cho người sử dụng khác.
- Xoá file: cho phép loại bỏ file khỏi thiết bị lưu trữ.

Để truy nhập tới các file, hệ file sử dụng hai phương pháp:

- Truy nhập tuần tự: các bản ghi logic trong file được truy nhập lần lượt từ đầu đến cuối theo đúng trình tự sắp xếp trong file. Với cách thức truy nhập này thì hoàn toàn có thể biết trước được bản ghi logic kế tiếp truy nhập sẽ là bản ghi nào, và vì vậy hệ điều hành biết được vị trí trên bộ nhớ ngoài của bản ghi logic

kế tiếp cần xử lý. Cách thức truy nhập tuân tự có mức độ tự động hoá cao, tuy nhiên chỉ áp dụng được với các file được tổ chức theo kiểu tuân tự. Mặt khác, để đảm bảo được mức độ tự động hoá cao thì hệ thống phải đảm bảo thực hiện mọi công việc chuẩn bị liên quan đến bản ghi cho chương trình của người sử dụng.

- Truy nhập trực tiếp: theo cách thức truy nhập này, hệ thống hoàn toàn không có trước thông tin về bản ghi nào là bản ghi kế tiếp cần xử lý. Người lập trình cần phải tự xác định bản ghi cần xử lý và để tìm được nó, mọi vấn đề đồng bộ hoá phải được đặt ra.

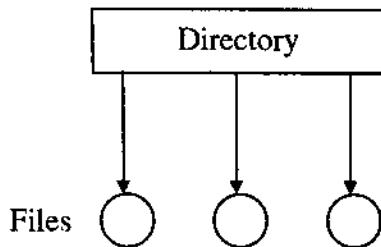
Tuy mức độ tự động hoá thấp nhưng cách thức truy nhập trực tiếp cho phép truy xuất các file hết sức mềm dẻo, linh hoạt, đạt được mức độ chủ động cao của chương trình người sử dụng đối với các file.

4. Quản lý file

Thông tin (các đặc trưng) về file được ghi trong thư mục thiết bị của ổ đĩa như: tên, kiểu, vị trí, kích thước, thời gian cập nhật, số lần sử dụng, thuộc tính... Do đó, quản lý file có nghĩa là tổ chức thư mục, thiết bị sao cho việc thao tác với các file là tối ưu nhất.

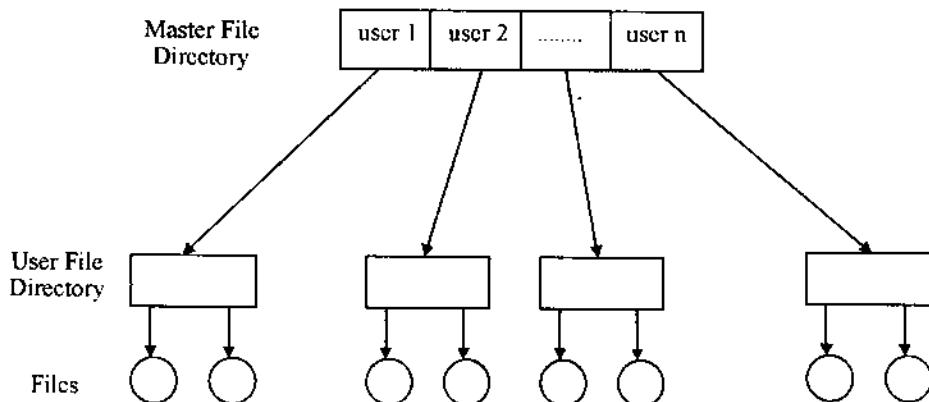
Các hệ điều hành trên thực tế tồn tại một số kiểu tổ chức:

- Tổ chức thư mục một mức (Single Level Directory): hệ điều hành chỉ thiết lập một thư mục dùng chung cho tất cả các file. Kiểu tổ chức này dễ cài đặt nhưng không thuận tiện cho người sử dụng.



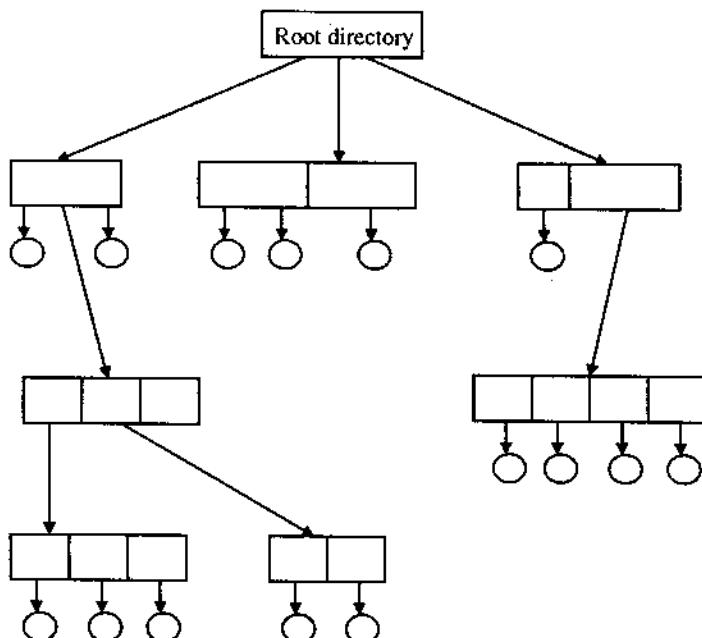
Hình 5.11 - Thư mục một cấp

- Tổ chức thư mục hai mức (Two Level Directory): hệ thống tổ chức hai mức thư mục, một mức cho hệ thống, một mức cho người sử dụng.



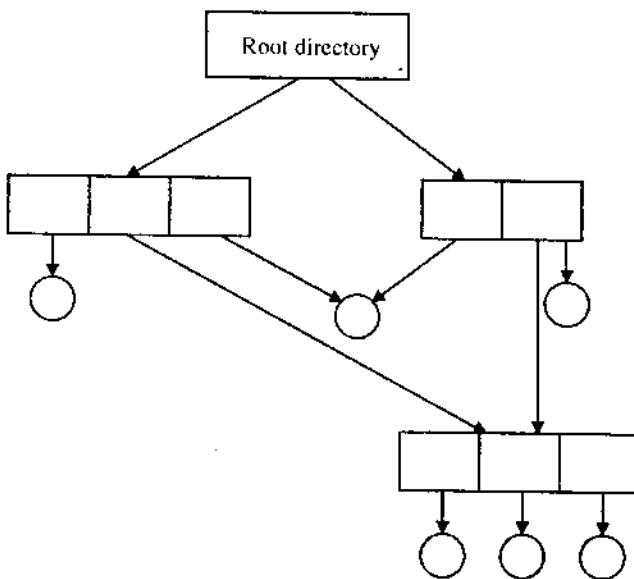
Hình 5.12 - Thư mục hai mức

- Tổ chức theo cấu trúc cây (Tree Directory): trên mỗi ổ đĩa có một thư mục gốc (root directory), trong thư mục gốc có các thư mục con cấp một, trong thư mục con cấp một tồn tại các thư mục con cấp hai... Tập hợp các thư mục trên đĩa tạo thành một cấu trúc cây thư mục.



Hình 5.13 - Cấu trúc cây

- Tổ chức theo đồ thị không chu trình (Acyclic Graph Directory): kiểu tổ chức này gần giống như tổ chức cây nhưng một thư mục con hoặc file có thể thuộc hai thư mục mẹ khác nhau.



Hình 5.14 - Cấu trúc thư mục đồ thị không chu trình

5. Bảo vệ file và đảm bảo tính toàn vẹn dữ liệu

Bảo vệ file thông qua việc giới hạn quyền truy nhập của người sử dụng như: đọc, viết, xử lý, bổ sung, xoá...

Một số phương pháp bảo vệ file được áp dụng là: đặt tên, đặt mật khẩu, liệt kê quyền truy nhập, lập nhóm truy nhập...

Để đảm bảo tính toàn vẹn dữ liệu, các hệ điều hành thường áp dụng các biện pháp như: lưu trạng thái của file qua các quá trình sử dụng, lưu trạng thái qua các thời điểm, lưu trữ file theo thế hệ...

Câu hỏi và bài tập

1. Trình bày sơ lược về cấu trúc và nguyên tắc hoạt động của đĩa từ. Hệ điều hành quản lý đĩa từ theo đơn vị nào? Thế nào là thư mục thiết bị?

2. Trình bày các phương pháp quản lý và cấp phát không gian nhớ tự do trên đĩa từ của hệ điều hành.

3. Trình bày các yếu tố liên quan đến thời gian truy nhập đĩa từ, từ đó nêu khái niệm về lập lịch cho đĩa (disk scheduler).

4. Trình bày khái niệm về hệ file, các yêu cầu của hệ file và các phương pháp tổ chức hệ file.

5. Giả sử vùng không gian nhớ của đĩa từ được mô tả qua hình vẽ sau (mỗi ô là một disk block).

0 1 2 3

4 5 6 7

8 9 10 11

12 13 14 15

File F1 được phân bổ tại các block có số hiệu: 0,2,4,5,9,13,14,15. Trình bày phương pháp cấp phát liên kết (block đầu là 0, block cuối là 2) và phương pháp cấp phát theo chỉ số (block chỉ số là 15).

6. Giả sử vùng không gian nhớ của đĩa từ được mô tả qua sơ đồ sau: (Các block sẫm màu là các block đã sử dụng).

0 1 2 3 4 5 6 7 8 9

10 11 12 13 14 15 16 17 18 19

20 21 22 23 24 25 26 27 28 29

a - Mô phỏng các phương pháp quản lý không gian nhớ tự do qua sơ đồ trên.

b - File F1 có kích thước 3 block. Mô phỏng các phương pháp cấp phát không gian nhớ cho F1 qua sơ đồ trên.

7. Giả sử một đĩa cứng có 200 track được ký hiệu từ 0 đến 199; các yêu cầu đọc ghi dữ liệu tại các track theo thứ tự sau đây: 45, 14, 9, 26, 87, 52, 122, 183, 68, 184, 185. Đầu từ đọc/ghi đang định vị tại track 60. Vẽ sơ đồ dịch chuyển đầu từ đọc/ghi theo các thuật toán: FCFS, SSTF, Scan, C-Scan, Look, C-Look.

Chương 6

QUẢN LÝ THIẾT BỊ

Mục tiêu

Sau chương này, người học có thể phân biệt được thế nào là thiết bị chuẩn và thiết bị mở rộng. Đồng thời có thể hình dung được nguyên tắc tổ chức, quản lý thiết bị ngoại vi của hệ điều hành và một số kỹ thuật áp dụng trong quản lý thiết bị ngoại vi.

Nội dung

Trình bày các yêu cầu về quản lý thiết bị, nguyên tắc tổ chức và quản lý thiết bị của hệ điều hành, các kỹ thuật áp dụng trong quản lý thiết bị.

I. NGUYÊN TẮC TỔ CHỨC VÀ QUẢN LÝ THIẾT BỊ

1. Yêu cầu của quản lý thiết bị

Chức năng của các thiết bị ngoại vi là đảm nhiệm việc truyền thông tin qua lại giữa các bộ phận của hệ thống. Do đó, yêu cầu của hệ điều hành là tìm phương pháp tổ chức và truy nhập thông tin trên các thiết bị.

Ngoài các thiết bị chuẩn có tính chất bắt buộc (màn hình, bàn phím, máy in...) thì các hệ thống máy tính phải có khả năng kết nối với số lượng tùy ý các thiết bị ngoại vi bổ sung. Các thiết bị này có thể khác nhau về bản chất và nguyên lý hoạt động, vì vậy hệ điều hành cần phải tìm cách quản lý, điều khiển và khai thác các thiết bị một cách có hiệu quả.

CPU không làm việc trực tiếp với các thiết bị ngoại vi, do đó cần phải tổ chức các thiết bị sao cho CPU không phụ thuộc vào sự biến động của các thiết bị.

2. Nguyên tắc tổ chức và quản lý thiết bị

Nguyên tắc cơ bản để tổ chức và quản lý thiết bị dựa trên cơ sở: CPU chỉ điều khiển các thao tác vào/ra chứ không trực tiếp thực hiện các thao tác này. Để đảm bảo được nguyên tắc này, các thiết bị không gắn trực tiếp với CPU mà gắn với các thiết bị đặc biệt - thiết bị quản lý (Control Device). Một thiết bị quản lý có thể kết nối với nhiều thiết bị vào/ra.

Thiết bị quản lý đóng vai trò như một máy tính chuyên dụng có nhiệm vụ điều khiển các thiết bị kết nối với nó và gọi là kênh vào/ra. Mỗi kênh vào ra có ngôn ngữ và hệ lệnh riêng. Chúng hoạt động độc lập với nhau, độc lập với CPU và độc lập với các thành phần khác trong hệ thống.

Ví dụ: để chuyển thông tin từ bộ nhớ trong ra ngoài và ngược lại, kênh phải truy nhập trực tiếp bộ nhớ theo một cơ chế đặc biệt, song song và độc lập với CPU. Cơ chế này được gọi là DMA (Direct Memory Access)

Một hệ thống máy tính có thể có nhiều kênh vào/ra, mỗi kênh vào/ra lại có thể có những kênh con của mình. Để điều khiển hoạt động của các kênh, cần có các chương trình điều khiển riêng gọi là chương trình điều khiển kênh.

Để hệ thống làm việc được với các kênh thì CPU phải hiểu được ngôn ngữ kênh. Ngôn ngữ kênh được nạp vào hệ thống khi nạp hệ điều hành hoặc ngay cả khi hệ điều hành đang hoạt động (ngôn ngữ kênh thực chất là các trình điều khiển kênh).

CPU điều khiển các thao tác vào/ra thông qua các chương trình điều khiển kênh tương ứng với công việc cần thực hiện (nguyên lý macro processor). Nguyên lý điều khiển này cho phép trong lúc các thao tác vào/ra được thực hiện ở thiết bị ngoại vi thì CPU vẫn hoạt động song song thực hiện tính toán và điều khiển chừng nào chưa cần tới kết quả vào/ra. Khi có kết quả vào/ra, kênh sẽ phát tín hiệu ngắt báo cho CPU biết. Tùy theo hoàn cảnh cụ thể, tín hiệu ngắt được xử lý ngay hoặc được lưu trữ lại để xử lý khi có điều kiện hoặc thậm chí có thể bị huỷ bỏ nếu hệ thống không còn quan tâm đến kết quả này. Như vậy, ngắt vào/ra xuất hiện sau khi phép vào/ra được thực hiện xong chứ không phải trước khi phép vào/ra được thực hiện. Để đảm bảo hiệu suất xử lý cao, hệ thống cần phải biết càng sớm càng tốt thời điểm kết thúc của phép vào/ra. Chính vì vậy, kênh sẽ báo cho hệ thống biết kết quả vào/ra vào thời điểm sớm nhất có thể được và do đó một phép vào/ra có thể kết thúc ở nhiều mức, nhiều nơi khác nhau như: tại thiết bị điều khiển, tại thời điểm khi lệnh được chuyển đến thiết bị vào/ra, khi thiết bị vào/ra nhận được tín hiệu điều khiển hoặc sau khi phép vào/ra được thực hiện xong tại thiết bị ngoại vi.

Phương pháp tổ chức này cho phép gắn thêm thiết bị đồng thời đảm bảo cho hệ thống không phụ thuộc cấu hình của thiết bị cụ thể, hệ thống có tính lưu động cao (thay đổi thiết bị mà không cần thay đổi hệ thống, không cần sửa đổi các chương trình ứng dụng).

II. CÁC KỸ THUẬT ÁP DỤNG TRONG QUẢN LÝ THIẾT BỊ

1. Kỹ thuật vùng đệm

1.1. Khái niệm và mục đích của vùng đệm

Vùng đệm (buffer) là một vùng nhớ trung gian dùng làm nơi lưu trữ thông tin tạm thời trong các thao tác vào/ra.

Để thực hiện một thao tác vào/ra, hệ thống cần phải thực hiện các bước sau:

- Kích hoạt thiết bị.
- Chờ thiết bị đạt trạng thái thích hợp.
- Chờ thao tác vào/ra được thực hiện.

Việc chờ đợi các thiết bị đạt trạng thái thích hợp chiếm một thời gian khá lớn trong tổng thời gian thực hiện thao tác vào/ra. Vì vậy, để đảm bảo tốc độ hoạt động chung của toàn hệ thống, thao tác vào/ra cần phải sử dụng vùng đệm nhằm mục đích:

- Giảm số lượng các thao tác vào/ra vật lý.
- Cho phép thực hiện song song các thao tác vào/ra với các thao tác xử lý thông tin khác nhau.

- Cho phép thực hiện trước các phép nhập dữ liệu.

1.2. Phân loại vùng đệm

Có nhiều phương pháp tổ chức vùng đệm khác nhau nhưng nói chung có thể chia chúng thành ba loại: vùng đệm trung chuyển, vùng đệm xử lý và vùng đệm vòng tròn.

1.2.1. Vùng đệm trung chuyển

Đối với kiểu vùng đệm trung chuyển, hệ thống tổ chức hai vùng nhớ riêng biệt: vùng nhớ vào và vùng nhớ ra. Vùng nhớ vào chỉ dùng để nhập thông tin còn vùng nhớ ra dùng để ghi thông tin. Tương ứng trong hệ thống có hai lệnh để đưa thông tin vào và lấy thông tin ra (read/write).

Trong chương trình ứng dụng, ngay sau khi mở file, thông tin sẽ được chuyển đến vùng nhớ vào. Khi gặp lệnh đọc (read), thông tin sẽ được chuyển từ vùng nhớ vào tới các địa chỉ tương ứng nêu trong chương trình ứng dụng, như vậy mỗi giá trị sẽ được lưu trữ hai nơi trong bộ nhớ. Sau khi giá trị cuối cùng của vùng đệm được lấy ra xử lý, vùng đệm trở nên rỗng và hệ thống tổ chức nhập thông tin mới vào thời điểm sớm nhất có thể được. Để giảm thời gian chờ đợi, hệ thống có thể tổ chức nhiều vùng đệm vào, khi hết thông tin ở một vùng đệm hệ thống sẽ chuyển sang vùng đệm kế tiếp.

Đối với vùng đệm ra, thông tin cũng được xử lý tương tự nhưng theo trình tự ngược lại. Lệnh ghi (write) không đưa trực tiếp thông tin ra thiết bị mà đưa vào vùng đệm ra. Khi một vùng đệm ra đầy, hệ thống sẽ chuyển sang làm việc với vùng đệm kế tiếp, đồng thời tổ chức đưa thông tin từ vùng đệm trước ra thiết bị.

Ưu điểm của vùng đệm trung chuyển là có hệ số song song cao, phổ dụng (áp dụng được cho mọi phép vào/ra), cách thức tổ chức đơn giản nhưng nhược điểm là tốn bộ nhớ (phải tổ chức hai vùng nhớ riêng), kéo dài thời gian trao đổi thông tin ở bộ nhớ trong.

1.2.2. Vùng đệm xử lý

Trong vùng đệm xử lý, cả thông tin vào và ra cùng được xử lý trong một vùng bộ nhớ, thông tin không cần phải lưu trữ ở nhiều vị trí khác nhau trong bộ nhớ. Trong trường hợp này, lệnh đọc (read) xác định địa chỉ thông tin chứ không cần cung cấp giá trị thông tin như trong vùng đệm trung chuyển.

Loại vùng đệm này có ưu điểm là tiết kiệm không gian nhớ, rút ngắn thời gian trao đổi thông tin ở bộ nhớ trong nhưng tốc độ giải phóng vùng đệm chậm, vì vậy hệ số song song thấp hơn so với vùng đệm trung chuyển. Mặt khác, không phải thao tác trao đổi vào/ra nào cũng có thể sử dụng được vùng đệm này. Đây là phương pháp tổ chức vùng đệm phức tạp.

1.2.3. Vùng đệm vòng tròn

Trong cách tổ chức này, hệ thống làm việc với ba vùng đệm: một vùng đệm để đưa thông tin vào, một vùng đệm để đưa thông tin ra và một vùng đệm để xử lý. Sau một khoảng thời gian nhất định thì chức năng của các vùng đệm được trao đổi cho nhau. Vòng tròn tức là vùng đệm vào thành vùng đệm xử lý, vùng đệm xử lý thành vùng đệm ra, vùng đệm ra thành vùng đệm vào. Như vậy, vùng đệm này sẽ đạt hiệu quả cao khi thời gian xử lý tương đương thời gian vào/ra.

Loại vùng đệm này có thể gắn với từng file cụ thể hoặc gắn với toàn hệ thống. Trong chế độ gắn với file, vùng đệm được xây dựng khi mở file, xoá khi đóng file và chỉ phục vụ riêng cho file đó. Phương pháp tổ chức này đặc biệt thích hợp khi mỗi file có một kích thước bản ghi vật lý riêng. Nếu tất cả các file đều có kích thước bản ghi vật lý giống nhau thì người ta thường dùng chế độ vùng đệm chung cho toàn bộ hệ thống. Vùng đệm được xây dựng khi nạp hệ thống và chưa gắn với một file cụ thể nào. Khi mở file, một hoặc một số vùng đệm được gắn với file và phục vụ cho sự truy nhập file đó. Khi đóng file, vùng đệm không bị xoá mà được trả về cho hệ thống như một tài nguyên chung.

Phương pháp tổ chức này tránh được việc phải thực hiện các thủ tục tạo vùng đệm nhiều lần nhưng nó cũng gặp một số hạn chế:

Có những thời điểm vùng đệm không được sử dụng hết, gây lãng phí bộ nhớ.

Vùng đệm có thể trở thành tài nguyên gãy khi có nhiều file được mở đồng thời. Để giảm khả năng xảy ra cạnh tranh vùng đệm, chúng ta có thể tăng số lượng vùng đệm ngay từ khi nạp hệ thống nhưng như vậy sẽ chiếm dụng nhiều bộ nhớ và làm tăng thời gian dịch vụ của hệ thống, đặc biệt là việc dàn thông tin vào các vùng đệm.

2. Kỹ thuật kết khối

Để giảm số lần truy nhập vật lý, hệ thống còn sử dụng kỹ thuật kết khối tức là ghép nhiều bản ghi logic thành một bản ghi vật lý và việc trao đổi thông tin giữa các bộ phận được tiến hành theo bản ghi vật lý.

