BỘ LAO ĐỘNG - THƯƠNG BINH VÀ XÃ HỘI
TỔNG CỤC DẠY NGHỀ

GIÁO TRÌNH
Hệ Điều Hành
NGHỀ: KỸ THUẬT LẮP RÁP VÀ SỬA CHỮA MÁY TÍNH
TRÌNH ĐỘ: TRUNG CẤP

(Ban hành theo Quyết định số: 120/QĐ-TCDN ngày 25 tháng 02 năm 2013 của Tổng cục trưởng Tổng cục dạy nghề)

TUYÊN BỐ BẢN QUYỀN:

LỜI GIỚI THIỆU

Trong hệ thống kiến thức chuyên nghành trang bị cho sinh viên Công nghệ Thông tin (CNTT), giáo trình hệ điều hành góp phần cung cấp những nội dung kiến thức chung nhất về hệ điều hành, nội dung liên quan đến việc mô tả các phương pháp giả quyết các bài toán điều khiển hoạt động của hệ thống máy tính. Nội dung giáo trình hệ điều hành thuộc vào hệ thống kiến thức về phần mềm hệ thống, cung cấp những kiến thức nhằm thực hiện một trong hai nguyên lý cơ bản trong hệ thống máy tính đã được Von Neumann phát biểu, đó là nguyên lý "hoạt động theo chương trình".

Các nội dung chính được trình bày trong giáo trình này bao gồm năm chương được giới thiệu như dưới đây.

Bài mở đầu: **Giới thiệu chung về hệ điều hành**

Chương 1: **Điều khiển dữ liệu**

Chương 2: **Điều khiển bộ nhớ**

Chương 3: **Điều khiển CPU, điều khiển quá trình**

Chương 4: **Hệ điều hành đa xử lý**

Hà Nội, 2013

Tham gia biên soạn

Khoa Công Nghệ Thông Tin

Trường Cao Đẳng Nghề Kỹ Thuật Công Nghệ

Địa Chỉ: Tổ 59 Thị trấn Đông Anh – Hà Nội

Tel: 04. 38821300

Chủ biên: Nguyễn Kim Dung

Mọi góp ý liên hệ: Phùng Sỹ Tiến – Trưởng Khoa Công Nghệ Thông Tin
Mobible: 0983393834
Email: tienphungktcn@gmail.com – tienphungktcn@yahoo.com

MỤC LỤC

MÔN HỌC: HỆ ĐIỀU HÀNH

Mã môn học: MH15

Vị trí, ý nghĩa, vai trò môn học:

- Vị trí:

Môn học được bố trí sau khi học xong các môn học chung, trước các môn học/mô đun đào tạo chuyên ngành.

- Tính chất:

  Là môn học chuyên ngành.

- Ý nghĩa và vai trò môn học:

  Là môn học ứng dụng cơ bản để phát triển các môn học tiếp theo

  Là môn không thể thiếu của nghề Sửa chữa, lắp ráp máy tính

Mục tiêu của môn học:

- Hiểu vai trò và chức năng của hệ điều hành trong hệ thống máy tính
- Biết các giai đoạn phát triển của hệ điều hành
- Hiểu các nguyên lý thiết kế, hoạt động của hệ điều hành
- Hiểu cách giải quyết các vấn đề phát sinh trong hệ điều hành
- Có ý thức tự giác, tính kỷ luật cao, tinh thần trách nhiệm trong học tập.
- Tự tin trong nghiên cứu, tím hiểu các công nghệ hệ thống

Nội dung của môn học

Mã bài	Tên chương mục	Thời gian			
		Tổng số	Lý thuyết	Thực hành	Kiểm tra*
MH15-01	**Giới thiệu chung về hệ điều hành** Khái niệm về hệ điều hành Phân loại hệ điều hành Sơ lược lịch sử phát triển của HĐH	**10**	**8**	**2**	
MH15-02	**Điều khiển dữ liệu** Các phương pháp tổ chức và truy nhập dữ liệu Bản ghi và khối Điều khiển buffer Quy trình chung điều khiển nhập-xuất Tổ chức lưu trữ dữ liệu trên đĩa từ	**24**	**12**	**10**	**2**
MH15-03	**Điều khiển bộ nhớ** Quản lý và bảo vệ bộ nhớ Điều khiển bộ nhớ liên tục Điều khiển bộ nhớ gián đoạn	**24**	**14**	**8**	**2**
MH15-04	**Điều khiển CPU, Điều khiển quá trình** Trạng thái của quá trình Điều phối quá trình Bài toán đồng bộ hóa Bế tắc-Giải pháp phòng ngừa và xử	**20**	**10**	**8**	**2**

	lý				
MH15-05	**Hệ điều hành đa xử lý** Hệ điều hành đa xử lý tập trung Thuật toán song song và ngôn ngữ lập trình song song Hệ điều hành đa xử lý phân tán	**12**	**8**	**2**	**2**
Cộng		**90**	**52**	**30**	**8**

BÀI MỞ ĐẦU:
GIỚI THIỆU CHUNG VỀ HỆ ĐIỀU HÀNH
Mã chương: MH15-01

Giới thiệu:

Nếu không có phần mềm, máy tính chỉ là một thiết bị điện tử thông thường. Với sự hỗ trợ của phần mềm, máy tính có thể lưu trữ, xử lý thông tin và người sử dụng có thể gọi lại được thông tin này. Phần mềm máy tính có thể chia thành nhiều loại: chương trình hệ thống, quản lý sự hoạt động của chính máy tính. Chương trình ứng dụng, giải quyết các vấn đề liên quan đến việc sử dụng và khai thác máy tính của người sử dụng. Hệ điều hành thuộc nhóm các chương trình hệ thống và nó là một chương trình hệ thống quan trọng nhất đối với máy tính và cả người sử dụng. Hệ điều hành điều khiển tất cả các tài nguyên của máy tính và cung cấp một môi trường thuận lợi để các chương trình ứng dụng do người sử dụng viết ra có thể chạy được trên máy tính. Trong chương này chúng ta xem xét vai trò của hệ điều hành trong trường hợp này.

Một máy tính hiện đại có thể bao gồm: một hoặc nhiều processor, bộ nhớ chính, clocks, đĩa, giao diện mạng, và các thiết bị vào/ra khác. Tất cả nó tạo thành một hệ thống phức tạp. Để viết các chương trình để theo dõi tất cả các thành phần của máy tính và sử dụng chúng một cách hiệu quả, người lập trình phải biết processor thực hiện chương trình như thế nào, bộ nhớ lưu trữ thông tin như thế nào, các thiết bị đĩa làm việc (ghi/đọc) như thế nào, lỗi nào có thể xảy ra khi đọc một block đĩa, … đây là những công việc rất khó khăn và quá khó đối với người lập trình. Nhưng rất may cho cả người lập trình ứng dụng và người sử dụng là những công việc trên đã được hệ điều hành hỗ trợ nên họ không cần quan tâm đến nữa. Chương này cho chúng ta một cái nhìn tổng quan về những gì liên quuan đến việc thiết kế cài đặt cũng như chức năng của hệ điều hành để hệ điều hành đạt được mục tiêu: Giúp người sử dụng khai thác máy tính dễ dàng và chương trình của người sử dụng có thể chạy được trên máy tính.

Mục Tiêu:

Học xong chương này người học có khả năng:

- Nắm được yêu cầu cần có hệ điều hành
- Nắm được khái niệm hệ điều hành, chức năng, phân loại và các thành phần cơ bản trong hệ điều hành

Nội Dung Chính:

1. Khái niệm về hệ điều hành

Mục tiêu:

- *Nắm được yêu cầu cần có hệ điều hành, khái niệm hệ điều hành, chức năng hệ điều hành.*

Hệ điều hành là gì?

Khó có một khái niệm hay định nghĩa chính xác về hệ điều hành, vì hệ điều hành là một bộ phận được nhiều đối tượng khai thác nhất, họ có thể là người sử dụng thông thường, có thể là lập trình viên, có thể là người quản lý hệ thống và tùy theo mức độ khai thác hệ điều hành mà họ có thể đưa ra những khái niện khác nhau về nó. Ở đây ta xem xét 3 khái niệm về hệ điều hành dựa trên quan điểm của người khai thác hệ thống máy tính:

- **Khái niệm 1:** Hệ điều hành là một hệ thống mô hình hoá, mô phỏng hoạt động của máy tính, của người sử dụng và của lập trình viên, hoạt động trong chế độ đối thoại nhằm tạo môi trường khai thác thuận lợi hệ thống máy tính và quản lý tối ưu tài nguyên của hệ thống.
- **Khái niệm 2:** Hệ điều hành là hệ thống chương trình với các chức năng giám sát, điều khiển việc thực hiện các chương trình của người sử dụng, quản lý và phân chia tài nguyên cho nhiều chương trình người sử dụng đồng thời sao cho việc khai thác chức năng của hệ thống máy tính của người sử dụng là thuận lợi và hiệu quả nhất.
- **Khái niệm 3:** Hệ điều hành là một chương trình đóng vai trò như là giao diện giữa người sử dụng và phần cứng máy tính, nó điều khiển việc thực hiện của tất cả các loại chương trình. Khái niệm này rất gần với các hệ điều hành đang sử dụng trên các máy tính hiện nay.

Từ các khái niệm trên chúng ta có thể thấy rằng: Hệ điều hành ra đời, tồn tại và phát triển là để giải quyết vấn đề sử dụng máy tính của người sử dụng, nhằm giúp người sử dụng khai thác hết các chức năng của phần cứng máy tính mà cụ thể là giúp người sử dụng thực hiện được các chương trình của họ trên máy tính.

1.1.Các thành phần hệ thống

Hệ điều hành là một hệ thống chương trình lớn, thực hiện nhiều nhiệm vụ khác nhau, do đó các nhà thiết kế thường chia hệ điều hành thành nhiều thành phần, mỗi thành phần đảm nhận một nhóm các nhiệm vụ nào

đó, các nhiệm vụ này có liên quan với nhau. Cách phân chia nhiệm vụ cho mỗi thành phần, cách kết nối các thành phần lại với nhau để nó thực hiện được một nhiệm vụ lớn hơn khi cần và cách gọi các thành phần này khi cần nó thực hiện một nhiệm vụ nào đó, ... , tất cả các phương thức trên tạo nên cấu trúc của hệ điều hành.

1.1.1.Thành phần quản lý tiến trình

Hệ điều hành phải có nhiệm vụ tạo lập tiến trình và đưa nó vào danh sách quản lý tiến trình của hệ thống. Khi tiến trình kết thúc hệ điều hành phải loại bỏ tiến trình ra khỏi danh sách quản lý tiến trình của hệ thống.

Hệ điều hành phải cung cấp đầy đủ tài nguyên để tiến trình đi vào hoạt động và phải đảm bảo đủ tài nguyên để duy trì sự hoạt động của tiến trình cho đến khi tiến trình kết thúc. Khi tiến trình kết thúc hệ điều hành phải thu hồi những tài nguyên mà hệ điều hành đã cấp cho tiến trình.

Trong quá trình hoạt động nếu vì một lý do nào đó tiến trình không thể tiếp tục hoạt động được thì hệ điều hành phải tạm dừng tiến trình, thu hồi tài nguyên mà tiến trình đang chiếm giữ, sau đó nếu điều kiện thuận lợi thì hệ điều hành phải tái kích hoạt tiến trình để tiến trình tiếp tục hoạt động cho đến khi kết thúc.

Trong các hệ thống có nhiều tiến trình hoạt động song song hệ điều hành phải giải quyết vấn đề tranh chấp tài nguyên giữa các tiến trình, điều phối processor cho các tiến trình, giúp các tiến trình trao đổi thông tin và hoạt động đồng bộ với nhau, đảm bảo nguyên tắc tất cả các tiến trình đã được khởi tạo phải được thực hiện và kết thúc được.

- Tóm lại, bộ phận quản lý tiến trình của hệ điều hành phải thực hiện những nhiệm vụ sau đây:
  - Tạo lập, hủy bỏ tiến trình.
  - Tạm dừng, tái kích hoạt tiến trình.
  - Tạo cơ chế thông tin liên lạc giữa các tiến trình.
  - Tạo cơ chế đồng bộ hóa giữa các tiến trình.

1.1.2.Thành phần quản lý bộ nhớ chính

Bộ nhớ chính là một trong những tài nguyên quan trọng của hệ thống, đây là thiết bị lưu trữ duy nhất mà CPU có thể truy xuất trực tiếp được.

Các chương trình của người sử dụng muốn thực hiện được bởi CPU thì trước hết nó phải được hệ điều hành nạp vào bộ nhớ chính, chuyển đổi các địa chỉ sử dụng trong chương trình thành những địa chỉ mà CPU có thể truy xuất được.

Khi chương trình, tiến trình có yêu cầu được nạp vào bộ nhớ thì hệ điều hành phải cấp phát không gian nhớ cho nó. Khi chương trình, tiến

trình kết thúc thì hệ điều hành phải thu hồi lại không gian nhớ đã cấp phát cho chương trình, tiến trình trước đó.

Trong các hệ thống đa chương hay đa tiến trình, trong bộ nhớ tồn tại nhiều chương trình/ nhiều tiến trình, hệ điều hành phải thực hiện nhiệm vụ bảo vệ các vùng nhớ đã cấp phát cho các chương trình/ tiến trình, tránh sự vi phạm trên các vùng nhớ của nhau.

- Tóm lại, bộ phận quản lý bộ nhớ chính của hệ điều hành thực hiện những nhiệm vụ sau:

  Cấp phát, thu hồi vùng nhớ.

  Ghi nhận trạng thái bộ nhớ chính.

  Bảo vệ bộ nhớ.

  Quyết định tiến trình nào được nạp vào bộ nhớ.

1.1.3.Thành phần quản lý xuất/ nhập

Một trong những mục tiêu của hệ điều hành là giúp người sử dụng khai thác hệ thống máy tính dễ dàng và hiệu quả, do đó các thao tác trao đổi thông tin trên thiết bị xuất/ nhập phải *trong suốt* đối với người sử dụng.

Để thực hiện được điều này hệ điều hành phải tồn tại một bộ phận điều khiển thiết bị, bộ phận này phối hợp cùng CPU để quản lý sự hoạt động và trao đổi thông tin giữa hệ thống, chương trình người sử dụng và người sử dụng với các thiết bị xuất/ nhập.

Bộ phận điều khiển thiết bị thực hiện những nhiệm vụ sau:

Gởi mã lệnh điều khiển đến thiết bị: Hệ điều hành điều khiển các thiết bị bằng các mã điều khiển, do đó trước khi bắt đầu một quá trình trao đổi dữ liệu với thiết bị thì hệ điều hành phải gởi mã điều khiển đến thiết bị.

Tiếp nhận yêu cầu ngắt (Interrupt) từ các thiết bị: Các thiết bị khi cần trao đổi với hệ thống thì nó phát ra một tín hiệu yêu cầu ngắt, hệ điều hành tiếp nhận yêu cầu ngắt từ các thiết bị, xem xét và thực hiện một thủ tục để đáp ứng yêu cầu tù các thiết bị.

Phát hiện và xử lý lỗi: quá trình trao đổi dữ liệu thường xảy ra các lỗi như: thiết bị vào ra chưa sẵn sàng, đường truyền hỏng, ... do đó hệ điều hành phải tạo ra các cơ chế thích hợp để phát hiện lỗi sớm nhất và khắc phục các lỗi vừa xảy ra nếu có thể.

1.1.4.Thành phần quản lý bộ nhớ phụ (đĩa)

Không gian lưu trữ của đĩa được chia thành các phần có kích thước bằng nhau được gọi là các block, khi cần lưu trữ một tập tin trên đĩa hệ điều hành sẽ cấp cho tập tin một lượng vừa đủ các block để chứa hết nội dung của tập tin. Block cấp cho tập tin phải là các block còn tự do, chưa cấp cho các tập tin trước đó, do đó sau khi thực hiện một thao tác cấp phát block hệ điều hành phải ghi nhận trạng thái của các block trên đĩa, đặc biệt

là các block còn tự do để chuẩn bị cho các quá trình cấp block sau này.

Trong quá trình sử dụng tập tin nội dung của tập tin có thể thay đổi (tăng, giảm), do đó hệ điều hành phải tổ chức cấp phát động các block cho tập tin.

Để ghi/đọc nội dung của một block thì trước hết phải định vị đầu đọc/ ghi đến block đó. Khi chương trình của người sử dụng cần đọc nội dung của một đãy các block không liên tiếp nhau, thì hệ điều hành phải chọn lựa nên đọc block nào trước, nên đọc theo thứ tự nào,..., dựa vào đó mà hệ điều hành di chuyển đầu đọc đến các block thích hợp, nhằm nâng cao tốc độ đọc dữ liệu trên đĩa. Thao tác trên được gọi là lập lịch cho đĩa.

- Tóm lại, bộ phận quản lý bộ nhớ phụ thực hiện những nhiệm vụ sau:

  Quản lý không gian trống trên đĩa.

  Định vị lưu trữ thông tin trên đĩa.

  Lập lịch cho vấn đề ghi/ đọc thông tin trên đĩa của đầu từ.

1.1.5.Thành phần quản lý tập tin

Máy tính có thể lưu trữ thông tin trên nhiều loại thiết bị lưu trữ khác nhau, mỗi thiết bị lại có tính chất và cơ chế tổ chức lưu trữ thông tin khác nhau, điều này gây khó khăn cho người sử dụng. Để khắc phục điều này hệ điều hành đưa ra khái niệm đồng nhất cho tất cả các thiết bị lưu trữ vật lý, đó là *tập tin* (file).

Tập tin là đơn vị lưu trữ cơ bản nhất, mỗi tập tin có một tên riêng. Hệ điều hành phải thiết lập mối quan hệ tương ứng giữa tên tập tin và thiết bị lưu trữ chứa tập tin. Theo đó khi cần truy xuất đến thông tin đang lưu trữ trên bất kỳ thiết bị lưu trữ nào người sử dụng chỉ cần truy xuất đến tập tin tương ứng thông qua tên của nó, tất cả mọi việc còn lại đều do hệ điều hành thực hiện.

Trong hệ thống có nhiều tiến trình đồng thời truy xuất tập tin hệ điều hành phải tạo ra những cơ chế thích hợp để bảo vệ tập tin trách việc ghi/ đọc bất hợp lệ trên tập tin.

- Tóm lại: Như vậy bộ phận quản lý tập tin của hệ điều hành thực hiện những nhiệm vụ sau:

  Tạo/ xoá một tập tin/ thư mục.

  Bảo vệ tập tin khi có hiện tượng truy xuất đồng thời.

  Cung cấp các thao tác xử lý và bảo vệ tập tin/ thư mục.

  Tạo mối quan hệ giữa tập tin và bộ nhớ phụ chứa tập tin.

  Tạo cơ chế truy xuất tập tin thông qua tên tập tin.

1.1.6.Thành phần mạng

Hệ phân tán là tập hợp các bộ xử lý, chúng không chia sẻ bộ nhớ, các thiết bị ngoại vi hay đồng hồ. Thay vào đó mỗi bộ xử lý có bộ nhớ, đồng

hồ và các bộ xử lý giao tiếp với nhau thông qua các đường giao tiếp như bus tốc độ cao hay mạng. Các bộ xử lý trong hệ thống phân tán khác nhau về kích thước và chức năng. Chúng có thể chứa các bộ vi xử lý, trạm làm việc, máy vi tính và các hệ thống máy tính thông thường. Các bộ xử lý trong hệ thống được nối với nhau thông qua mạng truyền thông có thể được cấu hình trong nhiều cách khác nhau. Mạng có thể được nối kết một phần hay toàn bộ. Thiết kế mạng truyền thông phải xem xét vạch đường thông điệp và các chiến lược nối kết, và các vấn đề cạnh tranh hay bảo mật.

Hệ thống phân tán tập hợp những hệ thống vật lý riêng rẻ, có thể có kiến trúc không đồng nhất thành một hệ thống chặt chẻ, cung cấp người dùng với truy xuất tới các tài nguyên khác nhau mà hệ thống duy trì. Truy xuất tới các tài nguyên chia sẻ cho phép tăng tốc độ tính toán, chức năng, khả năng sẵn dùng của dữ liệu, khả năng tin cậy. Hệ điều hành thường tổng quát hoá việc truy xuất mạng như một dạng truy xuất tập tin, với những chi tiết mạng được chứa trong trình điều khiển thiết bị của giao diện mạng. Các giao thức tạo một hệ thống phân tán có thể có một ảnh hưởng to lớn trên tiện ích và tính phổ biến của hệ thống đó. Sự đổi mới của World Wide Web đã tạo ra một phương pháp truy xuất mới cho thông tin chia sẻ. Nó đã cải tiến giao thức truyền tập tin (File Transfer Protocol-FTP) và hệ thống tập tin mạng (Network File System-NFS) đã có bằng cách xoá yêu cầu cho một người dùng đăng nhập trước khi người dùng đó được phép dùng tài nguyên ở xa. Định nghĩa một giao thức mới, giao thức truyền siêu văn bản (hypertext transfer protocol-http), dùng trong giao tiếp giữa một trình phục vụ web và trình duyệt web. Trình duyệt web chỉ cần gởi yêu cầu thông tin tới một trình phục vụ web của máy ở xa, thông tin (văn bản, đồ hoạ, liên kết tới những thông tin khác) được trả về.

1.1.7.Thành phần thông dịch lệnh

Đây là bộ phận quan trọng của hệ điều hành, nó đóng vai trò giao tiếp giữa hệ điều hành và người sử dụng. Thành phần này chính là shell mà chúng ta đã biết ở trên. Một số hệ điều hành chứa shell trong nhân (kernel) của nó, một số hệ điều hành khác thì shell được thiết kế dưới dạng một chương trình đặc biệt.

1.1.8.Thành phần bảo vệ hệ thống

Trong môi trường hệ điều hành đa nhiệm có thể có nhiều tiến trình hoạt động đồng thời, thì mỗi tiến trình phải được bảo vệ để không bị tác động, có chủ ý hay không chủ ý, của các tiến trình khác. Trong trường hợp này hệ điều hành cần phải có các cơ chế để luôn đảm bảo rằng các File, Memory, CPU và các tài nguyên khác mà hệ điều hành đã cấp cho một chương trình, tiến trình thì chỉ có chương trình tiến trình đó được quyền tác động đến các thành phần này.