Thông thường, tồn tại các cách tổ chức kết khối như sau:

- Mỗi bản ghi vật lý chứa một số nguyên lần các bản ghi logic và giá trị này là như nhau với mọi bản ghi vật lý.

- Mỗi bản ghi vật lý chứa một số nguyên lần các bản ghi logic nhưng số lượng các bản ghi logic không giống nhau với những bản ghi vật lý khác nhau.

- Bản ghi vật lý có độ dài cố định, không phụ thuộc vào độ dài của bản ghi logic. Vì vậy, bản ghi vật lý không nhất thiết phải chứa một số nguyên lần các bản ghi logic.

- Bản ghi vật lý chỉ chứa một phần bản ghi logic và vì vậy phải kết hợp nhiều bản ghi vật lý mới được một bản ghi logic.

Phương pháp kết khối được chọn phải tuỳ thuộc vào vấn đề cần giải quyết và phương thức hoạt động của thiết bị. Ví dụ như thiết bị là đĩa từ được quản lý theo kiểu phân trang thì chỉ áp dụng được phương pháp thứ ba, với băng từ thì có thể áp dụng phương pháp thứ hai.

Việc kết khối còn được sử dụng như một biện pháp hạn chế việc truy nhập bất hợp lệ. Nếu không nêu đúng hệ số kết khối (số bản ghi logic trong một bản ghi vật lý) thì hệ thống sẽ không tiếp tục thực hiện các phép truy nhập thông tin hoặc thông tin sẽ bị giải mã sai lệnh vì hệ số kết khối đã nêu không hợp lý.

Phương pháp kết khối thứ tư thường được áp dụng khi cần phải lưu trữ hoặc sao chép các file có kích thước lớn nhưng không muốn sử dụng các công cụ backup dữ liệu.

Thao tác kết khôi sẽ kéo theo các chi phí bổ sung như: cần phải có bộ nhớ lưu trữ các chương trình phục vụ kết khôi và mở khôi, tốn thời gian xử lý bản ghi, đặc biệt khi một bản ghi logic nằm trên nhiều bản ghi vật lý khác nhau. Tuy nhiên, việc giảm đáng kể số lần truy nhập vật lý là một ưu điểm rất lớn của kỹ thuật này.

3. Xử lý lỗi

Bất kỳ một thành phần nào của hệ thống cũng có thể thực hiện công việc một cách không chuẩn. Điều này không những chỉ đúng với các thiết bị kỹ thuật phần cứng mà còn đúng với cả các chương trình phần mềm, thậm chí cả với các chương trình điều khiển vốn được thiết kế chu đáo và kiểm tra kỹ lưỡng trước khi được đưa vào khai thác thực tế. Tuy nhiên, không có bộ phận nào lại bộc lộ nhiều sai sót trong hoạt động như các thiết bị vào/ra. Điều này cũng dễ hiểu vì các thiết bị vào/ra luôn chịu ảnh hưởng của yếu tố môi trường và có nhiều chi tiết bị hao mòn trong quá trình sử dụng như: các bộ phận chuyển động bị mòn, độ nhiễm từ trên các đĩa kém...

Phương pháp chủ yếu thường áp dụng trong chống lỗi vào/ra là giao trách nhiệm phát hiện lỗi cho hệ thống chứ không phải giao cho người sử dụng. Vì nguyên nhân sinh ra lỗi là rất nhiều nên hệ thống phải thực hiện linh hoạt các phép kiểm tra thiết bị (sử dụng cả phần cứng lẫn phần mềm). Các công đoạn kiểm tra được chú ý ngay từ giai đoạn thiết kế và chế tạo thiết bị.

Khi phát hiện lỗi, hệ thống cố gắng khắc phục bằng cách thực hiện lại nhiều lần thao tác vào/ra. Nếu vẫn có lỗi ổn định thì cố gắng khôi phục thông tin ban đầu, trong trường hợp không thể khắc phục được thì hệ thống thông báo lỗi cho người sử dụng tự giải quyết.

Để đảm bảo độ chính xác của thông tin lưu trữ, nhiều thiết bị tổ chức đọc lại thông tin ngay sau khi ghi và so sánh kết quả đọc với thông tin gốc hoặc so sánh tổng kiểm tra tính được khi đọc với tổng kiểm tra tính được theo thông tin gốc. Phương pháp này thường được áp dụng với các thiết bị có tốc độ nhanh như đĩa từ. Việc kiểm tra và so sánh thông thường do các thiết bị điều khiển vào/ra đảm nhiệm, sau đó mới thông báo lỗi cho hệ thống và hệ thống chịu trách nhiệm thực hiện các tác động tương ứng.

Với mục đích tránh mọi sai sót không đáng có (như cố gắng đọc đĩa từ trong khi chưa sẵn sàng). Trước và sau phép trao đổi vào/ra, hệ thống cũng có những thao tác kiểm tra đối với kênh vào/ra và phân tích kết quả xem đã có đủ điều

kiện truy nhập thiết bị hay chưa.

Việc áp dụng các mã sửa sai giúp hệ thống khắc phục các lỗi dữ liệu thường gặp, đặc biệt là đối với thông tin được lưu trữ dài hạn. Chính vì vậy, tuy tốn nhiều thời gian và chi phí xây dựng nhưng mã sửa sai vẫn được áp dụng rộng rãi khi cần phải lưu trữ thông tin dài hạn.

Cần lưu ý rằng hệ thống chỉ báo lỗi khi không tự khắc phục được. Trong đại đa số các trường hợp, hệ thống không kết thúc vào/ra mà nêu phương án cho người sử dụng tự giải quyết có tiếp tục công việc hay không và nếu có thì tiếp tục theo kiểu nào.

Tóm lại, việc kiểm tra và xử lý lỗi là một quá trình phức tạp liên quan chặt chẽ với đặc trưng của từng thiết bị cụ thể. Tuy vậy, mỗi thiết bị đều cung cấp một mã trả về (return code) cho hệ thống để các chương trình xử lý kết quả phân tích và đánh giá.

Để công việc phân tích, đánh giá không chiếm dụng giờ CPU, ảnh hưởng tới tốc độ hoạt động của hệ thống thì các thiết bị thường có xu hướng cục bộ hoá sai sót (phân tích, xử lý, đánh giá... ngay tại thiết bị).

* *Ví dụ về nguyên tắc hoạt động của cơ chế kiểm tra chẵn lẻ (Parity Checking):*

Phương pháp kiểm tra đơn giản nhất là VRC (Vertical Redundancy Check). Trong phương pháp này, mỗi chuỗi bit biểu diễn một ký tự trong dữ liệu cần kiểm tra được thêm vào một bit kiểm tra (gọi là parity bit). Bit này có giá trị bằng 0 nếu số lượng các bit 1 trong chuỗi bit là chẵn và ngược lại. Hệ thống sẽ căn cứ vào đó để phát hiện lỗi.

Vị trí bit trong ký tự	Chuỗi ký tự cần kiểm tra				
	A	S	C	I	I
1	1	1	1	1	1
2	0	0	0	0	0
3	0	1	0	0	0
4	0	0	0	1	1
5	0	0	0	0	0
6	0	1	1	0	0
7	1	1	1	1	1
VRC	0	0	1	1	1

Nhược điểm của VRC là không định vị được bit bị lỗi hoặc nếu có một số chẵn lân các bit trong chuỗi bit bị lỗi thì giá trị của parity bit vẫn không thay đổi.

Để khắc phục nhược điểm này, người ta sử dụng thêm phương pháp LRC (Longitudinal Redundancy Check). LRC áp dụng kiểm tra parity bit cho từng khối các ký tự. Nếu kết hợp cả hai phương pháp VRC - LRC sẽ cho phương pháp kiểm tra lỗi theo cả hai chiều, nâng cao hiệu quả đáng kể so với việc dùng riêng từng phương pháp.

Vị trí bit trong ký tự	Chuỗi ký tự cần kiểm tra					LRC
	A	S	C	I	I	
1	1	1	1	1	1	1
2	0	0	0	0	0	0
3	0	1	0	0	0	1
4	0	0	0	1	1	0
5	0	0	0	0	0	0
6	0	1	1	0	0	0
7	1	1	1	1	1	1
VRC	0	0	1	1	1	1

4. SPOOL (Simultaneous Peripheral Operations On Line)

Thông thường, các thiết bị vào/ra được xem xét như những công cụ kỹ thuật để nhận các chương trình kênh và dữ liệu, đồng thời là nơi gửi các mã trạng thái cho hệ thống phân tích. Nhưng trên thực tế, mọi chương trình và dữ liệu của nó đều hoạt động hoàn toàn tương tự như thiết bị vào/ra có thực. Như vậy, có thể dùng tiến trình để mô phỏng hoạt động vào/ra và ngược lại, mọi thiết bị đều có thể coi như các tiến trình. Trên thực tế, trong nhiều trường hợp, hệ thống đã mô phỏng hoạt động vào/ra bằng con đường chương trình. Các chương trình này có thể hoạt động song song và tuân thủ theo nguyên tắc của quản lý tiến trình.

Việc mô phỏng thiết bị ngoại vi đã làm xuất hiện thiết bị ảo. Mỗi thiết bị ngoại vi cộng với chương trình mô phỏng tương ứng sẽ tạo ra một thiết bị hoàn toàn khác trong tay người sử dụng.

Ngoài mục đích mô phỏng thiết bị, thiết bị ảo còn có hai ứng dụng khác là:

- Mô phỏng quá trình điều khiển và quản lý thiết bị mới đang chế tạo hoặc chưa có điều kiện lắp đặt.

- Tạo ra các SPOOL (Simultaneous Peripheral Operations On Line - hệ

thống mô phỏng các phép trao đổi ngoại vi trong chế độ trực tiếp).

Nhiệm vụ của SPOOL là tạo ra hiệu ứng sử dụng song song, các thiết bị chỉ được phép khai thác trong chế độ tuân tự. Kỹ thuật SPOOL mô phỏng các thiết bị này bằng các thiết bị ảo và cung cấp cho các tiến trình có yêu cầu. Các tiến trình sẽ gửi thông tin của mình ra thiết bị ảo giống như đối với thiết bị thật và vào thời điểm thích hợp, thông tin từ thiết bị ảo sẽ được chuyển sang thiết bị thật.

Ví dụ máy in là một thiết bị chỉ có thể hoạt động trong chế độ tuân tự. Khi có nhiều tiến trình cùng có nhu cầu sử dụng máy in thì hệ thống không thể cấp phát nó cho tất cả các tiến trình có nhu cầu. Vì như vậy máy in sẽ không thể hoạt động được hoặc nếu có thì kết quả in ra cũng không thể sử dụng được. Đối với trường hợp này, hệ thống sẽ mô phỏng các máy in ảo và cung cấp cho mỗi tiến trình có nhu cầu in một máy ảo. Các tiến trình sẽ gửi thông tin của mình ra máy in ảo giống như máy in thật. Như vậy, các tiến trình có thể hoạt động song song mà không cần xếp hàng đợi chờ tài nguyên máy in.

SPOOL được sử dụng rộng rãi để thay thế nhiều loại thiết bị không có khả năng sử dụng chung để nâng cao khả năng hoạt động song song của các tiến trình. Ngoài ra đối với các thiết bị phụ thuộc tốc độ thông tin đầu vào, các tiến trình sẽ nhận được những SPOOL thích hợp để đảm bảo hoạt động bình thường.

Câu hỏi ôn tập

1. Phân biệt hai loại thiết bị chuẩn và không chuẩn của hệ thống máy tính.
2. Nêu nguyên tắc tổ chức và quản lý thiết bị ngoại vi của hệ điều hành.
3. Trình bày khái niệm và mục đích của vùng đệm trong các phép trao đổi ngoại vi. Phân loại các kiểu vùng đệm và nêu rõ ưu nhược điểm.
4. Thế nào là SPOOL? Trình bày các ứng dụng của SPOOL, cho ví dụ minh họa.
5. Mục đích của kết khối là gì? Nêu các phương pháp kết khối và chỉ rõ ưu điểm, nhược điểm của từng phương pháp.
6. Thế nào là thiết bị có tính năng Plug and Play (PnP). Nêu rõ ưu điểm của tính năng này trong cài đặt thiết bị. Trong các hệ điều hành đã học, hệ điều hành nào hỗ trợ tính năng này.
7. Trình bày các bước cài đặt một thiết bị trong hệ điều hành DOS và Windows.

Chương 7

BẢO VỆ VÀ AN TOÀN HỆ THỐNG

Mục tiêu

Sau chương này, người học có thể nắm bắt được mục đích và các phương pháp bảo vệ hệ thống, tránh được sự can thiệp bất hợp pháp từ bên ngoài cũng như các nguyên nhân tiềm ẩn bên trong. Đồng thời, hiểu được cơ chế an toàn hệ thống và phòng tránh virus máy tính.

Nội dung

Trình bày mục tiêu của bảo vệ hệ thống, các phương pháp bảo vệ, các vấn đề về an toàn hệ thống và virus máy tính.

I. BẢO VỆ HỆ THỐNG

1. Mục tiêu của bảo vệ hệ thống

Khi các tiến trình hoạt động song hành trong hệ thống thì một tiến trình gặp lỗi có thể ảnh hưởng đến các tiến trình khác và ảnh hưởng tới toàn bộ hệ thống. Hệ điều hành cần phải phát hiện, ngăn chặn không cho lỗi lan truyền và đặc biệt là phát hiện các lỗi tiềm ẩn trong hệ thống để tăng cường độ tin cậy.

Mặt khác, mục tiêu của bảo vệ hệ thống còn là chống sự truy nhập bất hợp lệ, đảm bảo cho các tiến trình khi hoạt động trong hệ thống sử dụng tài nguyên phù hợp với quy định của hệ.

Bảo vệ hệ thống cần phải cung cấp một cơ chế và chiến lược để quản trị việc sử dụng tài nguyên, quyết định những đối tượng nào trong hệ thống được bảo vệ và quy định các thao tác thích hợp trên các đối tượng này.

2. Miền bảo vệ

2.1. Khái niệm miền bảo vệ

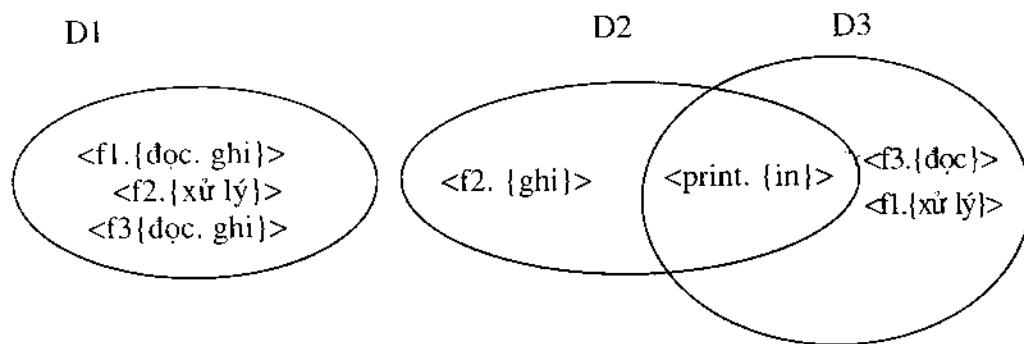
Một hệ thống máy tính bao gồm tập hợp các chủ thể (subject's) và tập hợp các khách thể (object's). Chủ thể bao gồm các tiến trình và người sử dụng còn khách thể có thể coi là các tài nguyên của máy tính (như bộ nhớ, ổ đĩa, dữ liệu...).

Để có thể kiểm soát được tình trạng sử dụng tài nguyên trong hệ thống, hệ điều hành chỉ cho phép các chủ thể truy nhập tới các khách thể mà nó có quyền sử dụng và vào những thời điểm cần thiết (nguyên lý need - to - know) nhằm hạn chế các lỗi xảy ra do tranh chấp tài nguyên.

Mỗi chủ thể trong hệ thống sẽ hoạt động trong một miền bảo vệ (protection domain) nào đó. Một miền bảo vệ sẽ xác định các khách thể mà chủ thể trong miền đó được phép truy nhập và thực hiện các thao tác.

2.2. Cấu trúc miền bảo vệ

Các khả năng thao tác mà chủ thể có thể thực hiện trên khách thể được gọi là quyền truy nhập (access right). Mỗi quyền truy nhập được định nghĩa bởi một bộ hai thành phần <đối tượng, {quyền thao tác}>. Như vậy, ta có thể hình dung miền bảo vệ là một tập hợp các quyền truy nhập, xác định các thao tác mà chủ thể có thể thực hiện trên các khách thể. Các miền bảo vệ khác nhau có thể giao nhau một số quyền truy nhập.



Hình 7.1 - Ví dụ về miền bảo vệ

Một tiến trình hoạt động và miền bảo vệ có thể tồn tại hai mối liên kết:

- Liên kết tĩnh: trong suốt thời gian tồn tại của tiến trình trong hệ thống, tiến trình chỉ hoạt động trong một miền bảo vệ. Trong trường hợp tiến trình trải qua các giai đoạn xử lý khác nhau, ở mỗi giai đoạn nó có thể thao tác trên những tập tài nguyên khác nhau. Như vậy trong liên kết tĩnh, miền bảo vệ phải xác định ngay từ đầu các quyền truy nhập cho tiến trình trong tất cả các giai đoạn xử lý. Điều này khiến cho tiến trình sẽ được dư thừa quyền trong một giai đoạn xử lý nào đó và vi phạm nguyên lý need - to - know. Để đảm bảo được nguyên lý này cần phải có khả năng cập nhật nội dung miền bảo vệ qua các giai đoạn

xử lý khác nhau để đảm bảo các quyền tối thiểu của tiến trình trong miền bảo vệ tại một thời điểm.

- Liên kết động: cơ chế này cho phép tiến trình chuyển đổi từ miền bảo vệ này sang miền bảo vệ khác trong suốt thời gian tồn tại trong hệ thống của nó. Để tuân thủ nguyên lý need - to - know, thay vì phải sửa đổi nội dung miền bảo vệ, hệ thống có thể tạo ra các miền bảo vệ mới với nội dung thay đổi tùy theo từng giai đoạn xử lý của tiến trình và chuyển tiến trình sang hoạt động tại các miền bảo vệ phù hợp với từng thời điểm.

3. Ma trận quyền truy nhập

3.1. Khái niệm về ma trận quyền truy nhập

Để biểu diễn miền bảo vệ, các hệ điều hành sẽ cài đặt các ma trận quyền truy nhập, trong đó các hàng của ma trận biểu diễn các miền bảo vệ, các cột biểu diễn khách thể. Phần tử (i,j) của ma trận xác định quyền truy nhập của chủ thể thuộc miền bảo vệ D_i , có thể thao tác đối với khách thể O_j .

Object Subject	F1	F2	F3	Printer
D1	đọc/ ghi	xử lý	đọc/ ghi	
D2		ghi		in
D3	xử lý		đọc	in

Hình 7.2 - Ma trận quyền truy nhập

Cơ chế bảo vệ được cung cấp khi ma trận quyền truy nhập được cài đặt, khi đó chúng ta có thể áp dụng các chiến lược bảo vệ bằng cách đặc tả nội dung các phân tử tương ứng của ma trận - xác định các quyền truy xuất ứng với từng miền bảo vệ.

Ma trận quyền truy nhập cũng cung cấp một cơ chế thích hợp để định nghĩa và thực hiện một sự kiểm soát nghiêm ngặt mối liên hệ giữa các chủ thể và khách thể.

3.2. Các phương pháp cài đặt ma trận quyền truy nhập

- Bảng toàn cục (Global Table): Phương pháp này đơn giản nhất, để cài đặt ma trận quyền truy nhập, hệ thống sử dụng một bảng toàn cục bao gồm các bộ

ba thành phần <miền bảo vệ, khách thể, quyền truy nhập>. Mỗi khi thực hiện thao tác M trên khách thể O_j trong miền bảo vệ D_i, cần tìm trong bảng toàn cục một bộ ba <D_i, O_j, R_k> mà M thuộc R_k (tập các quyền truy nhập). Nếu tìm thấy, thao tác M được phép thi hành, ngược lại xảy ra lỗi truy nhập.

- Danh sách quyền truy nhập (Access Control List - ACL): Trong phương pháp này, mỗi cột trong ma trận quyền truy nhập được xem như một danh sách các quyền truy nhập tới một khách thể. Mỗi khách thể trong hệ thống sẽ có một danh sách bao gồm các phần tử là các bộ hai thành phần <miền bảo vệ, các quyền truy nhập>, danh sách này sẽ xác định các quyền truy nhập được quy định trong từng miền bảo vệ có thể tác động trên khách thể. Mỗi khi thực hiện thao tác M trên khách thể O_j trong miền bảo vệ D_i, cần tìm trong danh sách quyền truy nhập của khách thể O_j một bộ hai <D_i, R_k> mà M thuộc R_k. Nếu tìm thấy, thao tác M được phép thi hành, ngược lại xảy ra lỗi truy nhập.

- Danh sách khả năng (Capability List): Mỗi dòng trong ma trận quyền truy nhập tương ứng với một miền bảo vệ sẽ được tổ chức thành một danh sách khả năng (capability list). Mỗi danh sách khả năng bao gồm các khách thể và các thao tác được phép thực hiện trên khách thể khi chủ thể hoạt động trong miền bảo vệ. Mỗi phần tử thuộc danh sách biểu diễn một khách thể và các quyền truy nhập hợp lệ trên khách thể này. Chủ thể chỉ có thể thực hiện thao tác M trên khách thể O_j trong miền bảo vệ D_i nếu trong danh sách khả năng của D_i có chứa khả năng tương ứng của O_j. Danh sách khả năng được gán tương ứng với từng miền bảo vệ và thực chất nó cũng là một đối tượng được hệ thống bảo vệ. Hệ điều hành cung cấp các thủ tục cho phép tạo lập, hủy bỏ, sửa đổi các khả năng của một khách thể và chỉ các chủ thể đóng vai trò hệ điều hành mới có thể sửa đổi nội dung của danh sách khả năng.