Nhiệm vụ trên thuộc thành phần bảo vệ hệ thống của hệ điều hành. Thành phần này điều khiển việc sử dụng tài nguyên, đặc biệt là các tài nguyên dùng chung, của các tiến trình, đặc biệt là các tiến trình hoạt động đồng thời với nhau, sao cho không xảy ra sự tranh chấp tài nguyên giữa các tiến trình hoạt đồng đồng thời và không cho phép các tiến trình truy xuất bất hợp lệ lên các vùng nhớ của nhau.

- Ngoài ra các hệ điều hành mạng, các hệ điều hành phân tán hiện nay còn có thêm thành phần kết nối mạng và truyền thông..
- Để đáp ứng yêu cầu của người sử dụng và chương trình người sử dụng các nhiệm vụ của hệ điều hành được thiết kế dưới dạng các dịch vụ:

Thi hành chương trình: hệ điều hành phải có nhiệm vụ nạp chương trình của người sử dụng vào bộ nhớ, chuẩn bị đầy đủ các điều kiện về tài nguyên để chương trình có thể chạy được và kết thúc được, có thể kết thúc bình thường hoặc kết thúc do bị lỗi. Khi chương trình kết thúc hệ điều hành phải thu hồi tài nguyên đã cấp cho chương trình và ghi lại các thông tin mà chương trình đã thay đổi trong quá trình chạy (nếu có).

Thực hiện các thao tác xuất nhập dữ liệu: Khi chương trình chạy nó có thể yêu cầu xuất nhập dữ liệu từ một tập tin hoặc từ một thiết bị xuất nhập nào đó, trong trường hợp này hệ điều hành phải hỗ trợ việc xuất nhập dữ liệu cho chương trình, phải nạp được dữ liệu mà chương trình cần vào bộ nhớ.

Thực hiện các thao tác trên hệ thống tập tin: Hệ điều hành cần cung cấp các công cụ để chương trình dễ dàng thực hiện các thao tác đọc ghi trên các tập tin, các thao tác này phải thực sự an toàn, đặc biệt là trong môi trường đa nhiệm.

Trao đổi thông tin giữa các tiến trình: Trong môi trường hệ điều hành đa nhiệm, với nhiều tiến trình hoạt động đồng thời với nhau, một tiến trình có thể trao đổi thông tin với nhiều tiến trình khác, hệ điều hành phải cung cấp các dịch vụ cần thiết để các tiến trình có thể trao đổi thông tin với nhau và phối hợp cùng nhau để hoàn thành một tác vụ nào đó.

Phát hiện và xử lý lỗi: Hệ điều hành phải có các công cụ để chính hệ điều hành và để hệ điều hành giúp chương trình của người sử dụng phát hiện các lỗi do hệ thống (CPU, Memory, I/O device, Program) phát sinh. Hệ điều hành cũng phải đưa ra các dịch vụ để xử lý các lỗi sao cho hiệu quả nhất.

1.2.Chức năng của hệ điều hành

Một hệ thống máy tính gồm 3 thành phần chính: phần cứng, hệ điều hành và các chương trình ứng dụng và người sử dụng. Trong đó hệ điều hành là một bộ phận quan trọng và không thể thiếu của hệ thống máy tính,

nhờ có hệ điều hành mà người sử dụng có thể đối thoại và khai thác được các chức năng của phần cứng máy tính.

Có thể nói hệ điều hành là một hệ thống các chương trình đóng vai trò trung gian giữa người sử dụng và phần cứng máy tính. Mục tiêu chính của nó là cung cấp một môi trường thuận lợi để người sử dụng dễ dàng thực hiện các chương trình ứng dụng của họ trên máy tính và khai thác triệt để các chức năng của phần cứng máy tính.

Để đạt được mục tiêu trên hệ điều hành phải thực hiện 2 chức năng chính sau đây:

- **Giả lập một máy tính mở rộng**: Máy tính là một thiết bị vi điện tử, nó được cấu thành từ các bộ phận như: Processor, Memory, I/O Device, Bus, ... , do đó để đối thoại hoặc khai thác máy tính người sử dụng phải hiểu được cơ chế hoạt động của các bộ phận này và phải tác động trực tiếp vào nó, tất nhiên là bằng những con số 0,1 (ngôn ngữ máy). Điều này là quá khó đối với người sử dụng. Để đơn giản cho người sử dụng hệ điều hành phải che đậy các chi tiết phần cứng máy tính bởi một máy tính mở rộng, máy tính mở rộng này có đầy đủ các chức năng của một máy tính thực nhưng đơn giản và dễ sử dụng hơn. Theo đó khi cần tác động vào máy tính thực người sử dụng chỉ cần tác động vào máy tính mở rộng, mọi sự chuyển đổi thông tin điều khiển từ máy tính mở rộng sang máy tính thực hoặc ngược lại đều do hệ điều hành thực hiện. Mục đích của chức năng này là: *Giúp người sử dụng khai thác các chức năng của phần cứng máy tính dễ dàng và hiệu quả hơn.*
- **Quản lý tài nguyên của hệ thống**: Tài nguyên hệ thống có thể là: processor, memory, I/O device, printer, file, ..., đây là những tài nguyên mà hệ điều hành dùng để cấp phát cho các tiến trình, chương trình trong quá trình điều khiển sự hoạt động của hệ thống. Khi người sử dụng cần thực hiện một chương trình hay khi một chương trình cần nạp thêm một tiến trình mới vào bộ nhớ thì hệ điều hành phải cấp phát không gian nhớ cho chương trình, tiến trình đó để chương trình, tiến trình đó nạp được vào bộ nhớ và hoạt động được. Trong môi trường hệ điều hành đa nhiệm có thể có nhiều chương trình, tiến trình đồng thời cần được nạp vào bộ nhớ, nhưng không gian lưu trữ của bộ nhớ có giới hạn, do đó hệ điều hành phải tổ chức cấp phát bộ nhớ sao cho hợp lý để đảm bảo tất cả các chương trình, tiến trình khi cần đều được nạp vào bộ nhớ để hoạt động. Ngoài ra hệ điều hành còn phải tổ chức bảo vệ các không gian nhớ đã cấp cho các chương trình, tiến trình để tránh sự truy cập bất hợp lệ và sự tranh chấp bộ nhớ giữa các chương trình, tiến trình, đặc biệt là các tiến trình đồng thời hoạt động trên hệ thống. Đây là một trong những nhiệm vụ quan trọng của hệ điều hành.

Trong quá trình hoạt động của hệ thống, đặc biệt là các hệ thống đa

người dùng, đa chương trình, đa tiến trình, còn xuất hiện một hiện tượng khác, đó là nhiều chương trình, tiến trình đồng thời sử dụng một không gian nhớ hay một tập tin (dữ liệu, chương trình) nào đó. Trong trường hợp này hệ điều hành phải tổ chức việc chia sẻ và giám sát việc truy xuất đồng thời trên các tài nguyên nói trên sao cho việc sử dụng tài nguyên có hiệu quả nhưng tránh được sự mất mát dữ liệu và làm hỏng các tập tin.

Trên đây là hai dẫn chứng điển hình để chúng ta thấy vai trò của hệ điều hành trong việc quản lý tài nguyên hệ thống, sau này chúng ta sẽ thấy việc cấp phát, chia sẻ, bảo vệ tài nguyên của hệ điều hành là một trong những công việc khó khăn và phức tạp nhất. Hệ điều hành đã chi phí nhiều cho công việc nói trên để đạt được mục tiêu: *Trong mọi trường hợp tất cả các chương trình, tiến trình nếu cần được cấp phát tài nguyên để hoạt động thì sớm hay muộn nó đều được cấp phát và được đưa vào trạng thái hoạt động.*

➢ Trên đây là hai chức năng tổng quát của một hệ điều hành, đó cũng được xem như là các mục tiêu mà các nhà thiết kế, cài đặt hệ điều hành phải hướng tới. Các hệ điều hành hiện nay có các chức năng cụ thể sau đây:

❑ Hệ điều hành cho phép thực hiện nhiều chương trình đồng thời trong môi trường đa tác vụ - **Multitasking Environment**. Hệ điều hành multitasking bao gồm: Windows NT, Windows 2000, Linux và OS/2. Trong hệ thống multasking hệ điều hành phải xác định khi nào thì một ứng dụng được chạy và mỗi ứng dụng được chạy trong khoản thời gian bao lâu thì phải dừng lại để cho các ứng dụng khác được chạy.

❑ Hệ điều hành tự nạp nó vào bộ nhớ - **It loads itself into memory**: Quá trình nạp hệ điều hành vào bộ nhớ được gọi là quá trình **Booting**. Chỉ khi nào hệ điều hành đã được nạp vào bộ nhớ thì nó mới cho phép người sử dụng giao tiếp với phần cứng. Trong các hệ thống có nhiều ứng dụng đồng thời hoạt động trên bộ nhớ thì hệ điều hành phải chịu trách nhiệm chia sẻ không gian bộ nhớ RAM và bộ nhớ cache cho các ứng dụng này.

❑ **Hệ điều hành và API: Application Programming Interface**: API là một tập các hàm/thủ tục được xây dựng sẵn bên trong hệ thống, nó có thể thực hiện được nhiều chức năng khác nhau như shutdown hệ thống, đảo ngược hiệu ứng màn hình, khởi động các ứng dụng, … Hệ điều hành giúp cho chương trình của người sử dụng giao tiếp với API hay thực hiện một lời gọi đến các hàm/thủ tục của API. Nạp dữ liệu cần thiết vào bộ nhớ - **It loads the requied data into memory**: Dữ liệu do người sử dụng cung cấp được đưa vào bộ nhớ để xử lý. Khi nạp dữ liệu vào bộ nhớ hệ điều hành phải lưu lại địa chỉ của bộ nhớ nơi mà dữ liệu được lưu ở đó. Hệ điều hành phải luôn theo dõi bản đồ cấp phát bộ nhớ, nơi dữ liệu và

chương trình được lưu trữ ở đó. Khi một chương trình cần đọc dữ liệu, hệ điều hành sẽ đến các địa chỉ bộ nhớ nơi đang lưu trữ dữ liệu mà chương trình cần đọc để đọc lại nó.

❑ Hệ điều hành biên dịch các chỉ thị chương trình - **It interprets program instructions**: Hệ điều hành phải đọc và giải mã các thao tác cần được thực hiện, nó được viết trong chương trình của người sử dụng. Hệ điều hành cũng chịu trách nhiệm sinh ra thông báo lỗi khi hệ thống gặp lỗi trong khi đang hoạt động.

❑ Hệ điều hành quản lý tài nguyên - **It managers resources**: Nó đảm bảo việc sử dụng thích hợp tất cả các tài nguyên của hệ thống như là: bộ nhớ, đĩa cứng, máy in, …

1.3.Quá trình phát triển 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. Điển hình là các giai đoạn sau:

Monitor

Interrupt and trap vectors
Device drivers
Job sequencing
Control card interpreter
User program area

Hình 1.1 – Cấu trúc monitor đơn giản

- Monitor đơn giản: Đây là hệ điều hành đầu tiên có thể tự động hóa, 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 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 programing 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 bố 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 bố 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 cảu các thành phần hệ thống.