- Cơ chế khoá và chìa (A Lock/Key Mechanism): Phương pháp này thực chất là sự kết hợp giữa danh sách quyền truy nhập và danh sách khả năng. Mỗi khách thể sở hữu một danh sách các mã nhị phân được gọi là khoá (lock); tương ứng mỗi miền bảo vệ sẽ sở hữu một danh sách mã nhị phân gọi là chìa (key). Một chủ thể hoạt động trong miền bảo vệ chỉ có thể truy nhập tới một khách thể nếu miền bảo vệ sở hữu một chìa tương ứng với một khoá trong danh sách của khách thể. Cũng như phương pháp danh sách khả năng, phương pháp khoá và chìa được quản lý bởi hệ điều hành, người sử dụng không thể truy nhập trực tiếp để thay đổi nội dung của nó.

4. Thu hồi quyền truy nhập

Trong nội dung bảo vệ hệ thống, đôi khi việc thu hồi một số quyền thao tác trên các khách thể của các chủ thể cũng được xem là một biện pháp bảo vệ. Khi thu hồi quyền truy nhập, cần chú ý tới một số vấn đề sau:

- Thu hồi tức khắc hay trì hoãn và nếu trì hoãn thì tới khi nào?

- Nếu loại bỏ một quyền truy nhập của chủ thể tới một khách thể thì loại bỏ tất cả hay chỉ áp dụng với một số chủ thể.

- Thu hồi một số quyền hay toàn bộ quyền trên một khách thể?

- Thu hồi tạm thời hay vĩnh viễn một quyền truy nhập?

Đối với các hệ thống sử dụng danh sách quyền truy nhập, việc thực hiện thu hồi quyền truy nhập có thể thực hiện một cách dễ dàng bằng cách tìm và hủy trong ACL. Như vậy, việc thu hồi sẽ có hiệu lực tức thời và có thể áp dụng cho tất cả các chủ thể hoặc một nhóm các chủ thể; thu hồi một cách vĩnh viễn hay tạm thời đều được.

Tuy nhiên, trong các hệ thống sử dụng danh sách khả năng, vấn đề thu hồi sẽ gặp khó khăn vì các khả năng được phân tán trên khắp các miền bảo vệ trong hệ thống, do đó cần phải tìm ra chúng trước khi loại bỏ. Để giải quyết vấn đề này có thể tiến hành theo các phương pháp:

- Tái yêu cầu: loại bỏ các khả năng ra khỏi miền bảo vệ sau mỗi chu kỳ, nếu miền bảo vệ vẫn còn cần khả năng nào thì nó sẽ tái yêu cầu khả năng đó.

- Sử dụng con trỏ ngược: với mỗi khách thể sẽ tồn tại các con trỏ, trỏ đến các khả năng tương ứng trên khách thể. Khi cần thu hồi quyền truy nhập nào trên khách thể, hệ thống sẽ dựa vào các con trỏ để tìm kiếm các khả năng tương ứng.

- Sử dụng con trỏ gián tiếp: trong phương pháp này, con trỏ không trỏ trực tiếp tới các khả năng của khách thể mà trỏ tới một bảng toàn cục được quản lý bởi hệ điều hành. Khi cần thu hồi quyền truy nhập chỉ cần xoá phần tử tương ứng trong bảng này.

Trong các hệ thống sử dụng cơ chế khoá và chìa, khi cần thu hồi quyền, chỉ cần thay đổi khoá và bắt buộc chủ thể yêu cầu chìa khoá mới.

II. AN TOÀN HỆ THỐNG

Bảo vệ hệ thống là một cơ chế kiểm soát việc sử dụng tài nguyên của các chủ thể (tiến trình và người sử dụng) để đối phó với các tình huống lối có thể phát sinh trong hệ thống. Trong khi đó, khái niệm an toàn hệ thống muốn đề

cập tới mức độ tin cậy mà hệ thống cần duy trì khi phải đối phó không những với các vấn đề nội bộ mà còn cả với những tác động đến từ môi trường bên ngoài.

1. Các vấn đề về an toàn hệ thống

Hệ thống được coi là an toàn nếu các tài nguyên được sử dụng đúng quy định trong mọi hoàn cảnh, điều này khó có thể đạt được trong thực tế. Thông thường, cơ chế an toàn hệ thống bị vi phạm vì các nguyên nhân vô tình hoặc cố ý. Việc ngăn chặn các nguyên nhân cố ý là rất khó khăn và gần như không thể đạt hiệu quả hoàn toàn.

Bảo đảm an toàn hệ thống ở cấp cao như chống lại các nguyên nhân hoả hoạn, thiên tai, mất điện... cần được thực hiện ở mức độ vật lý (trang bị các thiết bị đảm bảo an toàn cho hệ thống) và nhân sự (chọn lựa các nhân viên tin cậy làm việc trong hệ thống). Nếu an toàn môi trường được đảm bảo thì an toàn của hệ thống sẽ được duy trì tốt nhờ các cơ chế của hệ điều hành.

Cần chú ý rằng nếu bảo vệ hệ thống có thể đạt độ tin cậy 100% thì các cơ chế an toàn hệ thống được cung cấp chỉ nhằm ngăn chặn bớt các tình huống bất lợi hơn là đạt đến độ an toàn tuyệt đối.

2. Các cơ chế an toàn hệ thống

2.1. Kiểm định danh tính

Để đảm bảo an toàn, hệ điều hành cần phải giải quyết tốt vấn đề kiểm định danh tính (authentication). Hoạt động của hệ thống bảo vệ phụ thuộc vào khả năng xác định các tiến trình đang xử lý. Khả năng này, đến lượt nó lại phụ thuộc vào việc xác định người dùng đang sử dụng hệ thống để có thể kiểm tra người dùng này được phép thao tác trên những tài nguyên nào.

Cách tiếp cận phổ biến nhất để giải quyết vấn đề là sử dụng mật khẩu (password) để kiểm định danh tính của người sử dụng. Mỗi khi người dùng muốn sử dụng một tài nguyên, hệ thống sẽ so sánh mật khẩu của họ nhập vào với mật khẩu được lưu trữ, nếu đúng họ mới được phép sử dụng tài nguyên. Mật khẩu có thể được áp dụng để bảo vệ cho từng đối tượng trong hệ thống, thậm chí cùng một đối tượng sẽ có các mật khẩu khác nhau tương ứng với các quyền truy nhập khác nhau.

Cơ chế mật khẩu rất đơn giản và dễ sử dụng, do đó được các hệ điều hành áp dụng rộng rãi, tuy nhiên điểm yếu nghiêm trọng của nó là khả năng bảo mật mật khẩu rất khó đạt được sự hoàn hảo. Những tác nhân tiêu cực có thể tìm ra mật khẩu của người khác nhờ nhiều cách thức khác nhau.

2.2. Ngăn chặn nguyên nhân từ phía các chương trình

Trong môi trường hoạt động mà một chương trình được tạo lập bởi một người lại được người khác sử dụng rất có thể sẽ xảy ra các tình huống sử dụng sai chức năng, từ đó dẫn tới những hậu quả không lường trước. Hai trường hợp điển hình gây mất an toàn hệ thống có thể đề xuất là:

- Ngựa thành Troy: khi người sử dụng A kích hoạt một chương trình (do người sử dụng B viết) dưới danh nghĩa của mình (trong miền bảo vệ được gán tương ứng cho người sử dụng A), chương trình này có thể trở thành “chú ngựa thành Troy” vì khi đó các đoạn lệnh trong chương trình có thể thao tác với các tài nguyên mà người sử dụng A có quyền nhưng người sử dụng B lại bị cấm. Những chương trình kiểu này đã lợi dụng hoàn cảnh để gây ra các tác hại đáng tiếc.

- Cánh cửa nhỏ (Trap - door): mối đe doạ đặc biệt nguy hiểm và khó chống đỡ do vô tình hoặc cố ý của các lập trình viên khi xây dựng chương trình. Các lập trình viên có thể để lại một “cánh cửa nhỏ” trong phần mềm của họ để thông qua đó can thiệp vào hệ thống. Chính “cánh cửa nhỏ này” đã tạo cơ chế cho các hacker thăm nhập và phá hoại hệ thống của người sử dụng. Việc phát hiện các “cánh cửa nhỏ” để đối phó rất phức tạp vì chúng ta cần phải tiến hành phân tích chương trình nguồn để tìm ra chỗ sơ hở.

2.3. Ngăn chặn nguyên nhân từ phía hệ thống

Hầu hết các tiến trình hoạt động trong hệ thống đều có thể tạo ra các tiến trình con. Trong cơ chế hoạt động này, các tài nguyên hệ thống rất dễ bị sử dụng sai mục đích gây mất an toàn cho hệ thống. Hai mối đe doạ phổ biến theo phương pháp này là:

- Các chương trình sâu (worm): một chương trình sâu là chương trình lợi dụng cơ chế phát sinh ra các tiến trình con của hệ thống để đánh bại chính hệ thống. Chương trình sâu có khả năng tự động phát sinh các bản sao trong hệ thống sau đó chiếm dụng tài nguyên, làm ngừng trệ hoạt động của các tiến trình khác và toàn bộ hệ thống.

- Các chương trình virus: virus là một chương trình phá hoại khá nguy hiểm đối với các hệ thống thông tin. Khác với các chương trình sâu là những chương trình hoàn chỉnh, virus chỉ là các đoạn mã có khả năng lây nhiễm vào các chương trình chính thống và từ đó tàn phá hệ thống.

2.4. Giám sát các nguyên nhân

Nhìn chung, việc đảm bảo an toàn hệ thống là rất phức tạp vì nó liên quan tới yếu tố con người. Hệ điều hành chỉ có thể áp dụng một số biện pháp để giảm bớt thiệt hại như lập nhật ký sự kiện để ghi nhận các tình huống xảy ra trong hệ thống. Ví dụ như theo dõi:

- Người sử dụng cố gắng nhập mật khẩu nhiều lần.
- Các tiến trình với định danh nghi ngờ không được ủy quyền.
- Các tiến trình lạ trong các thư mục hệ thống.
- Các chương trình kéo dài thời gian xử lý một cách đáng ngờ.
- Các tệp tin và thư mục bị khoá không hợp lý.
- Kích thước các chương trình hệ thống bị thay đổi...

Việc kiểm tra thường kỳ và ghi nhận những thông tin này giúp hệ thống phát hiện kịp thời các nguy cơ, cho phép phân tích, dự đoán và tìm phương pháp đối phó.

III. VIRUS MÁY TÍNH

1. Khái niệm về virus

Virus máy tính là một chương trình có khả năng gián tiếp tự kích hoạt, tự lan truyền trong môi trường của hệ thống tính toán và làm thay đổi môi trường hệ thống hoặc cách thực hiện chương trình.

Virus tự kích hoạt và lan truyền trong môi trường làm việc của hệ thống mà người sử dụng không hề hay biết. Thông thường, virus nào cũng mang tính chất phá hoại, nó gây ra lỗi khi thực hiện chương trình, điều này có thể dẫn đến việc chương trình hoặc dữ liệu bị hỏng, không khôi phục được, thậm chí chúng có thể bị xoá. Như vậy, virus là chương trình thông minh, mang yếu tố tự thích nghi, lan truyền xa và do đó khả năng phá hoại của nó là rất lớn.

Khái niệm “gián tiếp kích hoạt” ở đây có nghĩa là trừ người viết virus và lần đầu tiên đưa vào hệ thống phải trực tiếp nạp chương trình virus và thực hiện nó, còn những người sử dụng khác chỉ nạp chương trình của mình. Nếu đó là chương trình bị nhiễm virus thì virus sẽ chiếm quyền điều khiển trước, tiến hành lây lan và sửa đổi sau đó mới trả quyền điều khiển cho chương trình được gọi.

Một số biểu hiện của máy tính bị nhiễm virus:

- Hệ thống hoạt động không ổn định.
- Các chương trình ứng dụng có thể không hoạt động hoặc hoạt động sai chức năng.

- Dữ liệu bị sai lệch.
- Kích thước các file tăng.
- Xuất hiện các file lạ trên đĩa.

...

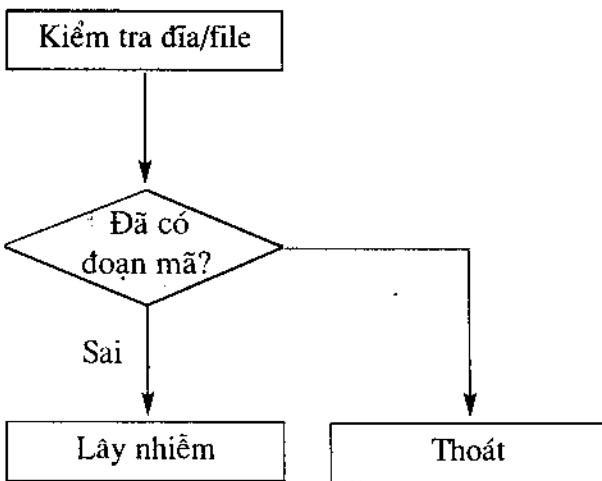
2. Phân loại virus

Dựa vào cơ chế lây lan của virus, người ta có thể phân thành một số loại như sau:

- Boot virus (B - virus): là những virus chỉ lây lan vào các boot sector hoặc master boot record của các ổ đĩa.
- File virus (F - virus): là những virus lây lan vào các file chương trình của người sử dụng (các file COM hoặc EXE).
- Virus lưỡng tính (B/F - virus): là những virus vừa có thể lây lan vào các boot sector hoặc master boot record, vừa có thể lây lan vào các file chương trình.
- Macro virus: là những virus được viết bằng các lệnh macro, chúng thường lây nhiễm vào các file văn bản hoặc bảng tính...
- Troyjan virus (Troyjan Horse): là những virus nằm tiềm ẩn trong hệ thống máy tính dưới dạng các chương trình ứng dụng nhưng trên thực tế, khi chương trình này được kích hoạt, các lệnh phá hoại sẽ hoạt động.
- Worm (sâu): trên thực tế không được coi là virus vì nó không gây tác hại cho phần mềm hoặc phần cứng. Sâu được di chuyển trong hệ thống mạng từ máy tính này sang máy tính khác. Nhiệm vụ chính của chúng là thu thập các thông tin cá nhân của người sử dụng (mật khẩu tài khoản, các thông tin quan trọng, các tài liệu mật...) để chuyển về một địa chỉ xác định cho người điều khiển.

3. Cơ chế hoạt động của virus

Về cơ chế hoạt động của virus, chúng ta có thể hình dung quá trình như sau: Khi đọc một đĩa hoặc thi hành một chương trình bị nhiễm virus, nó sẽ tạo ra một bản sao đoạn mã của mình và nằm thường trú trong bộ nhớ của máy tính. Khi đọc một đĩa hoặc thực hiện một chương trình, đoạn mã virus nằm trong bộ nhớ sẽ kiểm tra đĩa/file đó đã tồn tại đoạn mã chưa? Nếu chưa thì tạo một bản sao khác lây nhiễm vào đĩa/file.



Hình 7.3 - Cơ chế hoạt động của virus

Ví dụ về cơ chế chiếm quyền điều khiển của B-virus: khi máy tính bắt đầu khởi động, mọi thanh ghi CPU sẽ bị xoá. Các thanh ghi đoạn được gán giá trị 0FFFh còn tất cả các thanh ghi còn lại đều được xoá về 0, lúc này cặp CS:IP trỏ đến địa chỉ 0FFFh:0. Tại địa chỉ này, một lệnh JMP FAR chuyển quyền điều khiển đến một đoạn chương trình định sẵn trong ROM BIOS. Đoạn chương trình này sẽ thực hiện quá trình POST (Power On Self Test - Tự kiểm tra khi khởi động).

Quá trình POST sẽ lần lượt kiểm tra các thanh ghi, bộ nhớ, khởi tạo các chíp điều khiển DMA, bộ điều khiển ngắt đĩa... Nếu quá trình này hoàn thành tốt đẹp thì công việc tiếp theo sẽ là dò tìm các card mở rộng (card màn hình, âm thanh,...) và trao quyền điều khiển cho chúng tự khởi tạo sau đó lấy lại quyền điều khiển khi chúng đã hoàn tất việc khởi tạo. Tuy vậy, cần phải chú ý rằng toàn bộ các chương trình này đều nằm trong ROM (bộ nhớ chỉ đọc) của thiết bị nên không thể sửa đổi hoặc chèn thêm mã vào được.

Sau khi mọi việc khởi tạo hoàn thành, lúc này một chương trình trong ROM BIOS đọc boot sector từ đĩa vật lý đầu tiên vào bộ nhớ trong (RAM). Nếu quá trình đọc thành công thì quyền điều khiển sẽ được trao cho đoạn mã nằm trong boot sector bằng một lệnh JMP FAR 0:07C00 mà không cần biết đoạn mã này làm gì. Như vậy tại thời điểm này, bất kể trong boot record chứa đoạn mã nào

thì quyền điều khiển vẫn được trao cho nó, đây là một sơ hở đầu tiên mà máy tính mắc phải. Điều này cũng dễ hiểu, máy tính không thể kiểm tra được đoạn mã trong boot record vì nó ứng với mỗi hệ điều hành cụ thể. Ngay cả khi cùng một hệ điều hành nhưng với version khác nhau thì nội dung đoạn mã này cũng khác nhau.

Lợi dụng khe hở đầu tiên này, B - virus sẽ tấn công vào boot sector, nó thay thế boot sector chuẩn bằng một đoạn mã của mình. Như vậy, quyền điều khiển sẽ được trao cho virus trước khi boot sector chuẩn nhận được quyền điều khiển.

4. Phòng tránh virus

4.1. Các chương trình phòng tránh và phát hiện virus

Coi thường sự tồn tại của virus là con đường trực tiếp dẫn đến sự mất an toàn cho hệ thống nhưng mặt khác, giao phó toàn bộ sự an toàn của hệ thống cho một loạt các chương trình phòng chống virus hiện có thì cũng không ổn. Vì một số chương trình chống virus làm cho người sử dụng cảm thấy an tâm hơn nhưng trong một số trường hợp, các chương trình đó không đưa lại bất kỳ một sự bảo vệ nào.

Điều mà người sử dụng thường làm nhất là trang bị cho mình một phần mềm chống virus nhưng họ cần phải chú ý rằng: virus có trước còn phần mềm chống virus có sau. Như vậy, nếu một virus vừa mới xuất hiện thì nó sẽ “miễn dịch” với tất cả các phần mềm chống virus hiện có.

Các phần mềm chống virus hiện nay được chia thành hai loại: các chương trình phòng ngừa và các chương trình phát hiện.

- Các chương trình phòng ngừa: đây là các chương trình thường trú trong bộ nhớ của máy tính. Chúng hoạt động dựa vào việc giám sát thường xuyên các ngắt để phát hiện và ngăn chặn các yêu cầu được điều khiển bằng phần mềm như: nạp chương trình, ghi thông tin vào đĩa... Ví dụ như một chương trình được nạp và bí mật yêu cầu hệ điều hành cho phép ghi đè lên các boot sector... Khi đó các chương trình phòng ngừa phải báo động ngay và lập tức nhắc nhở người sử dụng về khả năng sẽ bị phá hỏng hệ thống và người sử dụng sẽ phải tự quyết định việc có ngăn chặn hay không.

Về mặt nguyên tắc thì đây là một giải pháp tốt nhưng trên thực tế, nó không

thể cảnh báo một cách chính xác và có hiệu quả cho người sử dụng. Mặt khác, vì thường trú trong bộ nhớ nên nó chiếm mất một phần không gian bộ nhớ, đồng thời thông báo về các ý đồ nạp chương trình; đọc/ghi đĩa là những hoạt động thường xuyên xảy ra trong hệ thống. Dẫn đến người sử dụng không thể phân biệt hết được hoạt động nào là của các chương trình, hoạt động nào là của virus.

Tuy nhiên ở một chừng mực nào đó, các chương trình phòng ngừa vẫn nên được sử dụng nhằm mục đích cung cấp một “lá chắn” cơ bản để ngăn chặn một số loại virus đơn giản.

- Các chương trình phát hiện: trong khi các chương trình phòng chống, ngăn chặn, giám sát các sự kiện xảy ra khi chương trình hoạt động thì các chương trình phát hiện lại tiến hành kiểm tra mã chương trình trước khi nó được thực hiện. Người sử dụng khi được nhắc nhở về mối nguy hiểm khi có mã lạ trong chương trình của mình sẽ phải tự quyết định xem có nên dùng chương trình hay loại bỏ nó. Nếu so sánh một cách trực quan, chúng ta sẽ thấy các chương trình phát hiện có giao diện thân thiện, gần gũi với người sử dụng hơn các chương trình phòng chống.

Các chương trình phát hiện được nạp, thực hiện và thoát ra giống như các chương trình ứng dụng thông thường. Chúng không thường trú trong bộ nhớ, không ngăn chặn và ngắt các chương trình của người sử dụng nhưng nhược điểm của nó là cần phải tuân thủ một số quy tắc nhất định trong quá trình kiểm tra và phát hiện virus.

4.2. Một số biện pháp phòng tránh virus

Chúng ta thấy, virus là một sản phẩm trí tuệ do con người tạo ra nên không thể có các biện pháp phòng chống tuyệt đối. Nguyên tắc chung của phòng chống virus là tạo ra một hàng rào ngăn chặn bất cứ một chương trình phần mềm “lạ” nào muốn thâm nhập và phá hoại hệ thống. Việc sử dụng càng nhiều mức ngăn chặn thì càng đảm bảo an toàn cho hệ thống của người sử dụng hơn. Một số biện pháp phòng tránh có thể đề cập tới như sau:

- Hạn chế trao đổi dữ liệu: con đường chính để virus phát tán được là do người sử dụng thường dùng các phương tiện trao đổi thông tin giữa các máy tính như đĩa mềm, flash disk, CD - ROM... hoặc trao đổi thông tin thông qua hệ

thông mạng. Nhưng nếu không trao đổi thông tin thì chúng ta sẽ không giải quyết được yêu cầu công việc. Do đó, chúng ta cần phải xác định rõ mục đích của việc trao đổi thông tin (có thực sự cần thiết hay không?). Nếu thật sự cần thiết thì phải kiểm tra độ an toàn của thông tin trước khi đưa vào sử dụng.

- Hạn chế sử dụng các phần mềm phá khoá hoặc các phần mềm không rõ nguồn gốc: những phần mềm kiểu này là môi trường mà các lập trình viên tạo virus gửi “sản phẩm” của mình vào để phát tán. Khi chúng ta sử dụng các phần mềm kiểu này, chúng ta sẽ không được đảm bảo an toàn từ phía các công ty sản xuất phần mềm chính thống.

- Sử dụng thường xuyên các phần mềm phòng ngừa và phát hiện virus: phát hiện virus càng sớm bao nhiêu thì các thiệt hại do virus gây ra càng hạn chế. Khi phát hiện được các file bị nhiễm virus, chúng ta cần xoá bỏ hoặc sử dụng chương trình diệt virus để xoá các đoạn mã của chúng.

- Thay đổi các thuộc tính của file: mỗi file thông thường tồn tại bốn thuộc tính bao gồm: chỉ đọc, ẩn, hệ thống và lưu trữ. Chúng ta có thể đặt thuộc tính cho một số file quan trọng nhằm hạn chế tác hại của virus.

- Đổi phần mở rộng của các file chương trình: virus khi lây nhiễm vào các file thường phải biết được cấu trúc của chúng. Với từng kiểu file khác nhau, chúng sẽ có các cơ chế lây nhiễm khác nhau. Nhưng trong một số hệ điều hành, chúng ta có thể đổi các file *.COM thành các file *.EXE mà không gây ra bất cứ hiệu ứng nào. Như vậy, chúng ta có thể đổi phần mở rộng từ COM thành EXE và ngược lại để “đánh lừa” các chương trình virus.

- Cài đặt lại các chương trình ứng dụng: Nếu cảm thấy có nguy hiểm hoặc theo định kỳ, chúng ta nên cài đặt lại các chương trình ứng dụng từ đĩa cài đặt gốc. Nếu có chương trình ứng dụng nào bị nhiễm virus trước khi cài đặt lại thì sau đó chúng sẽ loại bỏ được virus.

- Cài đặt lại hệ thống: là một mức cao hơn của cài đặt lại các chương trình ứng dụng vì lúc đó cả hệ điều hành cũng được thiết lập lại.

- Tạo lại khuôn dạng cho đĩa từ: một số loại virus cắt dấu đoạn mã của mình trong boot sector của đĩa từ và các sector này thường được đánh dấu là “hỏng” hoặc “không được sử dụng” đối với các hệ điều hành. Do đó, cần phải tạo lại

khuôn dạng cho đĩa từ để làm sạch các vùng đĩa này.

- Thường xuyên sao lưu dữ liệu: tốt hơn cả là chúng ta cần phải thường xuyên sao lưu lại dữ liệu để đảm bảo tính an toàn và có thể phục hồi lại khi hệ thống gặp sự cố. Các hệ điều hành đều cung cấp công cụ này với tên chương trình là backup

Câu hỏi ôn tập

1. Trình bày mục đích của bảo vệ hệ thống. Nêu khái niệm về miền bảo vệ, quyền truy nhập và các phương pháp bảo vệ hệ thống.
2. Cho một ví dụ về miền bảo vệ và từ đó cài đặt một ma trận truy nhập.
3. Nêu khái niệm về an toàn hệ thống. Trình bày các cơ chế đảm bảo an toàn hệ thống.
4. Nêu khái niệm, phân loại và cơ chế hoạt động của virus máy tính.
5. Trình bày các biện pháp phòng tránh virus.

Chương 8

HỆ ĐIỀU HÀNH ĐA XỬ LÝ

Mục tiêu

Sau chương này, người học có khả năng phân tích được cấu trúc các hệ thống đa xử lý, biết được mục đích hệ nhiều CPU và hệ phân tán. Đồng thời hình dung sơ lược một số vấn đề về quản lý tài nguyên, truyền thông tin, xử lý và truy nhập thông tin trong các hệ thống này.

Nội dung

Trình bày tổng quan về hệ đa xử lý, bao gồm hệ nhiều CPU và hệ phân tán. Đồng thời nêu rõ khái niệm, mục đích và các cơ chế quản lý tài nguyên cũng như cơ chế quản lý hoạt động của các tiến trình trong hệ thống.

I. TỔNG QUAN VỀ HỆ ĐA XỬ LÝ

Hiện nay, với sự phát triển nhanh của công nghệ, máy tính ngày càng được sử dụng phổ biến trong đời sống xã hội. Mức độ thâm nhập của máy tính vào cuộc sống càng cao thì yêu cầu nâng cao khả năng xử lý của máy tính càng lớn. Bộ nhớ chính ngày càng được mở rộng, dung lượng lưu trữ của đĩa từ ngày càng tăng, tốc độ truy nhập ngày càng cao và hệ thống thiết bị ngoại vi phong phú, hình thức giao tiếp người - máy càng đa dạng. Như chúng ta đã xét, CPU là một tài nguyên rất quan trọng thể hiện khả năng xử lý và tính toán của hệ thống. Vì vậy, một trong những vấn đề quan tâm nhất là tăng cường khả năng xử lý của CPU.

Giải pháp tăng cường khả năng tính toán cho một CPU riêng lẻ đang được ứng dụng một cách triệt để. Tuy nhiên, giải pháp này sẽ phải chịu hạn chế về mặt kỹ thuật như: tốc độ truyền tin không thể vượt quá tốc độ ánh sáng; khoảng cách tối thiểu giữa hai thành phần không thể bằng không...

Song song với giải pháp trên là giải pháp liên kết nhiều CPU lại để tạo ra một hệ thống tích hợp có khả năng xử lý mạnh. Việc đưa ra mô hình xử lý song song tạo nhiều lợi điểm:

- Cho phép chia công việc thành các phần nhỏ và giao cho các CPU đảm nhận. Như vậy, hiệu suất xử lý của hệ thống không chỉ tăng theo tỷ lệ thuận với số CPU mà còn cao hơn do không mất thời gian phải thực hiện các công việc trùng gian.

- Một khác, giải pháp này còn cho phép tích hợp các hệ thống máy tính đã có để tạo ra một hệ thống mới với sức mạnh tăng gấp rất nhiều lần.

Như vậy, với giải pháp nhiều CPU, chúng ta có thể có hai xu hướng tích hợp hệ thống:

- Hệ đa xử lý tập trung - Hệ nhiều CPU: tập hợp các xử lý trong một siêu máy tính (supercomputer). Đặc trưng của hệ thống này là các CPU được liên kết với nhau trong một máy tính duy nhất.

- Hệ đa xử lý phân tán - Hệ phân tán: thực chất là các mạng máy tính, bao gồm các máy tính được liên kết với nhau và đặt tại các vị trí với khoảng cách xa tuỳ ý.

Trong chương này, chúng ta tập trung xét chủ yếu về hai hệ thống trên.

II. HỆ NHIỀU CPU

1. Lý do để xây dựng hệ

Trong giai đoạn hiện nay, các hệ thống nhiều CPU đã được sử dụng rất phổ biến và được người sử dụng chấp nhận bởi ba lý do:

- Hiệu quả của hệ thống tăng khi các CPU được chuyên môn hoá, trước tiên điều này được thể hiện ở việc trang bị cho thiết bị ngoại vi các CPU riêng (ví dụ như card đồ họa có GPU - Graphic Processor Unit). Các thiết bị này có khả năng hoạt động độc lập, giảm nhẹ gánh nặng cho CPU chính.

- Hệ nhiều CPU hoạt động ổn định và có năng suất cao hơn hệ một CPU khi khối lượng công việc nhiều và đa dạng.

- Khi cần đảm bảo độ tin cậy cho hệ thống trong các công việc quan trọng, người ta cần phải làm nhiều bản sao để có thể so sánh kết quả và đánh giá. Như vậy, việc áp dụng hệ nhiều CPU là một giải pháp đáng tin cậy.

2. Cấu hình của hệ nhiều CPU

Trên thực tế, tồn tại nhiều phương pháp kết nối hai hoặc nhiều CPU nhưng chúng cần đảm bảo cho việc sử dụng chung bộ nhớ, chung chương trình và trao đổi dữ liệu với nhau. Cách xác lập mối quan hệ giữa các CPU tạo ra những cấu hình khác nhau của hệ. Có ba cấu hình chính được xem xét:

2.1. Sơ đồ phân cấp

Trong sơ đồ này, một CPU đóng vai trò chủ đạo, các CPU còn lại đóng vai trò thực hiện và được gọi là các CPU ngoại vi. Đây là một sơ đồ kết nối không đối xứng và được coi như là sự mở rộng của hệ thống một CPU. Các CPU ngoại vi có thể hoạt động độc lập trong khi giải quyết công việc của mình.

Sơ đồ này có ưu điểm là dễ tổ chức và chương trình điều khiển không phải sao chép ở nhiều nơi trong bộ nhớ, không phải tổ chức kiểu modul vào nhiều lần nhưng nhược điểm chính là số tín hiệu ngắn sẽ tăng lên nhiều và trách nhiệm xử lý ngắn chủ yếu do CPU chính đảm nhiệm. Mặt khác, độ tin cậy của sơ đồ thấp vì nếu CPU chính gặp sự cố vì một lý do nào đó thì hệ thống sẽ không tiếp tục hoạt động được và các CPU ngoại vi không thể thiết lập được quan hệ với nhau.

2.2. Sơ đồ liên kết mềm

Sơ đồ này có độ tin cậy cao hơn sơ đồ trên, giữa các CPU chỉ có mối quan hệ bán phụ thuộc. Mỗi CPU xử lý tiến trình của mình từ đầu đến cuối và hoạt động như hệ một CPU. Các chiến lược điều khiển tiến trình và quản lý tài nguyên không thay đổi.

Tuy vậy, các CPU vẫn có thể liên hệ, trao đổi thông tin với nhau và chuyển giao tiến trình trước khi nó bắt đầu thực hiện. Cấu hình này có nhược điểm là khó xác định sự cố của CPU và thường bị khởi động lại một cách tự động.

2.3. Sơ đồ liên kết bình đẳng

Trong sơ đồ này, các CPU được coi như tập tài nguyên cùng loại. Thay vì thực hiện chương trình điều khiển riêng biệt trên từng CPU, hệ thống phân chia công việc điều khiển cho tất cả các CPU. Mọi CPU đều phải tham gia điều khiển tiến trình và xử lý ngắn khi có điều kiện. Như vậy, một tiến trình có thể bắt đầu từ processor này và kết thúc ở processor khác.

Thông tin về hàng đợi và các bảng điều khiển được lưu ở bộ nhớ chung. Chương trình điều khiển phải tổ chức ở dạng vào lại được. Như vậy, tải trọng hoạt động của các CPU được phân bố đều.

Ưu điểm của sơ đồ là nếu có một vài CPU gặp sự cố thì chỉ có năng suất xử lý của hệ thống giảm, còn nói chung hệ thống vẫn hoạt động bình thường. Nhưng do chương trình điều khiển và dữ liệu dùng chung, do đó phải có phương

pháp sắp xếp, tổ chức hợp lý. Sơ đồ này có hiệu quả cao khi các CPU thuộc cùng một kiểu.

Chú ý: Dù áp dụng sơ đồ nào đi chăng nữa thì nhiệm vụ của hệ thống cũng là không để người sử dụng nhận thấy tính bất đối xứng của cấu hình cả về mặt phần cứng lẫn phần mềm vì sự bất đối xứng là bản chất của sơ đồ cấu hình hoặc là hệ quả của sự cố kỹ thuật.

Sơ đồ liên kết mềm và liên kết bình đẳng thể hiện hai mức độ phân tán chức năng điều khiển. Các sơ đồ này thể hiện hai ưu điểm lớn là:

- + Năng suất xử lý của hệ thống cao vì không phải tập chung việc xử lý ngắt về CPU chính.

- + Khi có sự cố kỹ thuật, có thể phân phối lại công việc để không ảnh hưởng nhiều tới năng suất chung của toàn hệ thống.

Tuy vậy, nếu chỉ đơn thuần là tách chương trình quản lý thành các phần riêng biệt và cho thực hiện ở nhiều nơi khác nhau thì chưa đảm bảo để hệ thống hoạt động bình thường khi có sự cố kỹ thuật.

3. Quản lý tài nguyên trong chế độ nhiều CPU

3.1. Quản lý bộ nhớ

Việc truy nhập tới bộ nhớ riêng đơn giản hơn vì không có sự cạnh tranh giữa các CPU. Đối với nó, sơ đồ quản lý không có gì đặc biệt.

Bộ nhớ chung khó truy nhập và thời gian truy nhập phụ thuộc mức độ cạnh tranh giữa các CPU. Để giảm mức độ cạnh tranh, hệ thống chia bộ nhớ thành nhiều phần, mỗi CPU chỉ được phép truy nhập tới một số phần nào đó nhưng cần đảm bảo điều kiện giữa hai CPU bao giờ cũng có ít nhất một vùng bộ nhớ chung để trao đổi dữ liệu.

3.2. Quản lý CPU (Lập lịch cho các CPU)

Về mặt lý thuyết thì có thể phân công công việc theo thời gian rỗi của các CPU nhưng như vậy công việc không được phân phối đều và độ phức tạp của các giải thuật lập lịch tăng. Hệ thống thường áp dụng biện pháp chia nhóm các CPU và phân công nhiệm vụ theo nhóm.

3.3. Quản lý tiến trình

Các sơ đồ quản lý tiến trình không phụ thuộc vào hệ thống có bao nhiêu CPU, nhưng trong hệ nhiều CPU sẽ phải tồn tại ngắt CPU để đảm bảo quan hệ giữa các tiến trình.

Các giải pháp quản lý tiến trình của hệ một CPU vẫn được áp dụng nhưng cần phải được biến đổi đôi chút. Ví dụ như phương pháp dùng đèn hiệu thì bản thân đèn hiệu bây giờ cũng là tài nguyên găng. Vì vậy, các tiến trình cần được xếp hàng để chờ xử lý.

3.4. Xử lý lỗi

Hệ thống được đánh giá qua tính sẵn sàng và toàn vẹn. Hệ thống được gọi là toàn vẹn nếu mọi thành phần của nó đều hoạt động bình thường. Sẵn sàng là khả năng đáp ứng mọi yêu cầu xuất hiện khi hệ thống đang hoạt động. Nói chung đảm bảo tính toàn vẹn khó hơn đảm bảo tính sẵn sàng.

Trong vấn đề đảm bảo tính toàn vẹn, nhiệm vụ khó nhất là làm sao biết được tình trạng hoạt động của một CPU (còn tốt hay đã bị sự cố). Để giải quyết vấn đề này, hệ thống tổ chức một vùng bộ nhớ chung, mỗi CPU được gắn với một byte trong vùng nhớ đó. Hệ thống quy định sau một khoảng thời gian nào đó, các CPU phải gửi về byte của mình giá trị 1. Hệ thống dựa vào bản đồ vùng nhớ để xác định được CPU có sự cố (không gửi thông tin về). Sau mỗi lần kiểm tra, hệ thống xoá vùng nhớ để chuẩn bị cho phép kiểm tra tiếp theo.

III. HỆ PHÂN TÁN

1. Khái niệm và mục đích của hệ phân tán (Distributed System)

Hệ phân tán (hay còn gọi là mạng máy tính) là một tập hợp các máy tính ghép nối với nhau bằng đường truyền theo một tiêu chuẩn quy định trước nhằm đạt các mục tiêu:

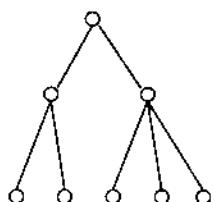
- Tạo khả năng làm việc phân tán.
- Nâng cao chất lượng, hiệu quả của việc khai thác và xử lý dữ liệu.
- Tăng độ tin cậy của hệ thống.
- Chia sẻ tài nguyên (dùng chung tài nguyên, chương trình và dữ liệu).
- Phục vụ công tác truyền tin.

2. Các cấu trúc của hệ phân tán

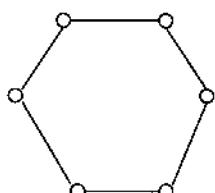
2.1. Cấu trúc kiểu điểm - điểm (Point to Point)

Trong cấu trúc này, đường truyền nối từng cặp nút với nhau, mỗi nút đều có trách nhiệm lưu và chuyển tiếp dữ liệu (store and forward).

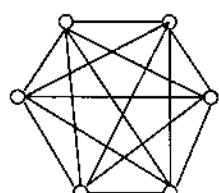
Ưu điểm của cấu trúc này là có độ an toàn và tốc độ cao nhưng chi phí dành cho đường truyền lớn.



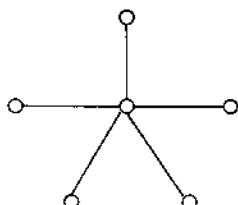
Cây (Tree)



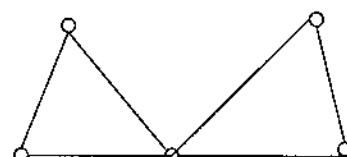
Chu trình (Loop)



Đầy đủ (Complete)



Sao (Star)

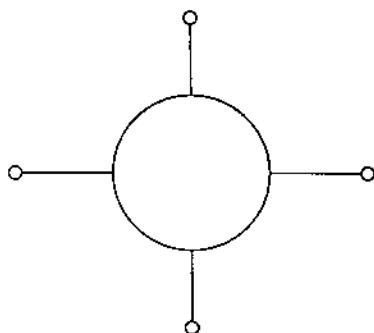


Đồ thị liên thông (Intergraph)

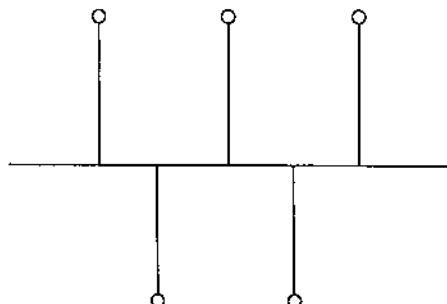
Hình 8.1 - Một số cấu trúc kiểu điểm - điểm

2.2. Cấu trúc kiểu điểm - nhiều điểm (Point to Multipoint)

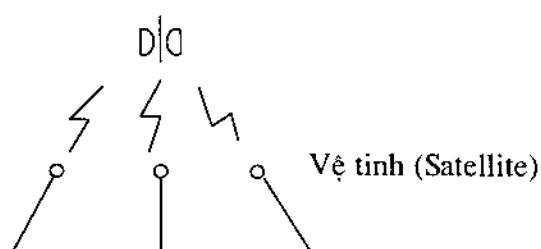
Tất cả các nút mạng đều sử dụng chung một đường truyền. Dữ liệu gửi đi từ một nút và được tiếp nhận ở tất cả các nút còn lại.



Vòng (Ring)



Tuyến tính (Bus)



Vệ tinh (Satellite)

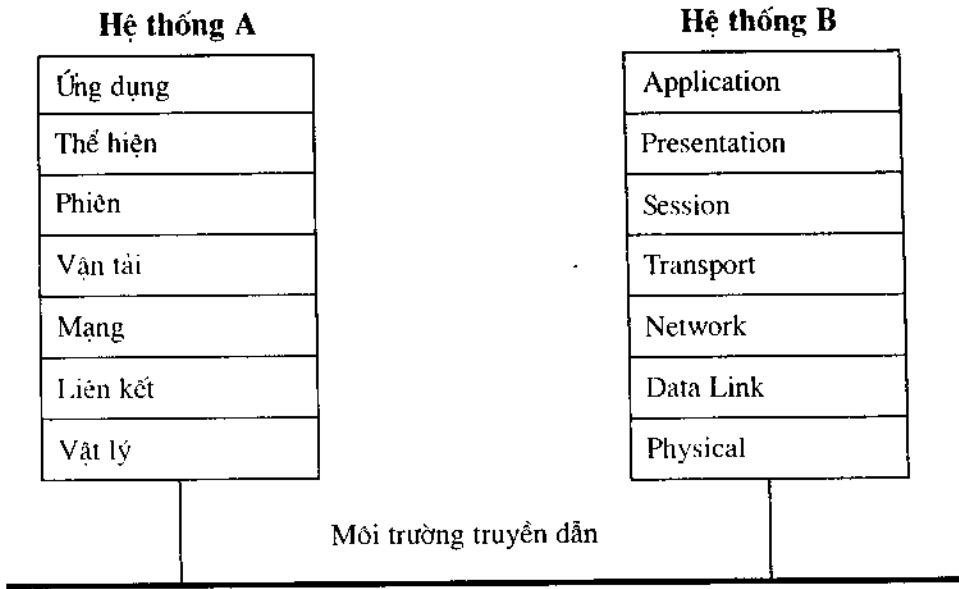
Hình 8.2 - Một số cấu trúc kiểu điểm - nhiều điểm

Cấu trúc này có ưu điểm là tiết kiệm chi phí đường truyền nhưng độ an toàn thấp và cơ chế giải quyết tranh chấp tài nguyên (đường truyền) phức tạp.

3. Thiết kế hệ phân tán

Theo tiêu chuẩn ISO thì hệ phân tán được thiết kế theo kiểu nhiều lớp. Giữa các lớp có giao thức để liên hệ với nhau. Chức năng cụ thể của các lớp trong mô hình OSI như sau:

- Lớp Vật lý (Physical layer): Cung cấp các phương tiện truyền tin, các thủ tục khởi động, duy trì và huỷ bỏ các liên kết vật lý cho phép truyền các dòng dữ liệu ở dạng bits.
- Lớp Liên kết dữ liệu (Data Link layer): Thiết lập, duy trì và huỷ bỏ các liên kết dữ liệu; kiểm soát luồng dữ liệu; phát hiện và khắc phục các sai sót trong quá trình truyền dữ liệu.
- Lớp Mạng (Network layer): Chức năng chính của lớp mạng là chọn đường đi và chuyển tiếp dữ liệu. Ngoài ra, lớp mạng còn có vai trò kiểm soát luồng dữ liệu, khắc phục sai sót, cắt hợp dữ liệu.
- Lớp Giao vận (Transport layer): Kiểm soát luồng dữ liệu từ nơi gửi đến nơi nhận, thực hiện ghép kênh, đóng gói và tổ hợp dữ liệu.
- Lớp Phiên (Session layer): Thiết lập, duy trì, đồng bộ hoá và huỷ bỏ các phiên truyền dữ liệu; cung cấp các thông số điều khiển, quản lý tiến trình truyền dữ liệu.
- Lớp Thể hiện (Presentation layer): Có nhiệm vụ biểu diễn dữ liệu theo cú pháp của người sử dụng, đồng thời cung cấp các phương tiện mã hoá và nén dữ liệu.
- Lớp Ứng dụng (Application layer): Cung cấp các phương tiện để người sử dụng giao tiếp với môi trường OSI và các dịch vụ thông tin phân tán như: dịch vụ truyền file, quản lý file, thư điện tử...