2. Phân loại hệ điều hành

Mục Tiêu:

- Nắm được khái niệm phân loại và các thành phần cơ bản trong hệ điều hành

2.1. Mainframe Systems

Những hệ thống máy tính mainframe là những máy tính đầu tiên được dùng để xử lý ứng dụng thương mại và khoa học. Trong phần này, chúng ta lần theo sự phát triển của hệ thống mainframe từ các hệ thống bó (batch systems), ở đó máy tính chỉ chạy một-và chỉ một -ứng dụng, tới các hệ chia sẻ thời gian (time-shared systems), mà cho phép người dùng giao tiếp với hệ thống máy tính

2.2.Desktop Systems

Máy tính cá nhân (PC) xuất hiện vào những năm 1970. Trong suốt thập niên đầu, CPU trong PC thiếu các đặc điểm cần thiết để bảo vệ hệ điều hành từ chương trình người dùng. Do đó, các hệ điều hành PC không

là đa người dùng hoặc đa nhiệm. Tuy nhiên, các mục tiêu của hệ điều hành này thay đổi theo thời gian; thay vì tối ưu hoá việc sử dụng CPU và thiết bị ngoại vi, các hệ thống chọn lựa tối ưu hoá sự tiện dụng và đáp ứng người dùng. Các hệ thống này gồm các PC chạy các hệ điều hành Microsoft Windows và Apple Macintosh. Hệ điều hành MS-DOS từ Microsoft được thay thế bằng nhiều ấn bản của Microsoft Windows và IBM đã nâng cấp MS-DOS thành hệ đa nhiệm OS/2. Hệ điều hành Apple Macintosh được gắn nhiều phần cứng hiện đại hơn và ngày nay chứa nhiều đặc điểm mới như bộ nhớ ảo và đa nhiệm.

Với sự phát hành MacOS X, lõi của hệ điều hành ngày nay dựa trên Mach và FreeBSD UNIX cho sự mở rộng, năng lực và đặc điểm nhưng nó vẫn giữ lại giao diện đồ hoạ người dùng GUI. LINUX, một hệ điều hành tương tự như UNIX sẵn dùng cho máy PC trở nên phổ biến gần đây.

Hệ điều hành cho các máy tính này có những thuận lợi trong nhiều cách từ sự phát triển của hệ điều hành cho mainframes. Máy vi tính (microcomputer) lập tức có thể được chấp nhận một số công nghệ được phát triển cho hệ điều hành lớn hơn. Thêm vào đó, chi phí phần cứng cho máy vi tính đủ thấp để các cá nhân có thể một mình sử dụng máy tính, và sử dụng CPU không còn quan trọng nữa. Do đó, những quyết định thiết kế được thực hiện trong hệ điều hành cho mainframes có thể không hợp lý cho các hệ thống nhỏ hơn.

Những quyết định thiết kế khác vẫn được áp dụng. Thí dụ, trước hết bảo vệ hệ thống tập tin không cần thiết trên máy cá nhân. Tuy nhiên, hiện nay các máy tính này thường được nối vào các máy tính khác qua mạng cục bộ hay Internet. Khi những máy tính khác và người dùng khác có thể truy xuất các tập tin này trên một PC, bảo vệ tập tin một lần nữa cũng trở thành một đặc điểm cần thiết của hệ điều hành. Sự thiếu bảo vệ tạo điều kiện dễ dàng cho những chương trình hiểm phá huỷ dữ liệu trên những hệ thống như MS-DOS và hệ điều hành Macintosh. Các chương trình này có thể tự nhân bản và phát tán nhanh chóng bằng cơ chế worm hay virus và làm tê liệt mạng của các công ty hay thậm chí mạng toàn cầu. Đặc điểm chia thời được cải tiến như bộ nhớ bảo vệ và quyền tập tin là chưa đủ để bảo vệ một hệ thống từ sự tấn công.

2.3.Multiprocessor Systems

Hầu hết các hệ thống ngày nay là các hệ thống đơn xử lý; nghĩa là chỉ có một CPU chính. Tuy nhiên, các hệ thống đa xử lý (hay còn gọi là hệ song song hay hệ kết nối chặt) được phát triển rất quan trọng. Các hệ thống như thế có nhiều hơn một bộ xử lý trong giao tiếp gần, chia sẻ bus máy tính, đồng hồ, đôi khi còn là bộ nhớ hay thiết bị ngoại vi.

Hệ thống đa xử lý có ba ưu điểm chính:

- **Thông lượng được gia tăng**: bằng cách tăng số lượng bộ xử lý,

chúng ta hy vọng thực hiện nhiều công việc hơn với thời gian ít hơn. Tỉ lệ giữa sự tăng tốc với N bộ xử lý không là N; đúng hơn nó nhỏ hơn N. Trong khi nhiều bộ xử lý cộng tác trên một công việc, một lượng chi phí phải chịu trong việc giữ các thành phần làm việc phù hợp. Chi phí này cộng với chi phí cạnh tranh tài nguyên được chia sẻ, làm giảm kết quả được mong đợi từ những bộ xử lý bổ sung. Tương tự như một nhóm gồm N lập trình viên làm việc với nhau không dẫn đến kết quả công việc đang đạt được tăng N lần.

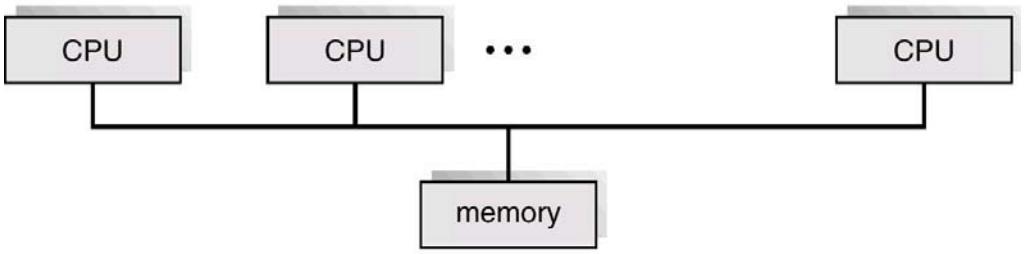
- **Tính kinh tế của việc mở rộng**: hệ thống đa xử lý có thể tiết kiệm nhiều chi phí hơn hệ thống đơn bộ xử lý, bởi vì chúng có thể chia sẻ ngoại vi, thiết bị lưu trữ và điện. Nếu nhiều chương trình điều hành trên cùng tập hợp dữ liệu thì lưu trữ dữ liệu đó trên một đĩa và tất cả bộ xử lý chia sẻ chúng sẽ rẻ hơn là có nhiều máy tính với đĩa cục bộ và nhiều bản sao dữ liệu.
- **Khả năng tin cậy được gia tăng**: nếu các chức năng được phân bổ hợp lý giữa các bộ xử lý thì lỗi trên một bộ xử lý sẽ không dừng hệ thống, chỉ năng lực bị giảm. Nếu chúng ta có 10 bộ xử lý và có 1 bộ xử lý bị sự cố thì mỗi bộ xử lý trong 9 bộ xử lý còn lại phải chia sẻ của công việc của bộ xử lý bị lỗi. Do đó, toàn bộ hệ thống chỉ giảm 10% năng lực hơn là dừng hoạt động. Các hệ thống được thiết kế như thế được gọi là hệ thống có khả năng chịu lỗi (fault tolerant).

Việc điều hành vẫn tiếp tục trong sự hiện diện của lỗi yêu cầu một cơ chế cho phép lỗi được phát hiện, chuẩn đoán và sửa lỗi nếu có thể. Hệ thống Tandem sử dụng sự nhân đôi phần cứng và phần mềm để đảm bảo sự điều hành vẫn tiếp tục mặc dù có lỗi xảy ra. Hệ thống này chứa hai bộ xử lý, mỗi bộ xử lý có bộ nhớ cục bộ riêng. Các bộ xử lý được nối kết bởi một bus. Một bộ xử lý chính và bộ xử lý kia là dự phòng. Cả hai bản sao được giữ ở mỗi bộ xử lý: một là chính và một là dự phòng. Tại các điểm kiểm tra (checkpoints) trong việc thực thi của hệ thống, thông tin trạng thái của mỗi công việc-gồm một bản sao hình ảnh bộ nhớ-được chép từ máy chính tới máy dự phòng. Nếu một lỗi được phát hiện, bản sao dự phòng được kích hoạt và được khởi động lại từ điểm kiểm tra mới nhất. Giải pháp này đắt vì nó bao gồm việc nhân đôi phần cứng.

Các hệ thống đa xử lý thông dụng nhất hiện nay sử dụng **đa xử lý đối xứng** (symmetric multiprocessing-SMP). Trong hệ thống này mỗi bộ xử lý chạy bản sao của hệ điều hành và những bản sao này giao tiếp với các bản sao khác khi cần. Vài hệ thống sử dụng **đa xử lý bất đối xứng** (asymmetric multiprocessing). Trong hệ thống này mỗi bộ xử lý được gán một công việc xác định. Một bộ xử lý chủ điều khiển hệ thống; những bộ xử lý còn lại hoặc chờ bộ xử lý chủ ra chỉ thị hoặc có những tác vụ được định nghĩa trước. Cơ chế này định nghĩa mối quan hệ chủ-tớ. Bộ xử lý

chính lập thời biểu và cấp phát công việc tới các bộ xử lý tớ.

Đa xử lý đối xứng có nghĩa tất cả bộ xử lý là ngang hàng; không có mối quan hệ chủ-tớ tồn tại giữa các bộ xử lý. Hình I-4 minh hoạ một kiến trúc đa xử lý đối xứng điển hình. Một thí dụ của đa xử lý đối xứng là ấn bản của Encore của UNIX cho máy tính Multimax. Máy tính này có thể được cấu hình như nó đang thực hiện nhiều bộ xử lý, tất cả bộ xử lý đều chạy bản sao của UNIX. Ưu điểm của mô hình này là nhiều quá trình có thể chạy cùng một lúc - N quá trình có thể chạy nếu có N CPU- không gây ra sự giảm sút to lớn về năng lực. Tuy nhiên, chúng ta phải điều khiển cẩn thận xuất/nhập để đảm bảo rằng dữ liệu dẫn tới bộ xử lý tương ứng. Vì các CPU là riêng rẽ, một CPU có thể đang rảnh trong khi CPU khác quá tải dẫn đến việc sử dụng không hữu hiệu tài nguyên của hệ thống. Sự không hiệu quả này có thể tránh được nếu các bộ xử lý chia sẻ các cấu trúc dữ liệu. Một hệ thống đa xử lý của dạng này sẽ cho phép các quá trình và tài nguyên – như bộ nhớ - được chia sẻ tự động giữa các quá trình khác nhau và có thể làm giảm sự khác biệt giữa các bộ xử lý. Hầu như tất cả hệ điều hành hiện đại - gồm Windows NT, Solaris, Digital UNIX, OS/2 và LINUX - hiện nay cung cấp sự hỗ trợ đa xử lý đối xứng.



Hình 1.2 Kiến trúc đa xử lý đối xứng

Sự khác biệt giữa đa xử lý đối xứng và bất đối xứng có thể là do phần cứng hoặc phần mềm. Phần cứng đặc biệt có thể khác nhau trên nhiều bộ xử lý, hoặc phần mềm có thể được viết để cho phép chỉ một chủ và nhiều tớ. Thí dụ, SunOS ấn bản 4 cung cấp đa xử lý không đối xứng, ngược lại, ấn bản 5 (Solaris 2) là đối xứng trên cùng phần cứng.