Hình 8.3 - Mô hình OSI

4. Các phương pháp truyền thông trong hệ phân tán

4.1. Gửi thông báo

- Phương pháp cố định (fixed): thông báo được gửi theo những con đường cố định, đặc tả trước và các trạm căn cứ vào đó để nhận thông báo. Phương pháp này đơn giản về cài đặt nhưng nếu thay đổi đặc tả thì sẽ gặp khó khăn.

- Phương pháp ảo (virtual): Mỗi phiên làm việc sẽ khởi tạo một đường truyền để gửi và nhận thông báo. Các phiên làm việc khác nhau sẽ có những đường truyền khác nhau.

- Phương pháp động (dynamic): Khi có thông báo mới khởi tạo đường truyền, sau khi truyền xong thông báo, đường truyền sẽ được huỷ bỏ ngay.

4.2. Tạo mốc nối giữa hai trạm

- Phương pháp chuyển mạch kênh (circuit - switched): Khi có hai nút cần liên lạc với nhau thì một kênh truyền cứng được thiết lập và duy trì cho tới khi một trong hai bên ngắt liên lạc.

- Phương pháp chuyển mạch bản tin (message - switched): Bản tin là một đơn vị thông tin của người sử dụng có khuôn dạng quy định trước. Mỗi bản tin chứa một vùng thông tin điều khiển, trong đó chỉ định rõ địa chỉ đích của bản tin. Các nút mạng căn cứ vào địa chỉ để chuyển bản tin tới nơi nhận.

- Phương pháp chuyển mạch gói (packet - switched): dữ liệu được chia thành các gói độ dài khoảng 256 bytes, phần đầu gói là địa chỉ để tập hợp các gói. Các gói tin được truyền một cách độc lập trên các đường truyền khác nhau.

5. Giải quyết tranh chấp tài nguyên

5.1. Đa truy nhập sử dụng sóng mang (Carrier Sence Multiple Access - CSMA)

Phương pháp CSMA được xây dựng dựa trên nguyên tắc LBT (Listen Before Talk) tức là khi một trạm muốn truyền dữ liệu thì nó cần phải kiểm tra đường truyền. Nếu đường truyền rỗi thì truyền dữ liệu theo khuôn dạng quy định, ngược lại trạm phải thực hiện 1 trong 3 giải thuật:

- Non - persistent: trạm tạm rút lui, chờ một khoảng thời gian ngẫu nhiên sau đó tiếp tục kiểm tra lại đường truyền.
- 1 - persistent: trạm tiếp tục kiểm tra đường truyền cho tới khi rỗi thì truyền dữ liệu với xác suất bằng 1.
- p - persistent: trạm tiếp tục kiểm tra đường truyền cho tới khi rỗi thì truyền dữ liệu với xác suất bằng p ($0 < p < 1$).

5.2. Đa truy nhập sử dụng sóng mang cùng với phát hiện xung đột (Carrier Sence Multiple Access with Detection Collision - CSMA/CD)

Phương pháp CSMA/CD được xây dựng dựa trên nguyên tắc LWT (listen while talk). Tương tự như CSMA nhưng CSMA/CD bổ sung thêm quy tắc: Khi một trạm truyền dữ liệu, nó vẫn tiếp tục kiểm tra đường truyền, nếu phát hiện xung đột thì trạm ngừng ngay việc truyền dữ liệu nhưng vẫn tiếp tục gửi tín hiệu để báo cho tất cả các trạm trên mạng cùng biết sự kiện xung đột đó.

5.3. Token BUS (Bus sử dụng thẻ bài)

Giữa các trạm có nhu cầu truyền dữ liệu được thiết lập một vòng logic. Mỗi trạm trong vòng logic sẽ biết được địa chỉ của trạm đứng trước và sau nó.

Thẻ bài (token) là một đơn vị dữ liệu đặc biệt được thiết lập và lưu chuyển trên vòng logic để cấp phát quyền truy nhập đường truyền cho các trạm.

Khi một trạm nhận được thẻ bài, nó có quyền sử dụng đường truyền trong một khoảng thời gian nhất định để truyền một hoặc nhiều đơn vị dữ liệu. Khi truyền hết dữ liệu hoặc hết thời gian quy định, trạm phải chuyển thẻ bài tới trạm kế tiếp trên vòng.

Các trạm không tham gia vòng logic (không hoặc chưa có nhu cầu truyền dữ liệu) thì chỉ có thể tiếp nhận dữ liệu chứ không có quyền tiếp nhận thẻ bài.

5.4. Token Ring (Vòng sử dụng thẻ bài)

Phương pháp này cũng sử dụng thẻ bài để truy nhập đường truyền nhưng thẻ bài được lưu chuyển trực tiếp trên vòng vật lý. Thẻ bài là một đơn vị dữ liệu đặc biệt trong đó có một bit biểu diễn trạng thái (free/busy).

Một trạm khi muốn truyền dữ liệu cần phải nhận được một thẻ bài ở trạng thái free, khi đó trạm sẽ đổi bit trạng thái thành busy và gửi kèm một đơn vị dữ liệu của mình cùng với thẻ bài theo chiều của vòng.

Khi đến trạm nhận, phần dữ liệu sẽ được sao lại sau đó cùng thẻ bài trở về trạm gửi; trạm gửi sẽ xoá dữ liệu, đổi bít trạng thái của thẻ bài thành free và cho lưu chuyển tiếp trên vòng để các trạm khác có yêu cầu sử dụng.

6. Truy cập thông tin trong hệ phân tán

- Phương pháp ARPANET: Mỗi trạm có một hệ file riêng và trên toàn mạng dùng một chương trình điều khiển truy nhập file là FTP (File Transfer Protocol). Người sử dụng ở trạm A muốn truy nhập file F1 ở trạm B thì FTP sẽ kiểm tra quyền truy nhập của người sử dụng. Nếu hợp lệ thì cho phép truy nhập, ngược lại sẽ thông báo lỗi.

- Phương pháp tiếp cận tập trung: Mỗi trạm có một hệ file riêng còn hệ file chung của toàn mạng nằm ở trạm trung tâm (File Server). Mỗi trạm chỉ có thể truy nhập tới hệ file của mình và hệ file chung của mạng.

- Phương pháp tiếp cận phân tán: Mỗi trạm có một hệ file riêng, tất cả các trạm đều có thể truy nhập tới hệ file của các trạm khác.

7. Xử lý thông tin trong hệ phân tán

- Di chuyển trên mạng: Mục đích chính của hệ phân tán là phục vụ công tác truyền tin. Do đó, việc trao đổi thông tin giữa các trạm được chú trọng một cách đặc biệt. Trên thực tế, tồn tại hai phương pháp di chuyển: di chuyển từng phần, di chuyển toàn bộ. Khi di chuyển thông tin, cần chú ý tới tính tương thích của các kiểu dữ liệu.

- Hỗ trợ khả năng tính toán dữ liệu của các trạm:

+ Phương pháp 1: Giả sử người sử dụng ở trạm A cần truy nhập (hoặc tính toán) dữ liệu ở trạm B. Khi đó, người sử dụng từ trạm A dùng một thủ tục từ xa (Remote Procedure) gọi đến một thủ tục ở trạm B, thủ tục này sẽ xử lý và trả lại kết quả cho người sử dụng ở trạm A.

+ Phương pháp 2: Người sử dụng ở trạm A gửi một thông báo cho B, hệ điều

hành ở B sẽ thực hiện công việc mà thông báo yêu cầu. Sau khi thực hiện xong sẽ trả lại kết quả cho người sử dụng ở trạm A.

- Di chuyển công việc giữa các trạm: Khi có một tiến trình cần thực hiện, cần phải di chuyển tiến trình đến trạm thích hợp sao cho đảm bảo các yêu cầu: thăng bằng tải, tăng tốc độ tính toán, tận dụng sự hỗ trợ của phần cứng và phần mềm. Có thể áp dụng hai phương pháp:

+ Phương pháp 1: Người sử dụng chỉ cần đăng ký công việc với trạm mà mình sẽ chuyển công việc tới, không cần làm phân mảnh của chương trình.

+ Phương pháp 2: Người sử dụng cần đặc tả rõ công việc với trạm mà mình sẽ chuyển công việc tới.

8. Quản lý tiến trình trong hệ phân tán

8.1. Phương pháp tiếp cận tập trung

Hệ thống chọn trong mạng một Processor đặc biệt để đảm nhiệm việc phối hợp các tiến trình khi đi vào đoạn tới hạn (CS - Critical Section). Processor này gọi là Co-ordinator (người điều phối).

Nếu một tiến trình muốn vào đoạn tới hạn thì nó gửi một thông báo yêu cầu (Request Message) tới Co-ordinator. Nếu được phép thì Co-ordinator sẽ gửi một thông báo đồng ý tới tiến trình và khi nhận được thông báo này, tiến trình sẽ đi vào đoạn tới hạn. Khi ra khỏi đoạn tới hạn, tiến trình sẽ gửi cho Co-ordinator một thông báo giải toả.

Về phía Co-ordinator, khi nhận được một thông báo, nó cần kiểm tra xem có tiến trình nào đang trong đoạn tới hạn hay không. Nếu không thì gửi thông báo đồng ý cho tiến trình yêu cầu, ngược lại tiến trình phải xếp vào hàng đợi. Hàng đợi các tiến trình được tổ chức theo kiểu FCFS.

Trường hợp Co-ordinator bị sự cố thì hệ thống sẽ tìm một Co-ordinator mới và Co-ordinator này sẽ thiết lập một hàng đợi mới để tiếp tục điều hành công việc.

8.2. Phương pháp tiếp cận phân tán

- Giải pháp LAMPORT: Hệ thống sắp thứ tự các thông báo yêu cầu vào đoạn tới hạn của các tiến trình và các yêu cầu được phục vụ theo nguyên tắc FCFS.

- Giải pháp RICART – AGRAWALA: Khi tiến trình P muốn đi vào đoạn tới hạn, nó sẽ sinh ra một dấu thời gian (time stamp) và gửi một thông báo yêu cầu có dạng (P, time stamp) tới các tiến trình còn lại. Nếu nó nhận được đầy đủ các thông báo trả lời đồng ý của các tiến trình thì nó được phép vào đoạn tới hạn. Một tiến trình Q khi nhận được một thông báo yêu cầu vào đoạn tới hạn có

thể xảy ra những tình huống sau:

- Nếu Q đang ở trong đoạn tối hạn thì nó trì hoãn việc trả lời cho đến khi ra khỏi đoạn tối hạn.
- Nếu Q không muốn vào đoạn tối hạn thì nó gửi ngay thông báo trả lời đồng ý.
- Nếu Q đang muốn vào đoạn tối hạn thì nó sẽ so sánh time stamp của mình với time stamp của P. Nếu nhỏ hơn, thì Q gửi thông báo đồng ý cho P.

Câu hỏi ôn tập

1. Trình bày mục đích và các cấu hình của hệ nhiều CPU. Nêu rõ ưu điểm và nhược điểm của từng cấu hình.
2. Trình bày phương pháp quản lý bộ nhớ trong hệ nhiều CPU.
3. Nêu khái niệm và mục đích của hệ phân tán.
4. Trình bày các biện pháp giải quyết tranh chấp tài nguyên trong hệ phân tán.
5. Nêu các biện pháp truy nhập thông tin trong hệ phân tán.
6. Trình bày các biện pháp quản lý tiến trình trong hệ phân tán.

Chương 9

HỆ ĐIỀU HÀNH DOS

Mục tiêu

Sau chương này, người học có khả năng vận dụng kiến thức đã học trong các chương trước vào một hệ điều hành cụ thể như: quản lý tiến trình, quản lý bộ nhớ, quản lý đĩa từ, quản lý thiết bị, v.v. để khai thác nó một cách hiệu quả hơn.

Nội dung

Trình bày tổng quan về hệ điều hành DOS bao gồm lịch sử phát triển, cấu trúc và quá trình khởi động DOS. Phương pháp thực hiện chương trình, phương pháp quản lý bộ nhớ, đĩa từ, file, thư mục và thiết bị ngoại vi của DOS.

I. TỔNG QUAN VỀ DOS

1. Lịch sử phát triển của DOS

Năm 1980, các hệ thống máy tính sử dụng bộ vi xử lý 8 bits với hệ điều hành CP/M 80. Tháng 8/1981, hãng IBM chế tạo ra hệ thống máy tính 16 bits (sử dụng bộ vi xử lý 8086). Jim Paterson, một kỹ sư lập trình đã xây dựng hệ điều hành mới dựa trên nền của hệ điều hành CP/M 80 gọi là 86-DOS và sau này là hệ điều hành MS - DOS hay PC - DOS.

Chúng ta có thể điểm qua đặc trưng của một số phiên bản chính sau:

1.1. Version 1 (8/1981)

- Hệ điều hành không phụ thuộc vào phần cứng, phân biệt độ dài dữ liệu vật lý và logic.

- Tên file dài 8 ký tự và 3 ký tự do phần mở rộng, cho phép mở nhiều file đồng thời.

- Cho phép thao tác với các thiết bị như thao tác với các file (mỗi thiết bị có một tên riêng. Ví dụ như: CON - bàn phím và màn hình; PRN - máy in; AUX - giao diện nối tiếp...).

- Xuất hiện loại file chương trình mới có đuôi là EXE. Đặc điểm của nó là có thể nạp vào một vùng nhớ gần như bất kỳ trong bộ nhớ.

- Bộ xử lý lệnh được dồn vào file COMMAND.COM và được chia thành hai phần nội trú và ngoại trú.

- Xuất hiện bảng FAT (file allocation table) để quản lý đĩa từ. Mỗi phân tử của bảng FAT tương ứng với một vùng 512 bytes trên đĩa gọi là một sector. Nó chỉ ra một sector đã bị chiếm hay còn tự do. Bảng FAT còn liên quan đến để mục thư mục trên đĩa.

- Xuất hiện xử lý lô (batch) cho phép người sử dụng nhóm một số lệnh của DOS vào một file văn bản. Khi gọi file này, DOS tự động đọc các lệnh trong file và thực hiện chúng theo thứ tự như là chúng được nhập vào lần lượt từ bàn phím.

- Lưu lại thời gian thay đổi file lần cuối, đây cũng là một vấn đề được người sử dụng rất quan tâm và do vậy, DOS ghi ngày, tháng và giờ file bị thay đổi vào để mục thư mục của file đó.

1.2. Version 2.0 (5/1983)

- Làm việc được với ổ đĩa cứng có dung lượng 10 Mb.

- Quản lý đĩa theo cấu trúc thư mục hình cây.

- Truy nhập file thông qua các hàm của DOS (các thẻ file- File Handle).

Trước đây sử dụng các khối điều khiển file FCB (File Control Block).

- Cho phép điều khiển các thiết bị thông qua các trình điều khiển (driver). Bổ sung trình ANSI. SYS cho phép đặt lại vị trí con trỏ và màn hình thông qua các hàm của DOS.

- Xuất hiện xử lý đa nhiệm cấp thấp, cho phép chạy các chương trình có mức ưu tiên thấp hơn khi các chương trình có mức ưu tiên cao hơn rồi.

- Quản lý đĩa mềm có 9 sector/1 track làm cho dung lượng đĩa tăng từ 160Kb lên 180Kb đối với đĩa một mặt và từ 320Kb lên 360Kb đối với đĩa hai mặt.

1.3. Version 3.0 (4/1984)

- Quản lý đĩa cứng 20 Mb của AT và đĩa mềm 1.2 Mb

- Có nhiều thay đổi trong các hàm của DOS, cho phép thực hiện nhanh nhiều thao tác.

1.4. Version 4.0 (8/1988)

- Cho phép sử dụng hệ thống menu trong các chương trình ứng dụng và tiện ích.

- Hỗ trợ dùng thiết bị chuột (mouse).

- Quản lý được đĩa cứng 2Gb và đảm bảo bộ nhớ mở rộng (EMS) theo chuẩn LIM cho phép lắp đặt tối 8Mb bộ nhớ RAM. Bộ nhớ mở rộng này có thể được dùng cho các chương trình ứng dụng.

2. Cấu trúc của DOS

2.1. Phần hệ thống

Phần hệ thống hay còn gọi là phân lõi của DOS bao gồm các thành phần: DOS-BIOS, hạt nhân của DOS và bộ xử lý lệnh. Mỗi thành phần này được đặt trong một file riêng biệt.

- DOS-BIOS được chứa trong các file IO.SYS (IBMBIO.COM) có thuộc tính ẩn (hidden) và hệ thống (system). Nó gồm các chương trình điều khiển thiết bị chuẩn như bàn phím, màn hình, máy in, đồng hồ, các ổ đĩa... Khi DOS muốn liên lạc với thiết bị nào thì nó gọi chương trình điều khiển thiết bị trong IO.SYS. Chương trình điều khiển này lại gọi các chương trình con trong ROM BIOS. Do vậy, DOS - BIOS là thành phần liên hệ chặt chẽ với phần cứng và thay đổi theo model của máy tính.

- Hạt nhân của DOS nằm trong file MSDOS.SYS (IBM.COM), nó gồm các chương trình con để truy nhập file, vào/ra ký tự và sử dụng RAM. Các chương trình con này được xây dựng không phụ thuộc vào phần cứng của máy tính mà nó dựa vào các chương trình điều khiển thiết bị của DOS-BIOS để truy nhập các thiết bị như bàn phím, màn hình và các ổ đĩa. Do vậy, hạt nhân DOS không cần thích nghi với phần cứng của các máy tính khác nhau. Các chương trình của người sử dụng có thể truy nhập các chương trình con trong hạt nhân DOS như là truy nhập các hàm của BIOS trong ROM. Mỗi chương trình con có thể được gọi bằng một ngắt mềm có dùng các thanh ghi của CPU để truyền các tham số cần thiết.

- Bộ xử lý lệnh nằm trong file COMMAND.COM. Nó nhận các lệnh của người sử dụng và thực hiện chúng. Bộ xử lý lệnh được chia thành ba phần:

+ Phần thường trú: nằm thường trực trong bộ nhớ gồm các chương trình con khác nhau gọi là các bộ nhớ điều khiển ngắt, chúng xử lý các sự kiện hoặc một số lỗi phát sinh khi liên lạc với thiết bị .

+ Phần tạm trú: là phần đưa ra dấu nhắc lệnh (A:> hoặc C:>). Bộ nhớ mà phần tạm trú chiếm không được bảo vệ, do đó nó có thể bị xóa bởi các chương trình ứng dụng. Sau khi chương trình ứng dụng kết thúc, quyền điều khiển được trả về cho phần thường trú và phần thường trú nạp lại phần tạm trú từ đĩa.

+ Phần cài đặt: được nạp khi khởi động hệ điều hành và chức năng của nó là khởi động hệ thống. Sau khi hoạt động xong, phần bộ nhớ mà nó chiếm sẽ được giải phóng.

2.2. Phần trợ giúp

Phần trợ giúp còn được gọi là phần ứng dụng, có chức năng dùng để thực hiện các lệnh ngoại trú của DOS.

Phần trợ giúp là một tập các chương trình được chứa trên đĩa và chúng được gọi vào bộ nhớ để thực hiện khi có lệnh gọi, sau khi thực hiện xong chúng được giải phóng khỏi bộ nhớ.

3. Khởi động DOS

Quá trình khởi động DOS được tuân thủ theo các bước sau:

- Kiểm tra xem có đĩa trong ổ mềm hay không. Nếu có đĩa mềm được xem là đĩa khởi động, ngược lại kiểm tra trong máy có đĩa cứng không để khởi động từ đĩa cứng. Nếu cùng không có thì xuất hiện một thông báo lỗi.

- Nếu tồn tại đĩa mềm (hoặc đĩa cứng), boot sector (sector khởi động) được nạp vào bộ nhớ và chương trình con khởi động trong boot sector được thực hiện.

- Chương trình con khởi động trong boot sector kiểm tra hai file đầu tiên trên đĩa có phải là IO.SYS và MSDOS hay không. Nếu không phải, thông báo lỗi đây không phải là đĩa khởi động DOS. Ngược lại, nạp IO.SYS vào trong bộ nhớ và trao quyền điều khiển cho một chương trình con trong IO.SYS.

- Chương trình con trong IO.SYS nạp MSDOS.SYS vào bộ nhớ và trao quyền điều khiển cho một chương trình con trong MSDOS.SYS.

- Chương trình con trong MSDOS.SYS tạo ra một số bảng và vùng dữ liệu, đồng thời khởi tạo các chương trình điều khiển thiết bị.

- Tìm file CONFIG.SYS ở trên đĩa, nếu có thì nạp CONFIG.SYS và cấu hình DOS theo chỉ dẫn của file này (bao gồm thiết lập các buffer, khởi điều khiển file

- FCB, nạp các chương trình điều khiển các thiết bị...).

- Nạp bộ xử lý lệnh COMMAND.COM và thông qua nó để kiểm soát các sự kiện. Giải phóng chương trình con khởi động khỏi bộ nhớ.

II. PHƯƠNG PHÁP THỰC HIỆN CHƯƠNG TRÌNH CỦA DOS

1. Phương pháp nạp chương trình và cấu trúc PSP (Program Segment Prefix)

DOS quản lý hai loại file chương trình là COM và EXE. Các chương trình này được nạp và khởi động thông qua hàm EXEC của DOS. Trước khi nạp chương trình vào bộ nhớ, hàm EXEC thông qua một số hàm khác của DOS, chuẩn bị vùng nhớ RAM mà chương trình được gọi sẽ chiếm. Hàm EXEC đặt vào đầu vùng nhớ một cấu trúc dữ liệu gọi là PSP (Program Segment Prefix -

tiên tố chương trình). Chương trình sẽ được nạp vào sau cấu trúc dữ liệu này, các thanh ghi đoạn và ngăn xếp được khởi đầu và chương trình được khởi động xong. Sau khi thực hiện chương trình, vùng nhớ mà chương trình và PSP chiếm được giải phóng.

Địa chỉ	Nội dung	Kiểu
00h	Lệnh INT 20	2 bytes
02h	Địa chỉ đoạn của ô nhớ cuối cùng dành cho chương trình	1 word
04h	Không công bố	1 byte
05h	FAR CALL tới chương trình điều phối hàm của DOS	5 bytes
0Ah	Bản sao vector ngắn 22h	1 point
0Eh	Bản sao vector ngắn 23h	1 point
12h	Bản sao vector ngắn 24h	1 point
16h	Không công bố	22 bytes
2Ch	Địa chỉ đoạn của khối biến môi trường	1 word
2Eh	Không công bố	46 bytes
5Ch	FCB #1	16 bytes
6Ch	FCB #2	16 bytes
80h	Số lượng ký tự trong dòng lệnh	1 byte
81h	Dòng lệnh	127 bytes
Tổng cộng		256 bytes

Bảng 9.1 - Cấu trúc PSP

PSP có độ dài 256 bytes và chứa nhiều thông tin quan trọng cho DOS và chương trình thực hiện. Ý nghĩa các trường trong PSP như sau:

- Tại ô nhớ 00h là lời gọi hàm của DOS dùng để kết thúc chương trình, giải phóng bộ nhớ và trả quyền điều kiện về cho bộ xử lý lệnh.
- Tại ô nhớ 02h là địa chỉ đoạn của ô nhớ cuối cùng dành cho chương trình.

Nếu chương trình cần thêm bộ nhớ, nó có thể thông qua địa chỉ này để yêu cầu thêm vùng nhớ.

- Tại ô nhớ 05h là lệnh FAR CALL gọi tới chương trình điều phối các hàm của DOS (có thể gọi các hàm từ 0h - 24h).

- Tại các ô nhớ 0Ah, 0Eh, 12h chứa nội dung 3 vector ngắn để kết thúc chương trình, phản ứng khi các phím Ctrl - C hoặc Ctrl - Break được bấm và phản ứng với các lỗi.

- Tại ô nhớ 2Ch lưu trữ địa chỉ đoạn của khối biến môi trường. Khối biến môi trường là một loạt các chuỗi kí tự ASCII chứa thông tin như: đường dẫn tìm kiếm (lệnh PATH), thư mục chứa bộ xử lý lệnh.

- Tại ô nhớ 5Ch, 6Ch chứa hai khối điều khiển file (File Control Block - FCB) là hai tham số đầu tiên trong dòng lệnh liên quan đến tên file.

- Tại ô nhớ 80h ghi số lượng ký tự trong dòng lệnh và có thể được DOS sử dụng làm vùng chuyển dữ liệu đĩa.

- Từ ô nhớ 81h đến OFFh chứa các tham số và dòng lệnh.

2. Chương trình COM và EXE

2.1. Chương trình COM

Các file COM được lưu trên đĩa là ảnh chính xác của bộ nhớ khi các file này được nạp. Sau khi được nạp, các file COM không cần thêm bất kỳ thông tin phụ nào và có thể thực hiện được ngay. Do vậy, chương trình COM có thể nạp và khởi động nhanh hơn chương trình EXE.

Khi thực hiện, chương trình COM được nạp vào bộ nhớ từ ô nhớ sát với ô nhớ cuối cùng của PSP, nghĩa là từ địa chỉ offset 100h. Ô nhớ này thường chứa một lệnh nhảy đến phần đầu thực sự của chương trình.

Kích thước chương trình COM dài không quá 64Kb (65536 bytes) tính cả độ dài của PSP (256 bytes) và ít nhất 1 từ (2 bytes) cho ngăn xếp. Mặc dù vậy, khi thực hiện, DOS vẫn dành cả bộ nhớ cho chương trình này và chương trình COM không thể gọi một chương trình khác thông qua hàm EXEC.

Khi điều khiển được trao cho chương trình COM, các thanh ghi đoạn đều hướng tới đầu của PSP. Đầu của chương trình COM (so với đầu của PSP) luôn được đặt ở địa chỉ 100h. Con trỏ ngăn xếp nhận giá trị FFFEh và hướng tới cuối đoạn 64Kb mà chương trình COM chiếm, ngăn xếp tiến lại gần cuối chương trình cứ 2 bytes một. Người lập trình cần phải đảm bảo cho ngăn xếp không lớn đến mức dụng vào phần cuối chương trình.

Để kết thúc chương trình COM và trả quyền điều khiển về cho DOS, tồn tại các khả năng:

- Gọi hàm 00h của ngắt 21h.
- Gọi ngắt 20h.
- Gọi hàm 40h của ngắt 21h.

Một số nhược điểm của chương trình COM:

- Khi gọi một chương trình COM, DOS dành toàn bộ nhớ cho chương trình, do đó nếu một chương trình COM là thường trú thì DOS không thể nạp tiếp chương trình khác.

- Chương trình COM trong chương trình thực hiện không thể gọi một chương trình khác thông qua hàm EXEC.

Để giải quyết hai nhược điểm này cần phải giải phóng vùng nhớ mà chương trình không dùng tới.

2.2. Chương trình EXE

So với chương trình COM, chương trình EXE không bị hạn chế trong đoạn 64Kb dành cho cả mã, dữ liệu và ngăn xếp. Cái giá phải trả cho ưu điểm này là các file EXE phức tạp lên do sự xuất hiện trong file một loạt thông tin không phải là của chương trình. Đối với các thế hệ sau cùng của DOS, các chương trình EXE còn thể hiện một ưu điểm nữa là dễ thích ứng với các đổi mới của DOS, ví dụ như khả năng làm việc đa nhiệm.

Chương trình EXE chứa các đoạn phân biệt cho mã, số liệu và ngăn xếp. Các đoạn này được sắp xếp theo một thứ tự bất kỳ. Khác với chương trình COM, chương trình EXE không thể nạp trực tiếp từ bộ nhớ ngoài vào bộ nhớ trong để thực hiện ngay lập tức mà trước khi thực hiện, nó cần được sự chuẩn bị bởi một chương trình con trong hàm EXEC của DOS. Sự chuẩn bị này là cần thiết để giải quyết một số vấn đề đã được đề cập tới khi mô tả chương trình COM.

Chương trình EXE không cần nạp vào một vị trí xác định trước mà có thể nạp vào một vị trí bất kỳ trong bộ nhớ (là bội nguyên của 16). Vì chương trình EXE có thể chứa nhiều đoạn nên việc sử dụng lệnh FAR (ngôn ngữ máy) là cần thiết nếu chương trình muốn từ một đoạn, gọi một chương trình con ở đoạn khác. Một lệnh FAR không những phải chỉ ra địa chỉ offset của đoạn mà còn phải chỉ ra địa chỉ của cả đoạn. Do vậy, từ đây nảy sinh ra vấn đề là địa chỉ đoạn này có thể thay đổi trong mỗi lần gọi thực hiện chương trình.

Trong chương trình COM, vấn đề này được giải quyết một cách đơn giản

nhưng nghiêm ngặt là kích thước chương trình không vượt quá 64Kb và không được phép dùng lệnh FAR trong chương trình, còn các chương trình EXE giải quyết vấn đề một cách phức tạp nhưng hiệu quả hơn bằng cách sử dụng một cấu trúc dữ liệu gọi là khối đầu của chương trình EXE. Cấu trúc dữ liệu này ngoài các thông tin khác còn chứa địa chỉ tương đối của các đoạn. Địa chỉ đoạn thực sự trong bộ nhớ được tính bằng cách cộng địa chỉ tương đối này với địa chỉ của đoạn mà chương trình được nạp vào đó (gọi tắt là địa chỉ bắt đầu đoạn, thường là địa chỉ đoạn của PSP + 10h).

Địa chỉ	Nội dung	Kiểu
00h	Đánh dấu đây là một chương trình EXE (5A 4Dh)	1 word
02h	(Độ dài của file) MOD 512	1 word
04h	(Độ dài của file) DIV 512	1 word
06h	Số lượng địa chỉ đoạn cần được sửa lại cho phù hợp	1 word
08h	Kích thước của header (đơn vị paragraph = 16 bytes)	1 word
0Ah	Số lượng nhỏ nhất các paragraph cần bổ sung	1 word
0Ch	Số lượng lớn nhất các paragraph cần bổ sung	1 word
0Eh	Địa chỉ đoạn tương đối của ngăn xếp	1 word
10h	Nội dung thanh ghi SP lúc khởi động chương trình	1 word
12h	Kiểm tra chẵn lẻ phần tiêu đề của file	1 word
14h	Nội dung thanh ghi IP lúc khởi động chương trình	1 word
16h	Địa chỉ bắt đầu của đoạn mã trong file EXE	1 word
18h	Địa chỉ của bảng định lại giá trị	1 word
1Ah	Số Overlay	1 word
1Ch	Bộ nhớ đệm	thay đổi
??	Bảng chứa các địa chỉ cần định lại giá trị biến	thay đổi
??	Mã chương trình, các đoạn số liệu và ngăn xếp biến	thay đổi

Bảng 9.2 - Cấu trúc khối đầu của chương trình EXE

Khi hàm EXEC nạp một chương trình EXE, nó biết được địa chỉ các ô nhớ chứa các địa chỉ đoạn cần phải sửa đổi lại cho phù hợp. Nó viết lại các giá trị này bằng cách cộng các giá trị đó với địa chỉ bắt đầu đoạn (PSP + 10h). Thao tác này làm cho thời gian khởi động và thực hiện một chương trình EXE lâu hơn chương trình COM, đồng thời kích thước chương trình EXE cũng lớn hơn kích thước một file COM tương đương. Bất lợi này là không đáng kể so với ưu điểm có thể xây dựng được một chương trình lớn hơn kích thước 64Kb.

Sau khi các địa chỉ đoạn được sửa lại thành các địa chỉ có hiệu lực. Hàm EXEC cố định các thanh ghi đoạn DS, ES theo đầu của PSP (DS = ES = PSP), do đó chương trình EXE có thể truy nhập dễ dàng đến các thông tin trong PSP như: địa chỉ khối biến môi trường và tham số các dòng lệnh. Địa chỉ của ngăn xếp và nội dung con trỏ ngăn xếp được lưu ở khối đầu của file EXE, từ đó nó được lấy lại để nạp vào bộ nhớ. Chú ý là địa chỉ đoạn của ngăn xếp SS phải được sửa lại, điều đó cũng đúng đối với địa chỉ đoạn mã và nội dung con trỏ lệnh. Sau khi chúng nhận được các giá trị xác định, chương trình mới bắt đầu thực hiện.

Để đảm bảo tính tương thích với các version sau của DOS nên chương trình EXE thường được kết thúc bằng hàm 4Ch của ngắt 21h.

Đối với chương trình EXE, chúng ta cũng phải dành bộ nhớ cho nó khi nạp chương trình nhưng điều này không giống với chương trình COM. Chương trình nạp EXE có thể nhận được từ file EXE kích thước các đoạn chương trình và do vậy cùng lúc cả kích thước tổng thể của chương trình trong bộ nhớ. Sau đó nó có thể yêu cầu bộ nhớ cần thiết thông qua một hàm khác. Bộ nhớ được dành ra đương nhiên phải lớn hơn kích thước của chương trình vì còn có bộ nhớ bổ sung cần thiết cho quá trình thực hiện của chương trình. Kích thước của bộ nhớ bổ sung này được xác định theo nội dung của hai trường trong phần đầu của file EXE, chúng chỉ ra kích thước nhỏ nhất và lớn nhất của vùng nhớ bổ sung tính theo đơn vị paragraph.

Cách xác định được thực hiện như sau: đầu tiên chương trình nạp EXE thử dành cho chương trình số lượng paragraph lớn nhất. Nếu không thể được, nó phải chấp nhận với phần bộ nhớ còn lại với điều kiện là phần nhớ này vẫn phải lớn hơn số lượng nhỏ nhất các paragraph. Hai trường trong địa chỉ đầu của file EXE được xác định không phải bởi chương trình LINK mà bởi chương trình COMPILER hay ASSEMBLER. Chương trình này cho số lớn nhất là FFFFh

(giá trị này tương đương $FFFF * 16 = FFFFFh = 1Mb$) và trên thực tế chỉ là tất cả bộ nhớ còn được tự do dành cho chương trình EXE và bây giờ chúng ta lại gặp phải các vấn đề như đối với chương trình COM.

Tuy nhiên, các chương trình EXE có cấu trúc không phù hợp để làm chương trình nội trú nhưng cấu trúc này lại phù hợp khi một chương trình nào đó có nhu cầu gọi một chương trình khác trong khi thực hiện. Điều này cũng như chương trình COM, chỉ thực hiện được nếu vùng nhớ không dùng đến được giải phóng.

3. Hàm EXEC - Nạp và thực hiện chương trình

Như trên chúng ta đã nói tới hàm EXEC của DOS. Nó có chức năng dùng để nạp và thực hiện chương trình. Hàm EXEC có thể gọi thông qua ngắt 21h của DOS (số hàm là 4Bh).

Hàm EXEC cho phép một chương trình mẹ gọi một chương trình con. Chương trình con sẽ được nạp vào bộ nhớ RAM từ bộ nhớ ngoài và sau đó được thực hiện. Nếu chương trình này không được cài đặt nội trú thì bộ nhớ mà nó chiếm sẽ được giải phóng sau khi chương trình này được thực hiện xong. Đến lượt mình, chương trình con lại có thể gọi một chương trình con khác. Như vậy, ta có thể tạo ra một chuỗi các chương trình mà độ dài của nó chỉ bị hạn chế bởi kích thước của bộ nhớ RAM còn tự do.

Một ví dụ điển hình về việc sử dụng hàm EXEC là bộ xử lý lệnh. Khi bộ xử lý lệnh thực hiện các chương trình của người sử dụng thì nó được coi là chương trình mẹ. Một số chương trình ứng dụng lại cho phép người sử dụng gọi thực hiện các lệnh của DOS (điều này cũng được thực hiện bởi hàm 4Bh của ngắt 21h).

Chương trình mẹ cũng có thể truyền một số tham số cho chương trình con. Điều này có thể được thực hiện bằng cách truyền tham số trong dòng lệnh hoặc thông qua khối biến môi trường. Ngoài ra, cũng có thể truyền cho chương trình con các tham số trong PSP. Trên thực tế, các chương trình con cũng có PSP của mình nên có thể viết thông tin vào hai FCB trong PSP này và như vậy, các thông tin này là truy nhập được đối với chương trình con.

Sau khi điều khiển được chuyển cho chương trình con, nó có thể truy nhập tới tất cả các file và thiết bị được mở bởi chương trình mẹ. Như vậy, một chương trình con có thể đọc/ghi thông tin vào một file mà nó không cần biết tên file này, miễn là nó biết được thẻ của file (thẻ file được truyền cho chương trình con

từ trước bởi chương trình mẹ bằng một trong ba cách đã nêu). Tất nhiên, con trỏ file sẽ bị thay đổi bởi sự truy nhập của các chương trình con. Giá trị con trỏ file mà chương trình con đã thay đổi không được khôi phục khi điều khiển được trả về cho chương trình mẹ và như vậy sự truy nhập file của chương trình con trở thành “nhìn thấy được” đối với chương trình mẹ.

Sau khi chương trình con kết thúc, điều khiển được trả về cho chương trình mẹ và chương trình mẹ lại tiếp tục được thực hiện và lúc kết thúc, chương trình con có thể truyền lại cho chương trình mẹ một số giá trị. Điều này có thể thực hiện bằng hàm 4Ch, hàm này cho phép kết thúc các chương trình đồng thời truyền một mã cho chương trình mẹ.

Tất nhiên, sự liên lạc giữa chương trình mẹ và chương trình con chỉ thực hiện được nếu cả hai cùng sử dụng chung một ngôn ngữ. Sau khi quyền điều khiển được trả cho chương trình mẹ, chương trình mẹ có thể kiểm tra mã được truyền bằng hàm 4Dh của ngắt 21h.

Chú ý: như chúng ta đã biết, hàm EXEC chỉ nạp chương trình con nếu bộ nhớ RAM tự do còn đủ lớn. Đối với các chương trình EXE, DOS có thể xác định được bộ nhớ cần thiết dành cho chúng nhưng với các chương trình COM, điều này là không thể cho nên DOS dành toàn bộ vùng nhớ RAM tự do cho các chương trình COM. Kết quả là chương trình COM không thể gọi một chương trình khác thông qua hàm EXEC của DOS vì không còn bộ nhớ RAM tự do.

III. QUẢN LÝ BỘ NHỚ RAM CỦA DOS

1. Phương pháp quản lý RAM theo MCB (Memory Control Block)

Dưới quan điểm của DOS, bộ nhớ quy ước của máy tính được chia thành hai vùng (một cách logic). DOS sử dụng vùng thứ nhất và do vậy chúng ta gọi đó là vùng hệ điều hành.

Vùng 1: Bắt đầu từ ô nhớ thấp nhất (0000:0000), vùng này DOS sử dụng để chứa các vector ngắt cũng như các bảng số liệu, các vùng đệm, các biến trong và phần mã phần nội trú của DOS. Ngoài ra, còn có các chương trình điều khiển thiết bị được ghép vào hệ thống và chúng có thể được gọi như một hàm của DOS. Kích thước của vùng này phụ thuộc vào thế hệ của DOS, kích thước của các chương trình điều khiển thiết bị được cài đặt và một số các yếu tố khác như số lượng các vùng đệm...

Vùng 2: Được gọi là Transient Program Area - TPA (vùng chương trình tạm thời) có nghĩa là vùng các chương trình ứng dụng. Vùng này bắt đầu ngay sau vùng hệ điều hành, nó chứa các chương trình cần chạy cũng như các khối môi trường tương ứng.

Tuỳ theo yêu cầu bộ nhớ của các chương trình, DOS cung cấp cho mỗi chương trình một vùng nhớ. Vùng nhớ này được quản lý bởi một khối dữ liệu gọi là Memory Control Block - MCB có kích thước 16 byte (=1 paragraph), đứng ngay trước vùng nhớ được cấp phát cho chương trình. Trong các hàm quản lý bộ nhớ, DOS luôn làm việc với địa chỉ đoạn của vùng nhớ được cấp phát cho chương trình nhưng địa chỉ đoạn của MCB có thể dễ dàng tính được bằng cách lấy địa chỉ đoạn của vùng nhớ trừ đi 1.

Địa chỉ	Nội dung	Kiểu
00h	ID	1 byte
01h	Địa chỉ đoạn của PSP tương ứng	1 word
03h	Số lượng các paragraph trong vùng nhớ được cấp	1 word
05h	Không sử dụng	11 bytes
10h	Vùng nhớ được cấp	
Tổng cộng		16 bytes

Bảng 9.3 - Cấu trúc một MCB

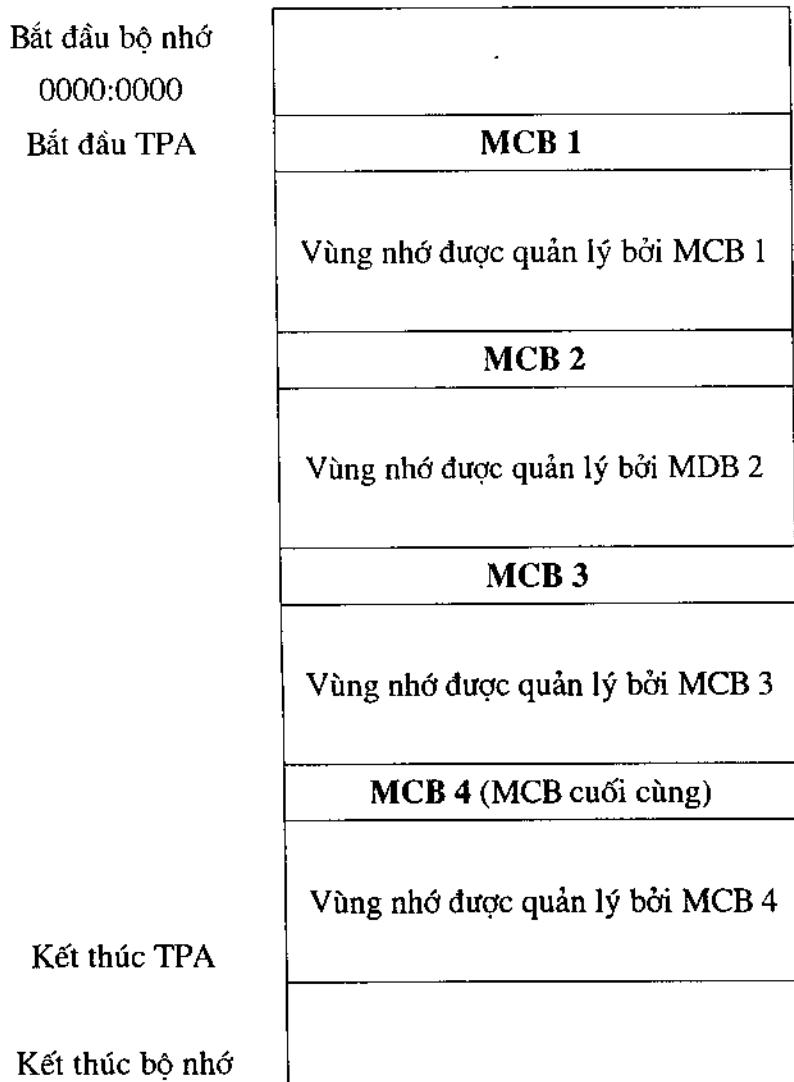
Ý nghĩa các trường trong MCB:

- Trường thứ nhất chứa một trong hai ký tự là “M” hoặc “Z”. Nếu ID là “M” thì sau MCB này còn có các MCB khác, nếu là “Z” thì đây là MCB cuối cùng trong bộ nhớ.