Khi các bộ vi xử lý trở nên rẻ hơn và mạnh hơn các chức năng bổ sung của hệ điều hành là chuyển tới bộ xử lý tớ. Thí dụ, tương đối dễ để thêm bộ vi xử lý với bộ nhớ riêng để quản lý hệ thống đĩa. Bộ vi xử lý có thể nhận một chuỗi các yêu cầu từ bộ nhớ chính và cài đặt hàng đợi đĩa riêng và giải thuật định thời. Sự sắp xếp này làm giảm chi phí định thời đĩa của CPU. PC chứa một bộ vi xử lý trong bàn phím để chuyển những phím nóng thành mã để gởi tới CPU. Thực tế, việc sử dụng các bộ vi xử lý trở nên quá phổ biến đến nỗi mà đa xử lý không còn được xem xét.

2.4.Distributed Systems

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 cường 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.

2.5.Real-Time Systems

Một dạng khác của hệ điều hành có mục đích đặc biệt là hệ thời thực (real-time system). Hệ thời thực được dùng khi các yêu cầu thời gian khắt khe được đặt trên thao tác của một bộ xử lý hay dòng dữ liệu; do đó, nó thường được dùng như một thiết bị điều khiển trong một ứng dụng tận hiến. Các bộ cảm biến mang dữ liệu tới máy tính. Máy tính phải phân tích dữ liệu và có thể thích ứng các điều khiển để hiệu chỉnh các dữ liệu nhập cảm biến. Các hệ thống điều khiển các thí nghiệm khoa học, hệ thống ảnh hoá y tế, hệ thống điều khiển công nghệ và các hệ thống hiển thị,... Các hệ thống phun dầu động cơ ôtô, các bộ điều khiển dụng cụ trong nhà, hệ thống vũ khí cũng là các hệ thống thời thực.

Một hệ thống thời thực có sự ràng buộc cố định, rõ ràng. Xử lý phải được thực hiện trong phạm vi các ràng buộc được định nghĩa hay hệ thống sẽ thất bại. Một hệ thời thực thực hiện đúng chức năng chỉ nếu nó trả về kết quả đúng trong thời gian ràng buộc. Tương phản với yêu cầu này trong hệ chia thời, ở đó nó mong muốn (nhưng không bắt buộc) đáp ứng nhanh, hay đối với hệ thống bó, nó không có ràng buộc thời gian gì cả.

Hệ thời thực có hai dạng: cứng và mềm. Hệ thời thực cứng đảm bảo rằng các tác vụ tới hạn được hoàn thành đúng giờ. Mục tiêu này đòi hỏi tất cả trì hoãn trong hệ thống bị giới hạn, từ việc lấy lại dữ liệu được lưu trữ thời gian hệ điều hành hoàn thành bất cứ yêu cầu cho nó. Các ràng buộc thời gian như thế ra lệnh các phương tiện sẵn có trong hệ thời thực cứng. Thiết bị lưu trữ phụ của bất cứ thứ hạng nào thường bị giới hạn hay bị mất với dữ liệu đang được lưu trong bộ nhớ lưu trữ ngắn hạn (short- term memory) hay trong bộ nhớ chỉ đọc (ROM). Hầu hết các hệ điều hành hiện đại không cung cấp đặc điểm này vì chúng có khuynh hướng tách rời người dùng từ phần cứng và sự tách rời này dẫn đến lượng thời gian không xác định mà thao tác sẽ mất. Thí dụ, bộ nhớ ảo hầu như chưa bao giờ thấy trong hệ thời thực. Do đó, những hệ thời thực cứng xung đột với thao tác của hệ chia thời và hai hệ này không thể đan xen nhau. Vì không có hệ điều hành đa mục đích đã có hỗ trợ chức năng thời thực cứng; chúng ta không

tập trung với loại hệ thống này trong chương này.

Một loại thời thực ít hạn chế hơn là hệ thời thực mềm, ở đó tác vụ thời thực tới hạn có độ ưu tiên hơn các tác vụ khác và duy trì độ ưu tiên đó cho đến khi chúng hoàn thành. Như trong hệ thời thực cứng, sự trì hoãn nhân (kernel) của hệ điều hành trì hoãn yêu cầu được giới hạn. Một tác vụ thời thực không thể giữ việc chờ không xác định đối với nhân để thực thi. Thời thực mềm là mục tiêu có thể đạt được và có thể được đan xen với các loại hệ thống khác. Tuy nhiên, hệ thời thực mềm có những tiện ích giới hạn hơn hệ thời thực cứng. Vì không hỗ trợ tốt cho thời điểm tới hạn, nên hệ thời thực mềm dễ gây rủi ro khi dùng cho việc kiểm soát công nghệ và tự động hoá. Tuy nhiên, chúng có ích trong nhiều lĩnh vực như đa phương tiện, thực tế ảo, dự án khoa học tiên tiến-như khám phá trong lòng đại dương và khám phá hành tinh. Những hệ thống này cần những đặc điểm hệ điều hành tiên tiến mà không được hỗ trợ bởi hệ thời thực cứng. Vì việc sử dụng chức năng thời thực mềm được mở rộng nên chúng ta đang tìm cách đưa chúng vào trong hầu hết các hệ điều hành hiện tại, gồm các ấn bản chính thức của UNIX.

2.6.Handheld Systems

Hệ xách tay gồm các máy hỗ trợ cá nhân dùng kỹ thuật số (personal digital assistants-PDAs) như Palm hay điện thoại di động (cellular telephone) với nối kết tới mạng như Internet. Những người phát triển hệ xách tay và ứng dụng gặp phải nhiều thử thách, nhất là sự giới hạn về kích thước của thiết bị. Thí dụ, một PDA điển hình cao khoảng 5 inches và rộng khoảng 3 inches và trọng lượng của nó ít hơn 0.5 pound. Do sự giới hạn về kích thước này, hầu hết các thiết bị xách tay có bộ nhớ nhỏ gồm các bộ xử lý thấp và màn hình hiển thị nhỏ. Bây giờ chúng ta sẽ xem xét mỗi sự giới hạn này.

Nhiều thiết bị xách tay có dung lượng bộ nhớ 512KB và 8 MB (ngược lại, các máy PC hay trạm làm việc có hàng trăm MB bộ nhớ). Do đó, hệ điều hành và các ứng dụng phải quản lý bộ nhớ hiệu quả. Điều này gồm trả về tất cả bộ nhớ được cấp phát tới bộ quản lý bộ nhớ một khi bộ nhớ không còn được dùng nữa. Hiện nay, nhiều thiết bị xách tay không dùng kỹ thuật bộ nhớ ảo do đó buộc người phát triển chương trình làm việc trong phạm vi giới hạn của bộ nhớ vật lý.

Vấn đề thứ hai quan tâm đến người phát triển các thiết bị xách tay là tốc độ của bộ xử lý được dùng trong thiết bị. Các bộ xử lý đối với hầu hết các thiết bị xách tay thường chạy với tốc độ chỉ bằng một phần tốc độ của một bộ xử lý trong máy PC. Các bộ xử lý nhanh hơn yêu cầu điện năng nhiều hơn. Để chứa một bộ xử lý nhanh hơn bên trong thiết bị xách tay nên yêu cầu nhiều pin hơn hơn và phải được nạp lại thường xuyên. Để tối thiểu hoá kích thước của các thiết bị xách tay đòi hỏi bộ xử lý nhỏ hơn,

chậm hơn tiêu thụ ít điện năng hơn. Do đó, hệ điều hành và các ứng dụng phải được thiết kế không đòi hỏi sử dụng nhiều bộ xử lý.

Vấn đề cuối cùng gây khó khăn cho người thiết kế chương trình cho các thiết bị xách tay là màn hình hiển thị nhỏ. Trong khi một màn hình cho máy tính ở nhà kích thước có thể 21 inches, màn hình cho thiết bị xách tay thường có diện tích không quá 3 inches. Những tác vụ quen thuộc như đọc e-mail hay hiển thị các trang web, phải được cô đọng vào màn hình nhỏ hơn. Một phương pháp để hiển thị nội dung các trang web là cắt xén web (web clipping), ở đó chỉ một tập hợp nhỏ trang web được phân phát và hiển thị trên thiết bị xách tay.

Một số thiết bị xách tay có thể dùng công nghệ không dây như BlueTooth, cho phép truy xuất từ xa tới e-mail và trình duyệt web. Các điện thoại di động với nối kết Internet thuộc loại này. Tuy nhiên, nhiều PDAs hiện tại không cung cấp truy xuất không dây. Để tải dữ liệu xuống các thiết bị này, trước tiên người dùng tải dữ liệu xuống PC hay trạm và sau đó tải dữ liệu xuống PDA. Một số PDA cho phép dữ liệu chép trực tiếp từ một thiết bị này tới thiết bị khác dùng liên kết hồng ngoại. Nhìn chung, các giới hạn trong chức năng của PDA được cân bằng bởi những tiện dụng và linh động của chúng. Việc sử dụng chúng tiếp tục mở rộng khi các nối kết mạng trở nên sẵn dùng và các chọn lựa khác như máy ảnh và MP3 players, mở rộng tiện ích của chúng.

2.7.Gaming Systems

Công nghệ Intel đang ngaỳ caàng mang lại nhiều trải nghiệm hấp dẫn hơn cho haàng loạt caác phần cứng maý tính. Vơí khả năng chơi game biết thích ứng theo nhu câù của bạn, biết tự động tăng vọt tôác độ xử lýkhi bạn cần vaè ép xung để đạt được tần sôślaàm việc liên tục cao hơn khi bạn muốn. Caác hệ thống chơi game chạy bộ xử ly Intel® Core™ i5 va Intel Core i7 thông minh trông thấy coómột bộ caác tính năng vêèkhả năng xử lyóvaèđôè họa tích hợp để đạt được nhưng khả năng thực têókhoótưởng tượng.

2.8.IOS (Internetwork Operating System)

Thông tin là một tài sản chiến lược có thể thực hiện hoặc phá vỡ tài sản của một công ty trong nền kinh tế toàn cầu hiện nay. Các mạng máy tính là những đường cao tốc điện tử khi mà thông tin đi và họ thống nhất thế giới để tạo ra những cách thức mới và tốt hơn để làm kinh doanh.

Với rất nhiều rủi ro, liên mạng của một tổ chức phải có khả năng tăng năng suất tổng thể của con người và nguồn lực của mình. Để làm điều này, nó có thể tối đa hóa sự sẵn có của các ứng dụng trong khi nó giảm thiểu tổng chi phí sở hữu. Điều này có nghĩa là cung cấp cho người sử dụng truy cập liên tục vào một mạng linh hoạt và đáng tin cậy. Nó cũng có

nghĩa là giữ trong kiểm tra các chi phí mà một tổ chức phải hấp thụ theo thời gian để phát triển và duy trì hệ thống thông tin và dịch vụ của mình.

Không có công ty trên thế giới có thể phù hợp với Systems khi nói đến tối đa hóa các ứng dụng có sẵn của một liên mạng và giảm thiểu tổng chi phí sở hữu. Trong thập kỷ qua, công nghệ đã được chứng minh của chúng tôi và nhiều giải pháp khả năng mở rộng đã giúp chúng tôi thiết lập tốc độ trong các nền công nghiệp mạng. Hơn bất cứ điều gì khác, vị trí lãnh đạo của mình để Internetwork Operating System độc đáo và mạnh mẽ của (IOS). IOS là phần mềm giá trị gia tăng mà nằm ở trung tâm của tất cả các giải pháp của nối mạng.

IOS là chìa khóa để giúp làm cho các công ty thông tin chuyên sâu tất cả các nơi trên thế giới hiệu quả hơn. Và cuối cùng, đó là lợi ích lớn nhất mà bất kỳ liên mạng có thể cung cấp.

3.Sơ lược lịch sử phát triển của hệ điều hành

Mục tiêu:

- Nắm được sơ lược về lịch sử phát triển của hệ điều hành

3.1.Lịch sử phát triển

3.1.1.Thế hệ 1 (1945 - 1955):

Vào những năm 1950 máy tính dùng ống chân không ra đời. Ở thế hệ này mỗi máy tính được một nhóm người thực hiện, bao gồm việc thiết kế, xây dựng chương trình, thao tác, quản lý, ...

Ở thế hệ này người lập trình phải dùng ngôn ngữ máy tuyệt đối để lập trình. Khái niệm ngôn ngữ lập trình và hệ điều hành chưa được biết đến trong khoảng thời gian này.

3.1.2Thế hệ 2 (1955 - 1965):

Máy tính dùng bán dẫn ra đời, và được sản xuất để cung cấp cho khách hàng. Bộ phận sử dụng máy tính được phân chia rõ ràng: người thiết kế, người xây dựng, người vận hành, người lập trình, và người bảo trì. Ngôn ngữ lập trình Assembly và Fortran ra đời trong thời kỳ này. Với các máy tính thế hệ này để thực hiện một thao tác, lập trình viên dùng Assembly hoặc Fortran để viết chương trình trên phiếu đục lỗ sau đó đưa phiếu vào máy, máy thực hiện cho kết qủa ở máy in.

Hệ thống xử lý theo lô cũng ra đời trong thời kỳ này. Theo đó, các thao tác cần thực hiện trên máy tính được ghi trước trên băng từ, hệ thống sẽ đọc băng từ , thực hiện lần lượt và cho kết quả ở băng từ xuất. Hệ thống xử lý theo lô hoạt động dưới sự điều khiển của một chương trình đặc biệt, chương trình này là hệ điều hành sau này.

3.1.3Thế hệ 3 (1965 - 1980)

Máy IBM 360 được sản xuất hàng loạt để tung ra thị trường. Các thiết bị ngoại vi xuất hiện ngày càng nhiều, do đó các thao tác điều khiển máy tính và thiết bị ngoại vi ngày càng phức tạp hơn. Trước tình hình này

nhu cầu cần có một hệ điều hành sử dụng chung trên tất cả các máy tính của nhà sản xuất và người sử dụng trở nên bức thiết hơn. Và hệ điều hành đã ra đời trong thời kỳ này.

Hệ điều hành ra đời nhằm điều phối, kiểm soát hoạt động của hệ thống và giải quyết các yêu cầu tranh chấp thiết bị. Hệ điều hành đầu tiên được viết bằng ngôn ngữ Assembly. Hệ điều hành xuất hiện khái niệm đa chương, khái niệm chia sẻ thời gian và kỹ thuật Spool. Trong giai đoạn này cũng xuất hiện các hệ điều hành Multics và Unix.

3.1.4.Thế hệ 4 (từ 1980-nay)

Máy tính cá nhân ra đời. Hệ điều hành MS_DOS ra đời gắn liền với máy tính IBM_PC. Hệ điều hành mạng và hệ điều hành phân tán ra đời trong thời kỳ này.

- Trên đây chúng tôi không có ý định trình bày chi tiết, đầy đủ về lịch sử hình thành của hệ điều hành, mà chúng tôi chỉ muốn mượn các mốc thời gian về sự ra đời của các thế hệ máy tính để chỉ cho bạn thấy quá trình hình thành của hệ điều hành gắn liền với quá trình hình thành máy tính. Mục tiêu của chúng tôi trong mục này là muốn nhấn mạnh với các bạn mấy điểm sau đây:

Các ngôn ngữ lập trình, đặc biệt là các ngôn ngữ lập trình cấp thấp, ra đời trước các hệ điều hành. Đa số các hệ điều hành đều được xây dựng từ ngôn ngữ lập trình cấp thấp trừ hệ điều hành Unix, nó được xây dựng từ C, một ngôn ngữ lập trình cấp cao.

Nếu không có hệ điều hành thì việc khai thác và sử dụng máy tính sẽ khó khăn và phức tạp rất nhiều và không phải bất kỳ ai cũng có thể sử dụng máy tính được.

Sự ra đời và phát triển của hệ điều hành gắn liền với sự phát triển của máy tính, và ngược lại sự phát triển của máy tính kéo theo sự phát triển của hệ điều hành. Hệ điều hành thực sự phát triển khi máy tính PC xuất hiện trên thị trường.

Ngoài ra chúng tôi cũng muốn giới thiệu một số khái niệm như: hệ thống xử lý theo lô, hệ thống đa chương, hệ thống chia sẻ thời gian, kỹ thuật Spool, ..., mà sự xuất hiện của những khái niệm này đánh dấu một bước phát triển mới của hệ điều hành. Chúng ta sẽ làm rõ các khái niệm trên trong các chương sau của tài liệu này.

3.2.Cấu trúc hệ thống

3.2.1.Hệ thống đơn khối (monolithic systems)

Trong hệ thống này hệ điều hành là một tập hợp các thủ tục, mỗi thủ tục có thể gọi thực hiện một thủ tục khác bất kỳ lúc nào khi cần thiết.

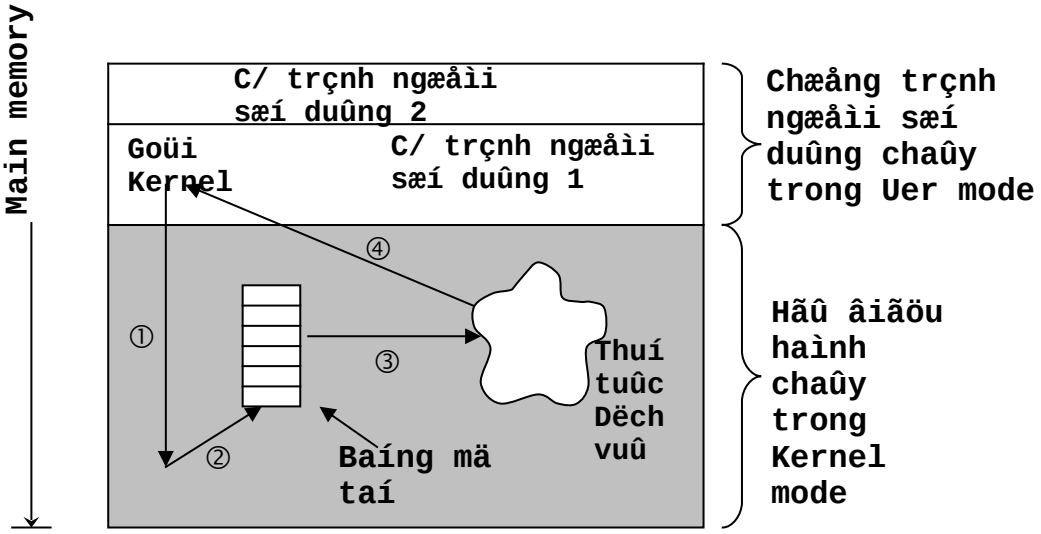
Hệ thống đơn khối thường được tổ chức theo nhiều dạng cấu trúc khác nhau:

- Sau khi biên dịch tất cả các thủ tục riêng hoặc các file chứa thủ

tục của hệ điều hành được liên kết lại với nhau và được chứa vào một file được gọi là file đối tượng, trong file đối tượng này còn chứa cả các thông tin về sự liên kết của các thủ tục.

❑ Sau khi biên dịch các thủ tục của hệ điều hành không được liên kết lại, mà hệ thống chỉ tạo ra file hoặc một bảng chỉ mục để chứa thông tin của các thủ tục hệ điều hành, mỗi phần tử trong bảng chỉ mục chứa một con trỏ trỏ tới thủ tục tương ứng, con trỏ này dùng để gọi thủ tục khi cần thiết. Ta có thể xem cách gọi ngắt (Interrupt) trong ngôn ngữ lập trình cấp thấp và cách thực hiện đáp ứng ngắt dựa vào bảng vector ngắt trong MS_DOS là một ví dụ cho cấu trúc này.

Hình vẽ 1.3 sau đây minh họa cho việc đáp ứng một lời gọi dịch vụ từ chương trình của người sử dụng dựa vào bảng chỉ mục.



Hçnh 1.3: Så âäö thæûc hiãûn låìi goüi hãû thäúng

Trong đó:

1. Chương trình của người sử dụng gởi yêu cầu đến Kernel.
2. Hệ điều hành kiểm tra yêu cầu dịch vụ.
3. Hệ điều hành xác định (vị trí) và gọi thủ tục dịch vụ tương ứng.
4. Hệ điều hành trả điều khiển lại cho chương trình người sử dụng.

Sau đây là một cấu trúc đơn giản của hệ thống đơn khối, trong cấu trúc này các thủ tục được chia thành 3 lớp:

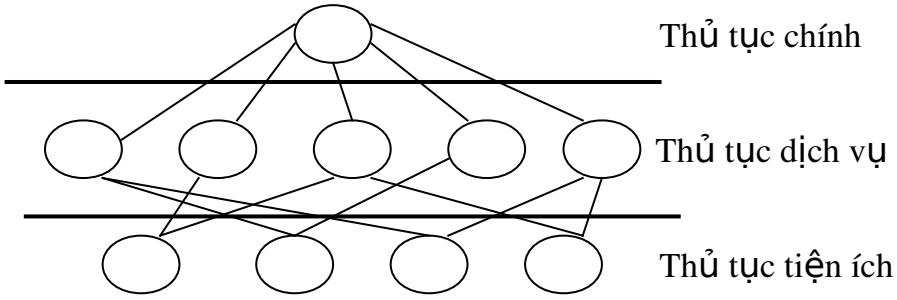
1.Một chương trình chính (chương trình của người sử dụng) gọi đến một thủ tục dịch vụ của hệ điều hành. Lời gọi này được gọi là lời gọi hệ thống.

2.Một tập các thủ tục dịch vụ (service) để đáp ứng những lời gọi hệ thống từ các chương trình người sử dụng.

3.Một tập các thủ tục tiện ích (utility) hỗ trợ cho các thủ tục dịch trong việc thực hiện cho các lời gọi hệ thống.

Trong cấu trúc này mỗi lời gọi hệ thống sẽ gọi một thủ tục dịch vụ tương ứng. Thủ tục tiện ích thực hiện một vài điều gì đó mà thủ tục dịch vụ cần,

chẳng hạn như nhận dữ liệu từ chương trình người sử dụng. Các thủ tục của hệ điều hành được chia vào 3 lớp theo như hình vẽ dưới đây.



.

Hình 1.4: Cấu trúc đơn giản của một monolithic system

Nhận xét:

Với cấu trúc này chương trình của người sử dụng có thể truy xuất trực tiếp đến các chi tiết phần cứng bằng cách gọi một thủ tục cấp thấp, điều này gây khó khăn cho hệ điều hành trong việc kiểm soát và bảo vệ hệ thống.