- Trường thứ hai chứa địa chỉ đoạn PSP của chương trình. Địa chỉ này chỉ có nghĩa khi vùng nhớ được cấp phát là khối môi trường của một chương trình, trường này chỉ ra địa chỉ PSP của chính chương trình đó và như vậy tạo ra mối liên hệ. Ngược lại, nếu vùng nhớ là một PSP thì trong đại đa số các trường hợp, trường này chỉ ra chính vùng nhớ của chương trình.

- Trường thứ ba chỉ ra kích thước của vùng nhớ được cấp phát theo đơn vị

paragraph. Thực vậy, vì MCB tiếp theo bắt đầu ngay sau vùng nhớ được cấp phát (nếu trường một không chứa ký tự Z), nên trường này cũng chỉ ra khoảng cách đến MCB tiếp theo. Mỗi một MCB chỉ ra MCB tiếp theo một cách gián tiếp, do vậy ta nhận thấy được một chuỗi cho phép tìm lại danh sách tất cả các MCB.



Hình 9.1 - Quản lý bộ nhớ bằng các MCB

2. Các hàm quản lý bộ nhớ của DOS

2.1. Cấp phát bộ nhớ

Để cấp phát bộ nhớ cho các chương trình, DOS sử dụng hàm 48h: nhận số hàm trong thanh ghi AH, dung lượng bộ nhớ cần cấp phát (tính theo paragraph) trong thanh ghi BX. Nếu số lượng paragraph yêu cầu được thỏa mãn thì hàm trả lại cờ Carry bằng 0 và thanh ghi AX chứa địa chỉ đoạn của bộ nhớ được cấp phát. Do vậy, nó được bắt đầu từ địa chỉ AX:0000 (MCB tương ứng nằm ở paragraph trước đó). Ngược lại, nếu không đủ số lượng paragraph yêu cầu, cờ Carry bằng 1, thanh ghi AX sẽ chứa mã lỗi và thanh ghi BX chứa kích thước bộ nhớ còn lại theo đơn vị tính paragraph.

2.2. Xác định kích thước bộ nhớ tự do

Trên thực tế, DOS không cung cấp hàm riêng để xác định không gian bộ nhớ tự do nhưng kết quả có thể nhận được từ hàm 48h theo phương pháp sau: nạp vào thanh ghi BX giá trị 0FFFFh để yêu cầu cấp bộ nhớ kích thước 1 Mb (tất nhiên là DOS không thể cấp đủ), khi đó thanh ghi BX sẽ trả lại kích thước bộ nhớ tự do.

2.3. Giải phóng bộ nhớ

Để giải phóng bộ nhớ mà hàm 48h đã cấp, DOS sử dụng hàm 49h. Địa chỉ đoạn của vùng nhớ cần giải phóng đặt trong thanh ghi ES. Hàm này sẽ sinh lỗi nếu MCB đã bị xoá bởi chương trình hoặc nếu địa chỉ đoạn trong thanh ghi ES không thuộc về một vùng nhớ đã được cấp nào.

2.4. Thay đổi kích thước vùng nhớ

Để thay đổi kích thước vùng nhớ đã cấp phát, DOS sử dụng hàm 41h. Thanh ghi ES nhận địa chỉ đoạn của vùng nhớ cần thay đổi kích thước, thanh ghi BX nhận số lượng các paragraph tương ứng với kích thước mới của vùng nhớ. Định nghĩa các thanh ghi sau khi gọi hàm này giống như hàm 48h.

IV. QUẢN LÝ ĐĨA TỪ CỦA DOS

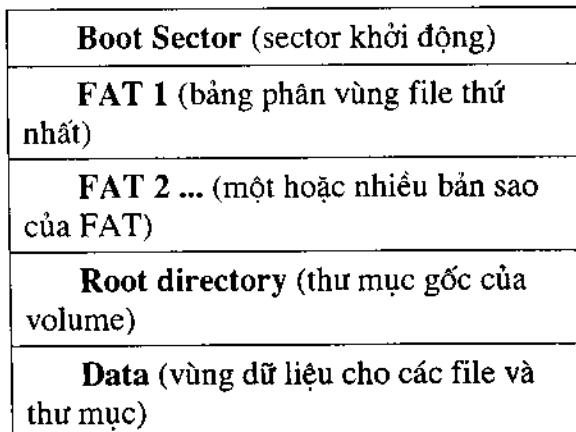
1. Cấu trúc logic của đĩa từ

Dưới con mắt của người sử dụng, DOS gọi các đĩa là các volume, mỗi volume được gán một chữ cái (ổ đĩa mềm A, B; ổ đĩa cứng C...). Mỗi ổ đĩa vật lý có thể chứa nhiều volume (tất nhiên việc phân chia đĩa mềm thành nhiều volume không có ý nghĩa vì kích thước của chúng quá nhỏ). Mỗi volume có thể có một tên gọi nhưng điều này không bắt buộc. Tên volume được thể hiện trong

lệnh DIR của DOS.

Mỗi volume có một thư mục gốc, trong thư mục này chứa các thư mục con và các file. Để truy nhập tới các file và các thư mục, người sử dụng dùng các hàm của DOS.

Mỗi volume được DOS chia thành các sector, mỗi sector chứa một số xác định các byte (về nguyên tắc là 512 bytes), chúng được đặt nối tiếp nhau. Các sector được đánh số liên tục, bắt đầu từ 0. Tuy nhiên, các lời gọi hàm không bao giờ liên quan đến các sector mà liên quan đến file, do vậy DOS cần phải chuyển đổi các lời gọi file thành lời gọi sector. Để làm việc này, nó sử dụng các thư mục và một cấu trúc dữ liệu gọi là FAT (File Allocation Table). Hai loại thông tin này được chứa trên đĩa. Sau khi các sector logic được xác định, điều khiển được trao cho driver thiết bị, nó chuyển các số sector logic thành các địa chỉ vật lý (liên quan đến số hiệu sector vật lý, số hiệu track, số hiệu đầu từ đọc/ghi).



Hình 9.2 - Cấu trúc của một volume

2. Sector khởi động (Boot sector)

Theo cấu trúc trên, trên mỗi volume sẽ tồn tại một boot sector hay còn gọi là sector khởi động. Sector này chứa các thông tin quan trọng để truy nhập đến các vùng trên volume và các cấu trúc. Boot sector được DOS khởi tạo trong quá trình phân vùng (`fdisk`) đĩa. Nó luôn có cùng một cấu trúc duy nhất và được đặt ở sector 0 để DOS có thể tìm thấy nó và giải nghĩa chính xác.

Địa chỉ	Nội dung	Kiểu
00h	Lệnh nhảy đến thủ tục khởi động	3 bytes
03h	Tên hãng sản xuất và version của DOS	8 bytes
0Bh	Số byte trên một sector	1 word
0Dh	Số sector trên một cluster (đơn vị cấp phát)	1 byte
0Eh	Số sector để dành	1 word
10h	Số bản FAT	1 byte
11h	Số đe mục trong thư mục gốc	1 word
13h	Số sector trong volume	1 word
15h	Byte mô tả đĩa	1 byte
16h	Số sector của một FAT	1 word
18h	Số sector trên một track	1 word
1Ah	Số đầu đọc/ghi	1 word
1Ch	Khoảng cách từ sector đầu tiên của volume đến sector đầu tiên chứa dữ liệu	1 word
1Eh - 1FFh	Thủ tục khởi động	482 bytes
		Tổng cộng 512 bytes

Bảng 9.4 - Cấu trúc boot sector của một volume

Sector này chứa tất cả các thông tin cần thiết để khởi động DOS. Trên thực tế, DOS không có mặt trong bộ nhớ ngay sau khi bật máy tính, do đó ban đầu cần phải nạp DOS và sau đó là khởi động. Để làm việc này, BIOS thực hiện khởi tạo hệ thống sau khi bật máy và cũng chính nó nạp vào bộ nhớ sector 0 của đĩa. Sau đó, chuyển quyền điều khiển cho chương trình con nạp hệ điều hành nằm trong sector này.

Boot sector luôn chứa một lệnh JUMP ở địa chỉ 0. Lệnh này có thể là một lệnh nhảy bình thường hoặc lệnh nhảy gần để chuyển đến thủ tục khởi động hay còn gọi là boot strap. Thủ tục này chịu trách nhiệm nạp và khởi động DOS.

Sau boot sector có thể có một số sector để dành, chúng có thể được dùng để chứa phần tiếp theo của mã chương trình boot strap.

3. Bảng phân vùng file (FAT)

Khi DOS muốn tạo các file mới hoặc tăng kích thước của các file đã tồn tại, chắc chắn nó phải biết các sector tự do trên đĩa. DOS lấy thông tin này từ một cấu trúc gọi là FAT (File Allocation Table) hay còn gọi là bảng phân vùng file. FAT được đặt trên volume ngay sau vùng để dành, nếu không có vùng để dành thì nó nằm ngay sau boot sector. Mỗi mục của FAT tương ứng với một cluster, mỗi cluster có thể chứa một hoặc nhiều sector đặt kế tiếp nhau. Số lượng các sector tạo nên một cluster được đặt trong ô nhớ 0Dh của boot sector, giá trị này luôn là lũy thừa của 2 và thay đổi tùy theo loại đĩa (có thể là 1,2,4 hoặc 8).

Việc nhóm các sector thành một cluster (hay còn gọi là allocation unit - đơn vị cấp phát) là cách mà DOS dùng để ghi các file. Trên thực tế, DOS không tự động chọn các sector kế tiếp nhau để ghi file vì sẽ tiết kiệm hơn nếu chia nhỏ các file sao cho có thể dùng tất cả các sector còn tự do ngay cả khi các sector này không liên tục. Nhưng phương pháp này làm giảm tốc độ truy nhập file vì đầu từ đọc/ghi cần phải định vị lại mỗi lần truy nhập. Do vậy, để tránh việc chia quá nhỏ các file, DOS nhóm nhiều sector liên tục thành một cluster và cluster là đơn vị để cấp phát không gian nhớ cho file.

Quay trở lại với bảng FAT, chúng ta thấy chức năng của bảng FAT là làm nhiệm vụ quản lý không gian nhớ tự do trên đĩa. Bảng FAT là một danh sách móc nối các cluster trên đĩa. Tuỳ theo dung lượng đĩa mà DOS sử dụng bảng FAT 12 bits hoặc bảng FAT 16 bits. Ý nghĩa các phần tử trong bảng FAT được thể hiện qua bảng sau:

Mã	Ý nghĩa
(0)000h	Cluster tự do
(F)FF0h - (F)FF6h	Cluster để dành
(F)FF7h	Cluster bị hỏng, không dùng
(F)FF8h - (F)FFFh	Cluster cuối cùng của một file
(X)XXXh	Cluster tiếp theo của file

Bảng 9.5 - Cấu trúc bảng FAT

4. Thư mục gốc (Root Directory)

Thư mục gốc của một volume có chức năng lưu trữ các thông tin về các file chứa trong volume đó, nó được đặt ngay sau bản sao cuối cùng của FAT. Mỗi phân tử trong thư mục gốc có kích thước 32 bytes cung cấp các thông tin về tên file, tên thư mục con và tên volume. Số lượng cực đại các phân tử trong thư mục gốc do lệnh Format quyết định.

Địa chỉ	Nội dung	Kiểu
00h	Tên file	8 bytes
08h	Phần mở rộng của file	3 bytes
0Bh	Thuộc tính file	1 byte
0Ch	Không công bố	10 bytes
16h	Giờ của lần thay đổi cuối cùng	1 word
18h	Ngày của lần thay đổi cuối cùng	1 word
1Ah	Cluster đầu tiên của file	1 word
1Ch	Kích thước file	2 word
Tổng cộng		32 bytes

Bảng 9.6 - Cấu trúc phân tử của thư mục gốc

* Ý nghĩa các trường trong thư mục gốc như sau:

- Trường đầu tiên chứa tên file, nếu tên file ngắn hơn 8 bytes, nó sẽ được làm đầy bằng các dấu trắng. Nếu một phân tử trong thư mục gốc không chứa các thông tin liên quan đến file thì byte đầu tiên của trường tên file sẽ nhận một mã đặc biệt:

Mã	Ý nghĩa
00h	Phân tử cuối cùng của thư mục
2Eh	File liên quan đến thư mục con hiện tại, (nếu có một 2Eh nữa tiếp theo, nó liên quan đến thư mục mẹ)
E5h	File đã bị xóa

- Trường thứ hai chứa phần mở rộng của file, nếu không có phần mở rộng hoặc phần mở rộng nhỏ hơn 3 ký tự, nó sẽ được làm đầy bằng các dấu trắng. Dấu chấm giữ tên file và phần mở rộng không được lưu trữ.

- Trường thuộc tính có kích thước 1 byte. Các thuộc tính khác nhau có thể tổ hợp lại, do đó một file có thể có đồng thời nhiều thuộc tính (chỉ đọc, ẩn, hệ thống, lưu trữ).

- Trường để dành DOS sẽ dùng cho các thao tác bên trong của nó.

- Các trường giờ và ngày cho biết file được tạo hay thay đổi lần cuối vào thời điểm nào.

- Trường tiếp theo cho biết số thứ tự cluster đầu tiên chứa dữ liệu của file. Giá trị này cũng cho biết số thứ tự của phần tử tương ứng với file trong bảng FAT và chỉ ra số thứ tự của cluster tiếp theo của file. Như vậy, trường này là mốc xích đầu tiên cho phép tìm ra các cluster trong file.

Kích thước file tính bằng byte được ghi dưới dạng một từ kép (2 từ) bao gồm một từ thấp và một từ cao. Công thức tính kích thước file như sau:

$$\text{Kích thước file} = \text{từ thấp} + \text{từ cao} * 65536$$

5. Thư mục con

Thư mục con được tổ chức theo kiểu kết hợp phương pháp tổ chức file và thư mục gốc. Các phần tử tương ứng với thư mục con được tổ chức theo kiểu của thư mục gốc trong khi bản thân nó được lưu trữ theo kiểu file (với trường độ dài luôn bằng 0). Để truy nhập thư mục con, cũng phải truy nhập một danh sách các cluster trong bảng FAT. Như vậy, khi truy nhập thư mục con phải kết hợp cả hai phương pháp truy nhập vào file và truy nhập vào thư mục gốc.

Khi thư mục con được tạo ra, hai phần tử lập tức xuất hiện với các tên “.” và “..”, các phần tử này chỉ mất đi khi toàn bộ thư mục con bị xoá. Phần tử đầu tiên “.” chỉ thư mục con hiện tại và số thứ tự cluster đầu tiên của thư mục con hiện tại. Phần tử “..” chỉ ra thư mục mẹ, nó là dấu mốc để tìm ra mối liên hệ giữa thư mục con hiện tại với các thư mục còn lại trong cây thư mục. Nếu thư mục mẹ đồng thời là thư mục gốc thì trường cluster có giá trị bằng 0. Chính phần tử này cho phép DOS tìm ra con đường dẫn đến thư mục gốc (vì mỗi lần tìm ra thư mục mẹ của thư mục con, chúng ta sẽ tiến dần đến thư mục gốc).

Mặc dù DOS xử lý các thư mục con như các file nhưng chúng ta không thể áp dụng các hàm truy nhập file để truy nhập thư mục con mà phải áp dụng phương pháp sau:

- Căn cứ vào cluster đầu tiên để xác định địa chỉ lấy thông tin trên đĩa và truy nhập tới bảng FAT để xác định danh sách cluster.

- Đọc thông tin trên đĩa theo danh sách tìm được.
- Phân tích thông tin tìm được theo kiểu truy nhập thư mục gốc.

V. QUẢN LÝ FILE CỦA DOS

Để quản lý các file, DOS sử dụng một số hàm cơ bản. Đây là những hàm mà một hệ điều hành bắt buộc phải cung cấp cho người lập trình như: hàm tạo file, hàm xoá file, hàm mở và đóng file, hàm đọc/ghi file. Ngoài ra, DOS còn có một loạt các hàm mở rộng như hàm cung cấp các thông tin về file, hàm đổi tên file, ...

Như ta đã biết, hệ điều hành DOS được xây dựng dựa trên nền của hệ điều hành CP/M 80 nên để đảm bảo tính tương thích với hệ điều hành này, trong DOS tồn tại hai dạng hàm để quản lý file đó là các hàm FCB và các hàm Handle.

1. Các hàm FCB

1.1. Cấu trúc FCB

Các hàm FCB, được xây dựng dựa trên một cấu trúc dữ liệu gọi là FCB (File Control Block - Khối quản lý file). FCB là một cấu trúc dữ liệu có độ dài 37 bytes được chia thành các trường có kích thước khác nhau.

Địa chỉ	Nội dung	Kiểu
00h	Số thiết bị	1 byte
01h	Tên file	8 bytes
09h	Phần mở rộng của file	3 bytes
0Ch	Số khối hiện thời	1 word
0Eh	Kích thước bản ghi	1 word
10h	Kích thước file	2 word
14h	Ngày tháng lần thay đổi gần nhất	1 word
16h	Giờ thay đổi gần nhất	1 word
18h	Không công bố	8 bytes
20h	Số bản ghi hiện thời	1 byte
21h	Số bản ghi cho sự truy nhập ngẫu nhiên	2 word
Tổng cộng		37 bytes

Bảng 9.7 - Cấu trúc khối điều khiển file FCB

* Ý nghĩa các trường trong FCB:

- Số thiết bị: nếu byte này có giá trị
 - 0: chỉ ổ đĩa hiện thời
 - 1: chỉ ổ đĩa A
 - 2: chỉ ổ đĩa B
 - 3: ...

- Tên file: là chuỗi ký tự dài 8 bytes

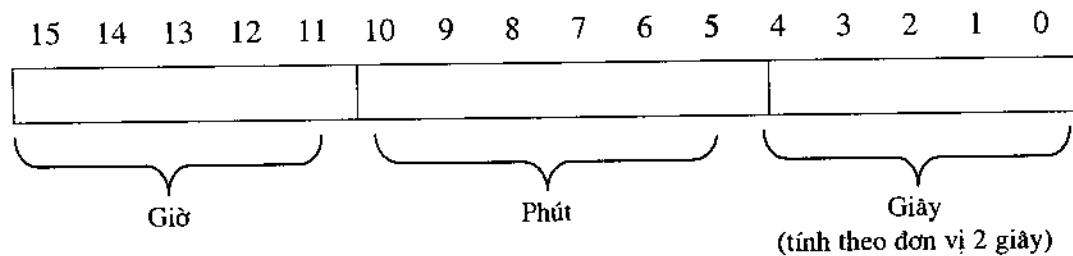
- Phần mở rộng: là chuỗi ký tự dài 3 bytes

- Số khối hiện thời: cho biết số thứ tự của khối đang được xử lý trong chế độ tuần tự

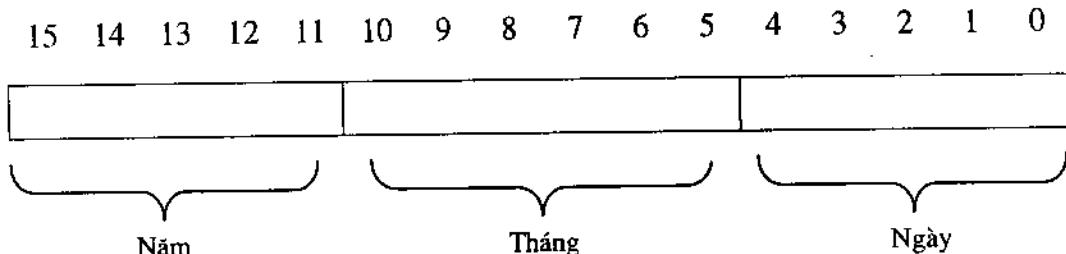
- Kích thước bản ghi: cho biết độ dài một lần đọc/ghi (từ 1-65533) và số lượng ký tự DOS chuyển đồng thời trong một lần đọc/ghi. Kích thước file được tính theo công thức:

$$\hat{O} \text{ nhớ } 10h + \hat{O} \text{ nhớ } 11h * 256 + (\hat{O} \text{ nhớ } 12h + \hat{O} \text{ nhớ } 13h * 256) * 65536$$

- Thông tin về ngày, tháng, giờ của lần thay đổi gần nhất: được lưu kể từ ô 14h. Hai thông tin này được lưu trữ dưới dạng mã như sau:



Hình 9.3 - Khuôn dạng của trường giờ trong FCB



Hình 9.4 - Khuôn dạng của trường ngày tháng trong FCB

- Không được công bố: là một vùng 8 bytes dành riêng cho DOS, người sử dụng không được phép thay đổi 8 bytes này. Ý nghĩa của nó không được Microsoft công bố và nó thay đổi qua các thế hệ của DOS.

- Số bản ghi hiện thời: cho biết số hiệu của bản ghi hiện thời đang xử lý trong chế độ tuần tự.

- Số bản ghi cho sự truy nhập ngẫu nhiên: cho biết số hiệu của bản ghi đầu tiên trong chế độ ngẫu nhiên. Chế độ truy nhập này không gọi lần lượt các bản ghi theo trật tự logic mà thao tác theo một trật tự bất kỳ.

Để truy nhập tới các file có thuộc tính đặc biệt (như các file ẩn hoặc hệ thống), DOS cung cấp một FCB mở rộng có độ dài 44 bytes. Cấu trúc của FCB mở rộng tương tự như FCB thường nhưng được bổ sung thêm một vùng có kích thước 7 bytes được đặt tại đầu FCB. Do đó, các trường về sau được tịnh tiến 7 bytes so với các FCB thường.

Địa chỉ	Nội dung	Kiểu
00h	Dấu hiệu FCB mở rộng (FFh)	1 byte
01h	Không công bố	5 bytes
06h	Thuộc tính file	1 byte
07h	Số thiết bị	1 byte
08h	Tên file	8 bytes
10h	Phân mở rộng của file	3 bytes
13h	Số khối hiện thời	1 word
15h	Kích thước bản ghi	1 word
17h	Kích thước file	2 word
1Bh	Ngày tháng lần thay đổi gần nhất	1 word
1Dh	Giờ thay đổi gần nhất	1 word
1Fh	Không công bố	8 bytes
27h	Số bản ghi hiện thời	1 byte
28h	Số bản ghi cho sự truy nhập ngẫu nhiên	2 word
		Tổng cộng 44 bytes

Bảng 9.8 - Cấu trúc FCB mở rộng

* Ý nghĩa các trường bổ sung trong FCB mở rộng:

- Dấu hiệu FCB mở rộng: luôn chứa giá trị 255, nó dùng để nhận dạng FCB mở rộng. Trong FCB thường, ô nhớ này chứa số ổ đĩa mà số ổ đĩa thì không bao giờ là 255. Do đó, DOS có thể phân biệt một cách chính xác giữa FCB thường và FCB mở rộng. Sự phân biệt này là lý do chính cho phép DOS chấp nhận cả FCB thường và FCB mở rộng.