Các thủ tục dịch vụ mang tính chất tỉnh, nó chỉ hoạt động khi được gọi bởi chương trình của người sử dụng, điều này làm cho hệ điều hành thiếu chủ động trong việc quản lý môi trường.

3.2.2.Các hệ thống phân lớp (Layered Systems)

Hệ thống được chia thành một số lớp, mỗi lớp được xây dựng dựa vào lớp bên trong. Lớp trong cùng thường là phần cứng, lớp ngoài cùng là giao diện với người sử dụng.

Mỗi lớp là một đối tượng trừu tượng, chứa dựng bên trong nó các dữ liệu và thao tác xử lý dữ liệu đó. Lớp n chứa dựng một cấu trúc dữ liệu và các thủ tục có thể được gọi bởi lớp n+1 hoặc ngược lại có thể gọi các thủ tục ở lớp n-1.

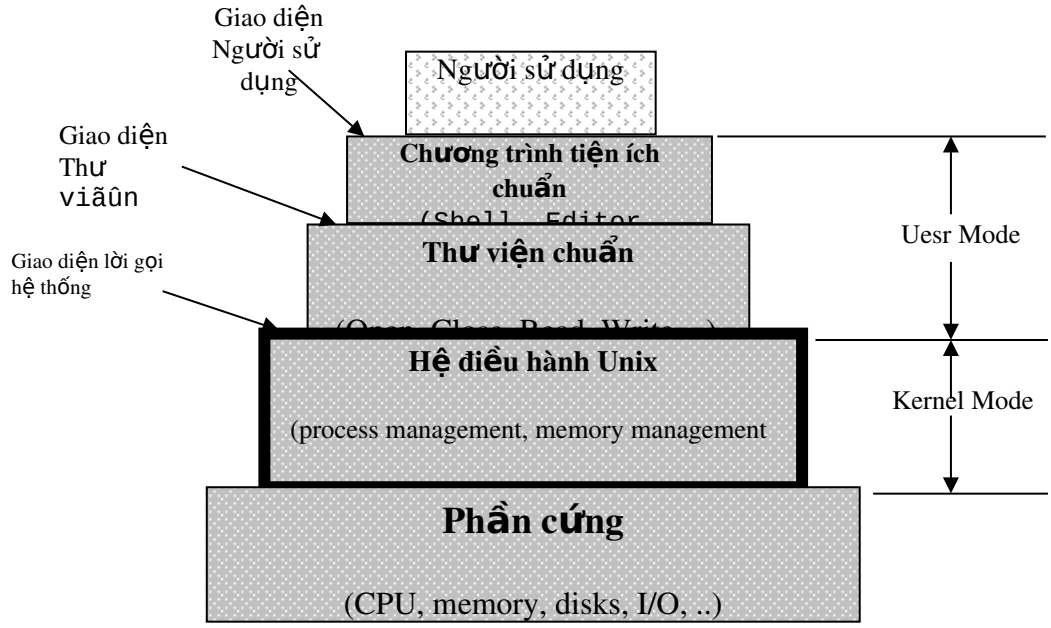
Ví dụ về một hệ điều hành phân lớp:

Lớp 5: Chương trình ứng dụng
Lớp 4: Quản lý bộ đệm cho các thiết bị xuất nhập
Lớp 3: Trình điều khiển thao tác console
Lớp 2: Quản lý bộ nhớ
Lớp 1: Điều phối processor
Lớp 0: Phần cứng hệ thống

Hình vẽ 1.5 sau đây cho ta thấy cấu trúc phân lớp trong hệ điều hành Unix.

Nhận xét:

Khi xây dựng hệ điều hành theo hệ thống này các nhà thiết kế gặp khó khăn trong việc xác định số lượng lớp, thứ tự và chức năng của mỗi lớp.



Hình 1.5: Hệ thống phân lớp của UNIX

Hệ thống này mang tính đơn thể, nên dễ cài đặt, tìm lỗi và kiểm chứng hệ thống.

Trong một số trường hợp lời gọi thủ tục có thể lan truyền đến các thủ tục khác ở các lớp bên trong nên chi phí cho vấn đề truyền tham số và chuyển đổi ngữ cảnh tăng lên, dẫn đến lời gọi hệ thống trong cấu trúc này thực hiện chậm hơn so với các cấu trúc khác.

3.2.3.Máy ảo (Virtual Machine)

Thông thường một hệ thống máy tính bao gồm nhiều lớp: phần cứng ở lớp thấp nhất, hạt nhân ở lớp kế trên. Hạt nhân dùng các chỉ thị (lệnh máy) của phần cứng để tạo ra một tập các lời gọi hệ thống. Các hệ điều hành hiện đại thiết kế một lớp các chương trình hệ thống nằm giữa hệ điều hành và chương trình của người sử dụng.

Các chương trình hệ thống có thể sử dụng các lời gọi hệ thống hoặc sử dụng trực tiếp các chỉ thị phần cứng để thực hiện một chức năng hoặc một thao tác nào đó, do đó các chương trình hệ thống thường xem các lời gọi hệ thống và các chỉ thị phần cứng như ở trên cùng một lớp.

Một số hệ điều hành tổ cho phép các chương trình của người sử dụng có thể gọi dễ dàng các chương trình hệ thống và xem mọi thành phần dưới chương trình hệ thống đều là phần cứng máy tính. Lớp các ứng dụng này sử dụng khái niệm máy ảo.

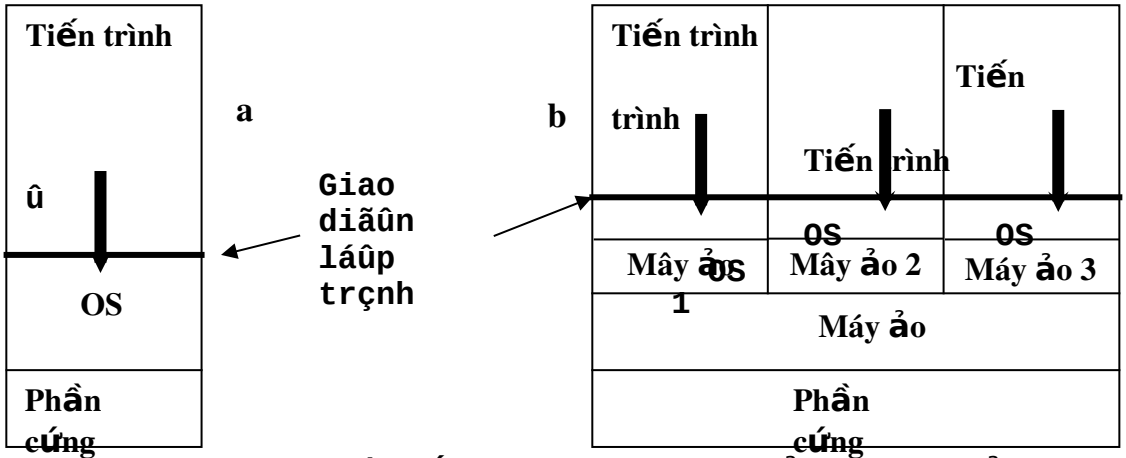
Mục đích của việc sử dụng máy ảo là xây dựng các hệ thống đa chương với nhiều tiến trình thực hiện đồng thời, mỗi tiến trình được cung

cấp một máy ảo với đầy đủ tài nguyên, tất nhiên là tài nguyên ảo, để nó thực hiện được.

Trong cấu trúc này phần nhân của hệ thống trở thành bộ phận tổ chức giám sát máy ảo, phần này chịu trách nhiệm giao tiếp với phần cứng, chia sẻ tài nguyên hệ thống để tạo ra nhiều máy ảo, hoạt động độc lập với nhau, để cung cấp cho lớp trên.

Ở đây cần phân biệt sự khác nhau giữa máy ảo và máy tính mở rộng, máy ảo là bản sao chính xác các đặc tính phần cứng của máy tính thực sự và cho phép hệ điều hành hoạt động trên nó, sau đó hệ điều hành xây dựng máy tính mở rộng để cung cấp cho người sử dụng.

Với cấu trúc này mỗi tiến trinh hoạt động trên một máy ảo độc lập và nó có cảm giác như đang sở hữu một máy tính thực sự.



Hình 1.6: Mô hình hệ thống (a) Không có máy ảo (b) Máy ảo

Hình vẽ trên đây cho chúng ta thấy sự khác nhau trong hệ thống không có máy ảo và hệ thống có máy ảo:

Nhận xét:

Việc cài đặt các phần mềm giả lập phần cứng để tạo ra máy ảo thường rất khó khăn và phức tạp.

Trong hệ thống này vấn đề bảo vệ tài nguyên hệ thống và tài nguyên đã cấp phát cho các tiến trình, sẽ trở nên đơn giản hơn vì mỗi tiến trình thực hiện trên một máy tính (ảo) độc lập với nhau nên việc tranh chấp tài nguyên là không thể xảy ra.

Nhờ hệ thống máy ảo mà một ứng dụng được xây dựng trên hệ điều hành có thể hoạt động được trên hệ điều hành khác. Trong môi trường hệ điều hành Windows 9x người sử dụng có thể thực hiện được các ứng dụng được thiết kế để thực hiện trên môi trường MS_DOS, sở dĩ như vậy là vì Windows đã cung cấp cho các ứng dụng này một máy ảo DOS (VMD: Virtual Machine DOS) để nó hoạt động như đang hoạt động

trong hệ điều hành DOS. Tương tự trong môi trường hệ điều hành Windows NT người sử dụng có thể thực hiện được các ứng dụng được thiết kế trên tất cả các hệ điều hành khác nhau, có được điều này là nhờ trong cấu trúc của Windows NT có chứa các hệ thống con (subsystems) môi trường tương thích với các môi trương hệ điều hành khác nhau như: Win32, OS/2,..., các ứng dụng khi cần thực hiện trên Windows NT sẽ thực hiện trong các hệ thống con môi trường tương ứng, đúng với môi trường mà ứng dụng đó được tạo ra.

2.3.4.Mô hình Client/ Server (client/ server model)

Các hệ điều hành hiện đại thường chuyển dần các tác vụ của hệ điều hành ra các lớp bên ngoài nhằm thu nhỏ phần cốt lõi của hệ điều hành thành hạt nhân cực tiểu (kernel) sao cho chỉ phần hạt nhân này phụ thuộc vào phần cứng. Để thực hiện được điều này hệ điều hành xây dựng theo mô hình Client/ Server, theo mô hình này hệ điều hành bao gồm nhiều tiến trình đóng vai trò Server có các chức năng chuyên biệt như quản lý tiến trình, quản lý bộ nhớ, ..., phần hạt nhân cuả hệ điều hành chỉ thực hiện nhiệm vụ tạo cơ chế thông tin liên lạc giữa các tiến trình Client và Server.
Như vậy các tiến trình trong hệ thống được chia thành 2 loại:

Tiến trình bên ngoài hay tiến trình của chương trình người sử dụng được gọi là các tiến trình Client.