- Không được công bố: là một vùng 5 bytes dành riêng cho DOS, người sử dụng không được phép thay đổi 5 bytes này. Ý nghĩa của nó không được Microsoft công bố và nó thay đổi qua các thế hệ của DOS.

- Thuộc tính file: chứa thuộc tính của file.

1.2. Danh sách các hàm FCB

Số hàm	Chức năng
0Fh	Mở file
10h	Đóng file
13h	Xóa file
14h	Đọc tuần tự
15h	Ghi tuần tự
16h	Tạo file
17h	Đổi tên file
1Ah	Đặt địa chỉ DTA(data area)
21h	Đọc ngẫu nhiên bản ghi
22h	Ghi ngẫu nhiên (1 bản ghi)
23h	Xác định kích thước file
24h	Đặt tham số bản ghi để truy nhập ngẫu nhiên
27h	Đọc ngẫu nhiên (1 hay nhiều bản ghi)
28h	Ghi ngẫu nhiên (1 hay nhiều bản ghi)
29h	Phân tích tên file trong FCB

Bảng 9.9 - Danh sách các hàm FCB.

1.3. Phương pháp gọi hàm

Các hàm FCB cho phép truy nhập đến nhiều file, do đó mỗi file có một FCB riêng biệt. Để chỉ cho DOS biết cần truy nhập file nào, mỗi hàm FCB phải có địa chỉ FCB dành cho file đó.

Địa chỉ đoạn đặt trong thanh ghi DS, địa chỉ offset đặt trong thanh ghi DX. Sau khi kết thúc, các hàm trả lại giá trị trong thanh ghi AL cho biết việc thực hiện có lỗi hay không. Nếu không có lỗi, trả lại giá trị 0, ngược lại trả giá trị 255.

2. Các hàm Handle

2.1. Khái niệm về Handle

So với việc truy nhập file bằng FCB, truy nhập file bằng các hàm Handle thuận lợi hơn nhiều vì không cần phải cung cấp cho DOS một cấu trúc dữ liệu dưới dạng FCB, sự truy nhập file được thực hiện thông qua tên file. Khi mở file hoặc tạo file, tên file sẽ được truyền cho hàm dưới dạng một chuỗi ký tự ASCII. Chuỗi này không chỉ chứa tên file mà còn chứa cả tên thiết bị, nó chỉ ra đường dẫn đầy đủ cũng như phần mở rộng của file. Sau đó, các hàm trả lại một chuỗi kí tự 16 bit gọi là Handle (thé file) và từ đó trở đi, tất cả các thao tác truy nhập file đều thông qua thé này.

Số lượng các thé file được mở đồng thời xác định thông qua một lệnh của hệ thống trong file CONFIG.SYS. Lệnh đó là: FILES = n

Trong đó n là số thé file được mở đồng thời (có giá trị cực đại là 255). Người sử dụng có thể thay đổi giá trị này sao cho phù hợp với các yêu cầu của mình nhưng mọi thay đổi chỉ có hiệu lực sau khi khởi động lại hệ thống. Tuy số lượng thé file được mở là khá lớn nhưng DOS không cho phép một chương trình của người sử dụng dùng quá 20 thé file.

2.2. Danh sách các hàm Handle

Số hàm	Chức năng
3Ch	Tạo file
3Dh	Mở file
3Eh	Đóng file

42h	Đặt con trỏ file, tính kích thước file
43h	Đọc/ghi thuộc tính file
56h	Đổi tên file
57h	Đọc/ghi thời gian thay đổi file

Bảng 9.10 - Danh sách các hàm FCB

2.3. Phương pháp gọi hàm

Cách mở hoặc tạo file, nhận tên file (thực chất là địa chỉ của tên file) như tham số vào. Địa chỉ đoạn đặt trong thanh ghi DS, địa chỉ offset đặt trong thanh ghi DX. Sau khi thực hiện, các hàm trả lại một thẻ file trong thanh ghi AX.

Các hàm còn lại nhận thẻ file như tham số vào và đặt trong thanh ghi BX. Sau khi thực hiện sẽ thông báo trạng thái qua cờ Carry. Nếu có lỗi, cờ Carry có giá trị 1 và thanh ghi AX sẽ chứa mã lỗi (mã này chỉ ra nguyên nhân gây lỗi hoặc bản thân lỗi).

VI. TRUY NHẬP THƯ MỤC CỦA DOS

Các hàm của DOS liên quan đến thư mục được chia thành hai nhóm: nhóm các hàm thao tác với thư mục con và nhóm các hàm tìm kiếm file trong thư mục.

1. Các hàm thao tác với thư mục con

Ở mức người sử dụng, các hàm thao tác với thư mục con được cung cấp dưới dạng các lệnh MD (tạo thư mục), RD (xoá thư mục), CD (chuyển thư mục hiện thời). Xét về mức độ hệ thống, các lệnh này tương ứng với các hàm 39h, 3Ah và 3Bh của ngắt 21h. Đó là các hàm mà chúng ta sẽ xét sau đây:

- Hàm 39h - Tạo thư mục: đường dẫn mà hàm này nhận chứa tên thư mục cần tạo như là thành phần cuối cùng của đường dẫn. Lỗi có thể xuất hiện nếu một hay nhiều thư mục trong tên đường dẫn không tồn tại hoặc thư mục cần tạo đã tồn tại.

- Hàm 3Ah - Xoá thư mục: DOS sẽ không thực hiện hàm này trong một số trường hợp như khi thư mục cần xoá không rỗng (có chứa các file, thư mục con) hoặc thư mục cần xoá là thư mục hiện thời.

- Hàm 3Bh - Chuyển thư mục hiện thời: khi chuyển thư mục hiện thời, các tên trong đường dẫn phải thực sự tồn tại. Đây là nguồn phát sinh lỗi duy nhất trong hàm này.

Đối với các hàm này, ý nghĩa các thanh ghi trước và sau khi gọi hàm khá giống nhau. Cần phải đặt số hàm vào thanh ghi AH, và phải truyền cho hàm tên đường dẫn thư mục. Tên đường dẫn có thể là đầy đủ, nó chứa cả tên ổ đĩa (được đặt ở đầu, sau đó là dấu “:”). Nếu bỏ qua tên ổ đĩa, DOS sẽ coi lời gọi hàm liên quan đến ổ đĩa hiện thời.

Các hàm đều nhận địa chỉ đoạn của vùng nhớ chứa tên đường dẫn trong thanh ghi DS và địa chỉ offset trong thanh ghi DX. Sau khi thực hiện, nếu cờ Carry bằng 0 tức là hàm thực hiện không có lỗi, ngược lại, cờ Carry bằng 1 thì một mã lỗi được trả lại trong thanh ghi AX.

2. Các hàm tìm file

2.1. Các hàm FCB

Để tìm file bằng FCB, DOS cung cấp cho người sử dụng hai hàm 11h và 12h. Hàm 11h dùng để tìm file đầu tiên trong thư mục hiện thời thoả mãn điều kiện đã chỉ ra còn hàm 12h tìm tất cả các file còn lại thoả mãn điều kiện.

Phương pháp tìm kiếm: gọi hàm 11h với số hàm trong AH, đặt địa chỉ đoạn của FCB cần tìm trong DS và địa chỉ offset trong thanh ghi DX. Nếu tìm thấy file thì AL chứa giá trị 0 ngược lại 255. Muốn tiếp tục tìm kiếm các file khác thì gọi hàm 12h. Việc định nghĩa các thanh ghi cho hàm 12h tương tự hàm 11h.

2.2. Các hàm Handle

Để tìm kiếm file bằng hàm Handle, người sử dụng có thể dùng hai hàm 4Eh và hàm 4Fh. Tương tự như hàm FCB 11h và 12h, hàm 4Eh cho phép tìm kiếm file đầu tiên thoả mãn điều kiện còn 4Fh cho phép tìm kiếm các file còn lại thoả mãn điều kiện.

Phương pháp tìm kiếm: số hàm đặt trong AH, thuộc tính file đặt trong CX, địa chỉ vùng nhớ chứa tên file cần tìm đặt trong DS:DX (địa chỉ đoạn và địa chỉ offset). Nếu tìm thấy file, cờ Carry chứa giá trị 0 ngược lại, nó chứa giá trị 1 và thanh ghi AX chứa mã lỗi. Mã này bằng 1 nếu đường dẫn đã chỉ không tồn tại, bằng 12h nếu không tìm thấy file. Mọi sự tìm kiếm tiếp theo đều dựa vào hàm 4Fh, hàm này chỉ cần số hàm trong thanh ghi AH, cờ Carry sẽ cho biết có còn file thoả mãn trong thư mục nữa hay không.

VII. QUẢN LÝ THIẾT BỊ CỦA DOS

Chúng ta đã biết driver thiết bị là một thành phần của hệ điều hành, chúng dùng để điều khiển và liên lạc với các thiết bị phần cứng. Chúng liên quan đến mức thấp nhất của hệ điều hành vì chúng cho phép tất cả các mức hoạt

động độc lập với phần cứng của máy tính. Theo cách này, chúng ta chỉ cần làm cho các driver thiết bị tương thích với phần cứng máy tính mà không cần phải thay đổi lại hệ điều hành.

Trong các phiên bản trước, các driver thiết bị được nạp vào mã của hệ điều hành một cách cứng nhắc, không thể thay đổi được. Các phiên bản sau của DOS (từ version 2.0) sử dụng một phương pháp mềm dẻo hơn là ghép các driver thiết bị vào DOS dần đến có thể sửa chữa và bổ sung thêm các driver mới mà không gây ảnh hưởng tới hệ điều hành.

Thao tác nạp driver của DOS bao gồm: trước tiên là nạp các driver thiết bị chuẩn như NUL, \$CLOCK, CON, AUX, PRN và các driver điều khiển ổ đĩa mềm, đĩa cứng nếu có. Các driver này được đặt trực tiếp trong bộ nhớ, cái này sau cái kia tạo thành một chuỗi liên kết. Nếu người sử dụng muốn nạp các driver đặc biệt thì cần phải khai báo cho DOS biết trong file CONFIG.SYS, file này chứa một số thông tin hướng dẫn cho DOS cấu hình hệ thống, nó được nạp trong quá trình nạp DOS. Nếu DOS gặp trong file CONFIG.SYS dòng lệnh DEVICE = , nó sẽ hiểu rằng có một driver thiết bị mới cần được nạp. Tên của driver và đường dẫn cần được chỉ ra sau dấu “=” trong câu lệnh DEVICE.

1. Phân loại các driver thiết bị của DOS

Các driver điều khiển thiết bị được DOS chia thành hai loại:

- Các driver điều khiển ký tự: loại này chỉ truyền một byte mỗi lần gọi hàm. Chúng phù hợp cho việc liên lạc với các thiết bị như: bàn phím, màn hình, máy in và modem. Mỗi driver chỉ phục vụ duy nhất cho một thiết bị nên khi nạp hệ thống, trong DOS tồn tại mỗi thiết bị một driver riêng.

- Các driver khối: dùng để liên lạc với bộ nhớ ngoài như đĩa cứng, đĩa mềm và một số thiết bị khác. Chúng không truyền một ký tự trong mỗi lần gọi hàm mà truyền một số lượng xác định các ký tự. Ta gọi số lượng các ký tự đó là một khối. Trong một số trường hợp, nhiều khối có thể được truyền qua mỗi lần gọi hàm, kích thước các khối có thể thay đổi tùy theo loại thiết bị.

2. Cấu trúc một driver thiết bị

Mặc dù hai loại driver thiết bị khác nhau ở một số khía cạnh quan trọng nhưng chúng lại có chung một cấu trúc. Cấu trúc này bao gồm một đầu driver (header), một thủ tục chiến lược và một thủ tục ngắn.

- Phần header: gồm một số thông tin chính cho phép DOS biết cách sử dụng driver. Nó luôn nằm ở phần đầu của driver thiết bị và có cấu trúc như sau:

Địa chỉ	Nội dung	Kiểu
00h	Địa chỉ của driver tiếp theo	1 point
04h	Thuộc tính của thiết bị	1 word
06h	Địa chỉ offset của thủ tục chiến lược	1 word
08h	Địa chỉ offset của thủ tục ngắt	1 word
0Ah	Tên của driver	8 bytes
Tổng cộng		18 bytes

Bảng 9.11 - Cấu trúc đầu của driver thiết bị

- Thủ tục chiến lược: dùng để cài đặt driver thiết bị khi nạp hệ thống. Cũng như mỗi lần gọi một hàm nào đó, thủ tục này nhận trong cặp thanh ghi ES:BX địa chỉ của một cấu trúc dữ liệu chứa các thông tin về hàm cần thực hiện và các dữ liệu tương ứng. Tuy nhiên, thủ tục chiến lược không thực hiện bản thân các hàm, nó chỉ lưu lại nội dung của khối dữ liệu được truyền, sau đó chuyển điều khiển về cho DOS.

Địa chỉ	Nội dung	Kiểu
00h	Độ dài của khối dữ liệu tính bằng byte	1 byte
01h	Số thiết bị được gọi (chỉ áp dụng đối với driver khối)	1 byte
02h	Số hàm cần gọi	1 byte
03h	Tù trạng thái	1 word
05h	Không công bố	8 bytes
0Dh	Byte mô tả thiết bị (chỉ áp dụng đối với driver khối)	1 byte
0Eh	Địa chỉ vùng đệm	1 point
12h	Số lượng các sector cần thao tác (đối với driver khối) Số lượng các byte cần thao tác (đối với driver thiết bị)	1 word
14h	Số thứ tự sector đầu tiên (chỉ áp dụng đối với driver khối)	1 word
Tổng cộng		18 bytes

Bảng 9.12 - Cấu trúc khối tham số để gọi hàm của driver thiết bị

- Thủ tục ngắn: được gọi trực tiếp ngay sau khi gọi thủ tục chiến lược. Nhiệm vụ đầu tiên là nó cất ngăn xếp các thanh ghi của bộ xử lý tránh việc có thể bị thay đổi. Sau đó nó đọc trong trường 3 của khối dữ liệu số hàm cần thực hiện và hàm này được gọi ngay lập tức. Sau khi thực hiện hàm, thủ tục ngắn viết một giá trị vào trường trạng thái của khối dữ liệu. Cuối cùng, nó khôi phục các thanh ghi và trả quyền điều khiển về cho DOS.

Giá trị trong trường trạng thái rất quan trọng đối với DOS, nó cho biết hàm thực hiện có kết quả hay gặp lỗi.

3. Các hàm của một driver thiết bị

Mỗi driver thiết bị cần phải được đảm bảo một số hàm cần thiết. Một số hàm này liên quan đến cả hai loại driver thiết bị nhưng cũng có những hàm chỉ liên quan đến driver kiểu ký tự hoặc driver kiểu khối. Danh sách các hàm được mô tả qua bảng sau:

Số hàm	Chức năng
00h	Khởi tạo driver thiết bị
01h	Kiểm tra đĩa
02h	Tạo một khối tham số BIOS (BPB)
03h	Đọc trực tiếp
04h	Đọc
05h	Đọc ký tự
06h	Trạng thái vào
07h	Xoá vùng đệm vào
08h	Ghi
09h	Ghi và kiểm tra
0Ah	Trạng thái ra
0Bh	Xoá vùng đệm ra

0Ch	Ghi trực tiếp
0Dh	Mở thiết bị
0Eh	Đóng thiết bị
0Fh	Đĩa thay được
10h	Đưa dữ liệu ra khi nào không bận

Bảng 9.13 - Danh sách các hàm của một driver thiết bị

Câu hỏi ôn tập

- Trình bày cấu trúc hệ điều hành DOS và quá trình khởi động DOS.
- So sánh chương trình COM và chương trình EXE.
- Trình bày phương pháp quản lý bộ nhớ RAM của DOS theo phương pháp MCB.
- Trình bày cấu trúc logic của đĩa từ dưới quan điểm của DOS. Nêu các khái niệm về Boot sector, FAT, Root Directory.
- Nêu mục đích của driver thiết bị và phân loại các driver thiết bị của DOS.
- Trình bày cấu trúc của một driver thiết bị.

TÀI LIỆU THAM KHẢO

1. *Operating System Concepts.* 1990. Abraham Silberschats - University of Texas at Austin and James L. Peterson - Microelectronic and Computer Technology Corporation.
2. *Giáo trình Hệ điều hành.* 1995. Nguyễn Thanh Tùng - Khoa Công nghệ thông tin - Đại học Bách khoa Hà Nội.
3. *Giáo trình Nguyên lý các hệ điều hành.* 1998. Hà Quang Thụy - Đại học Khoa học tự nhiên - Đại học Quốc gia Hà Nội.
4. *Cẩm nang lập trình hệ thống.* 1992. Michael Tischer - Người dịch: Nguyễn Mạnh Hùng, Phạm Tiến Dũng.

MỤC LỤC

<i>Lời giới thiệu</i>	3
<i>Lời nói đầu</i>	5
<i>Bài mở đầu</i>	7
<i>Chương 1.</i> TỔNG QUAN VỀ HỆ ĐIỀU HÀNH	10
I. Các khái niệm cơ bản	10
II. Các chức năng cơ bản của hệ điều hành	18
III. Các thành phần của hệ điều hành	20
<i>Chương 2.</i> QUẢN LÝ TIẾN TRÌNH	23
I. Các khái niệm cơ bản	23
II. Các phương pháp giải quyết bài toán đoạn tối hạn	28
III. Hiện tượng bế tắc	34
<i>Chương 3.</i> LẬP LỊCH CHO CPU	43
I. Các khái niệm cơ bản	43
II. Các thuật toán lập lịch	46
III. Ngắt	52
<i>Chương 4.</i> QUẢN LÝ BỘ NHỚ TRONG	56
I. Các khái niệm cơ bản	56
II. Các cấu trúc cơ bản của chương trình	58
III. Các sơ đồ quản lý bộ nhớ	63
IV. Bộ nhớ ảo	76
<i>Chương 5.</i> QUẢN LÝ BỘ NHỚ NGOÀI	80
I. Các khái niệm cơ bản	80
II. Các phương pháp quản lý không gian nhớ tự do	81
III. Các phương pháp cấp phát không gian nhớ tự do	83
IV. Lập lịch cho đĩa	86
V. Hệ file	90

<i>Chương 6. QUẢN LÝ THIẾT BỊ</i>	96
I. Nguyên tắc tổ chức và quản lý thiết bị	96
II. Các kỹ thuật áp dụng trong quản lý thiết bị	98
<i>Chương 7. BẢO VỆ VÀ AN TOÀN HỆ THỐNG</i>	105
I. Bảo vệ hệ thống	105
II. An toàn hệ thống	109
III. Virus máy tính	112
<i>Chương 8. HỆ ĐIỀU HÀNH ĐA XỬ LÝ</i>	119
I. Tổng quan về hệ đa xử lý	119
II. Hệ nhiều CPU	120
III. Hệ phân tán	123
<i>Chương 9. HỆ ĐIỀU HÀNH DOS</i>	131
I. Tổng quan về DOS	131
II. Phương pháp thực hiện chương trình của DOS	134
III. Quản lý bộ nhớ RAM của DOS	141
IV. Quản lý đĩa từ của DOS	144
V. Quản lý file của DOS	150
VI. Quản lý thư mục của DOS	155
VII. Quản lý thiết bị ngoại vi của DOS	156
<i>Tài liệu tham khảo</i>	161

NHÀ XUẤT BẢN HÀ NỘI
4 - TỔNG DUY TÂN, QUẬN HOÀN KIẾM, HÀ NỘI
ĐIỆN THOẠI: (04)8.257063; 8.252916. FAX: (04)8.257063

GIÁO TRÌNH
NGUYỄN LÝ HỆ ĐIỀU HÀNH
NHÀ XUẤT BẢN HÀ NỘI - 2005

Chịu trách nhiệm xuất bản:
NGUYỄN KHẮC OÁNH

Biên tập:
TRƯỜNG ĐỨC HÙNG

Bìa:

PHAN ANH TÚ

Trình bày - Kỹ thuật vi tính:
HOÀNG LAN HƯƠNG

Sửa bản in:
ĐÀM LY

In 1.560 cuốn, khổ 17 x 24 cm, tại Công ty In Khoa học Kỹ thuật - Hà Nội.
Số in: 184. Giấy phép xuất bản số: 113GT/407 CXB cấp ngày 29/3/2005.
In xong và nộp lưu chiểu quý II năm 2005.

BỘ GIÁO TRÌNH XUẤT BẢN NĂM 2005
KHỐI TRƯỜNG TRUNG HỌC KINH TẾ KỸ THUẬT TIN HỌC

1. THUẬT TOÁN LẬP TRÌNH
2. ĐÁNH MÁY VI TÍNH
3. SOẠN THẢO VÀ ĐÁNH MÁY VĂN BẢN
4. NGHIỆP VỤ THƯ KÝ
5. KẾ TOÁN MÁY
6. MARKETING
7. NGÔN NGỮ LẬP TRÌNH C
8. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI (C++)
9. BẢNG TÍNH ĐIỆN TỬ (EXCEL)
10. VISUAL BASIC
11. CẤU TRÚC MÁY TÍNH
12. GIAO TIẾP
13. ACCESS
14. MẠNG MÁY TÍNH
15. THIẾT KẾ WEB
16. BẢO TRÌ PC
17. HỆ ĐIỀU HÀNH
18. CƠ SỞ DỮ LIỆU
19. PHÂN TÍCH THIẾT KẾ HỆ THỐNG
20. KỸ THUẬT SỐ
21. PHOTOSHOP
22. CAD/CAM

giáo trình nguyên lý hệ điều



1 005083 100240
21.500 VND
5 073 905519

Giá: 21.500 đ