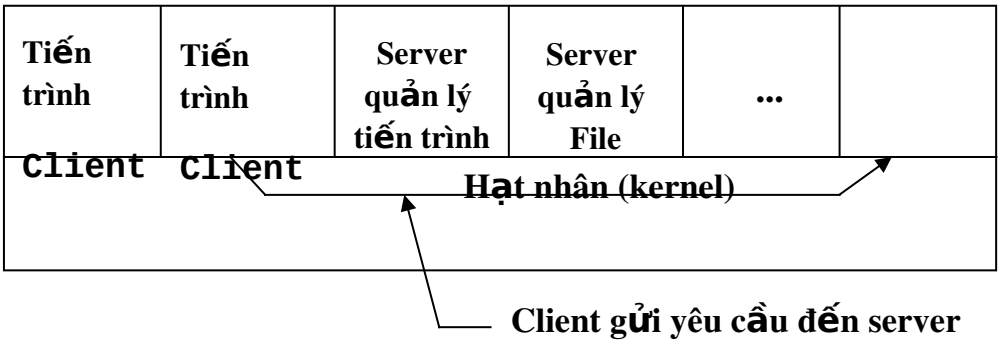
Tiến trình của hệ điều hành được gọi là tiến trình Server.

Khi cần thực hiện một chức năng hệ thống các tiến trình Client sẽ gởi yêu cầu tới tiến trình server tương ứng, tiến trình server sẽ xử lý và trả lời kết quả cho tiến trình Client.

Nhận xét:

Hệ thống này dễ thay đổi và dễ mở rộng hệ điều hành. Để thay đổi các chức năng của hệ điều hành chỉ cần thay đổi ở server tương ứng, để mở rộng hệ điều hành chỉ cần thêm các server mới vào hệ thống.

Các tiến trình Server của hệ điều hành hoạt động trong chế độ không đặc quyền nên không thể truy cập trực tiếp đến phần cứng, điều này giúp hệ thống được bảo vệ tốt hơn.



Hình 1.7: Mô hình client- server

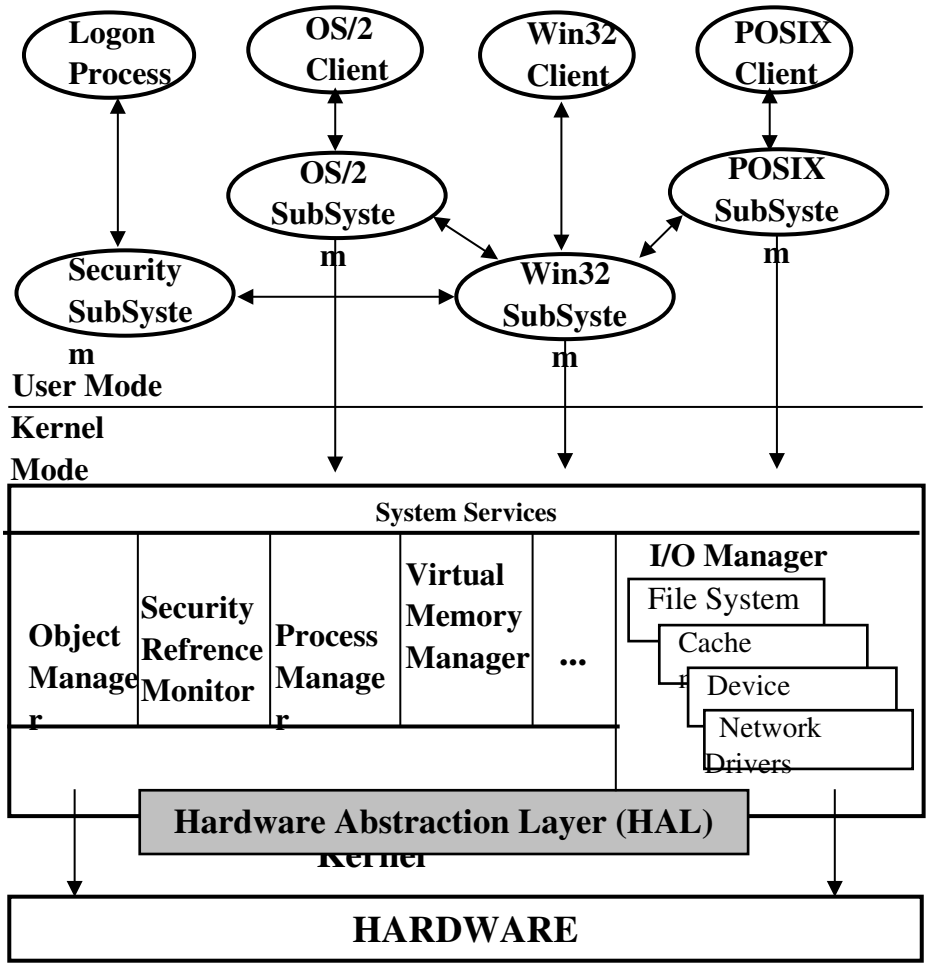
➢ Hình vẽ sau đây cho thấy cấu trúc của hệ điều hành Windows NT. Đây là một cấu trúc phức tạp với nhiều thành phần khác nhau và nó được xây dựng dựa trên mô hình hệ điều hành Client/ Server.

Trong cấu trúc này chúng ta thấy nổi rõ hai điểm sau đây:

Cấu trúc của windows NT được chia thành 2 mode: Kernel mode và User mode. Các chương trình ứng dụng của người sử dụng chỉ chạy trong User mode, các dịch vụ của hệ điều hành chỉ chạy trong Kernel mode. Nhờ vậy mà việc bảo vệ các chương trình của người sử dụng cũng như các thành phần của hệ điều hành, trên bộ nhớ, được thực hiện dễ dàng hơn.

Trong User mode của Windows NT có chứa các hệ thống con môi trường như: OS/2 subsystem và POSIX subsystem, nhờ có các hệ thống con môi trường này mà các ứng dụng được thiết kế trên các hệ điều hành khác vẫn chạy được trên hệ điều hành Windows NT. Đây là điểm mạnh của các hệ điều hành Microsoft của từ Windows NT.

Chúng tôi sẽ giải thích rõ hơn về hai khái niệm **Kernel mode** và **User mode**, và các thành phần trong cấu trúc của hệ điều hành Windows NT ở phần sau, thông qua việc giới thiệu về hệ điều hành Windows 2000.



Hình 1.8: Cấu trúc của Windows NT

3.3.Cài đặt Linux

3.3.1.Giới thiệu hệ điều hành Linux

Linux là hệ điều hành mô phỏng Unix, được xây dựng trên phần nhân (kernel) và các gói phần mềm mã nguồn mở. Linux được công bố dưới bản quyền của GPL (General Public Licence).

Unix ra đời giữa những năm 1960, ban đầu được phát triển bởi AT&T, sau đó được đăng ký thương mại và phát triển theo nhiều dòng dưới các tên khác nhau. Năm 1990 xu hướng phát triển phần mềm mã nguồn mở xuất hiện và được thúc đẩy bởi tổ chức GNU. Một số licence về mã nguồn mở ra đời ví dụ BSD, GPL. Năm 1991, Linus Torvald viêt thêm phiên bản nhân v0.01 (kernel) đầu tiên của Linux đưa lên các BBS, nhóm người dùng để mọi người cùng sử dụng và phát triển. Năm 1996, nhân v1.0 chính thức công bố và ngày càng nhận được sự quan tâm của người dùng. Năm 1999,

phiên bản nhân v2.2 mang nhiều đặc tính ưu việt và giúp cho linux bắt đầu trở thành đối thủ cạnh tranh đáng kể của MSwindows trên môi trường server. Năm 2000 phiên bản nhân v2.4 hỗ trợ nhiều thiết bị mới (đa xử lý tới 32 chip, USB, RAM trên 2GB...) bắt đầu đặt chân vào thị trường máy chủ cao cấp. Quá trình phát triển của linux như sau:

- Năm 1991: 100 người dùng.
- Năm 1997: 7.000.000 người dùng.
- Năm 2000: hàng trăm triệu người dùng, hơn 15.000 ngườ i tham gia phát triển Linux. Hàng năm thị trường cho Linux tăng trưởng trên 100%.

Các phiên bản Linux là sản phẩm đóng gói Kernel và các gói phần mềm miễn phí khác. Các phiên bản này được công bố dưới licence GPL. Một số phiên bản nổi bật là: Redhat, Caldera, Suse, Debian, TurboLinux, Mandrake.

Giống như Unix, Linux gồm 3 thành phần chính: kernel, shell và cấu trúc file.

Kernel là chương trình nhân, chạy các chương trình và quản lý các thiết bị phần cứng như đĩa và máy in.

Shell (môi trường) cung cấp giao diện cho người sử dụng, còn được mô tả như một bộ biên dịch. Shell nhận các câu lệnh từ người sử dụng và gửi các câu lệnh đó cho nhân thực hiện. Nhiều shell được phát triển. Linux cung cấp một số shell như: desktops, windows manager, và môi trường dòng lệnh. Hiện nay chủ yếu tồn tại 3 shell: Bourne, Korn và C shell. Bourne được phát triển tại phòng thí nghiệm Bell, C shell được phát triển cho phiên bản BSD của UNIX, Korn shell là phiên bản cải tiến của Bourne shell. Những phiên bản hiện nay của Unix, bao gồm cả Linux, tích hợp cả 3 shell trên.

Cấu trúc file quy định cách lưu trữ các file trên đĩa. File được nhóm trong các thư mục. Mỗi thư mục có thể chứa file và các thư mục con khác. Một số thư mục là các thư mục chuNn do hệ thống sử dụng. Người dùng có thể tạo các file/thư mục của riêng mình cũng như dịch chuyển các file giữa các thư mục đó. Hơn nữa, với Linux người dùng có thể thiết lập quyền truy nhập file/thư mục, cho phép hay hạn chế một người dùng hoặc một nhóm truy nhập file. Các thư mục trong Linux được tổ chức theo cấu trúc cây, bắt đầu bằng một thư mục gốc (root). Các thư mục khác được phân nhánh từ thư mục này.

Kernel, shell và cấu trúc file cấu thành nên cấu trúc hệ điều hành. Với những thành phần trên người dùng có thể chạy chương trình, quản lý file, và tương tác với hệ thống.

3.3.2.Cài đặt Linux

Lưu ý: trước khi cài đặt, cần tìm hiểu các thông tin về phần cứng của hệ thống, bao gồm

- Thông tin về ổ đĩa cứng
- Thông tin về card mạng
- Thông tin về card đồ hoạ
- Thông tin về màn hình
- Thông tin về giao thức và cấu hình mạng nếu kết nối mạng
- Thông tin về các thiết bị ngoài.

Có thể chọn nhiều phương án cài đặt như cài đặt từ đĩa mềm, từ đĩa cứng, từ đĩa CD Rom hoặc qua mạng. Tài liệu này chọn hướng dẫn quá trình cài đặt phiên bản 7.0 từ đĩa CDRom. Yêu cầu máy cài đặt có khả năng khởi động (boot) từ ổ đĩa CD-Rom (được hỗ trợ hầu hết trong các máy tính hiện nay).

Sau đây là các bước cài đặt cụ thể. Khi kết thúc bước trước chương trình cài đặt tự động chuyển sang bước sau. Một số bước cài đặt cho phép quay lại bước trước bằng cách chọn Back.

1. Đưa đĩa CD Rom Redhat vào ổ đĩa. Khởi động lại máy (lưu ý phải đảm bảo máy có khả năng khởi động từ đĩa CD-Rom. Chọn chế độ cài ***text***
2. Chọn chế độ cài ***text***

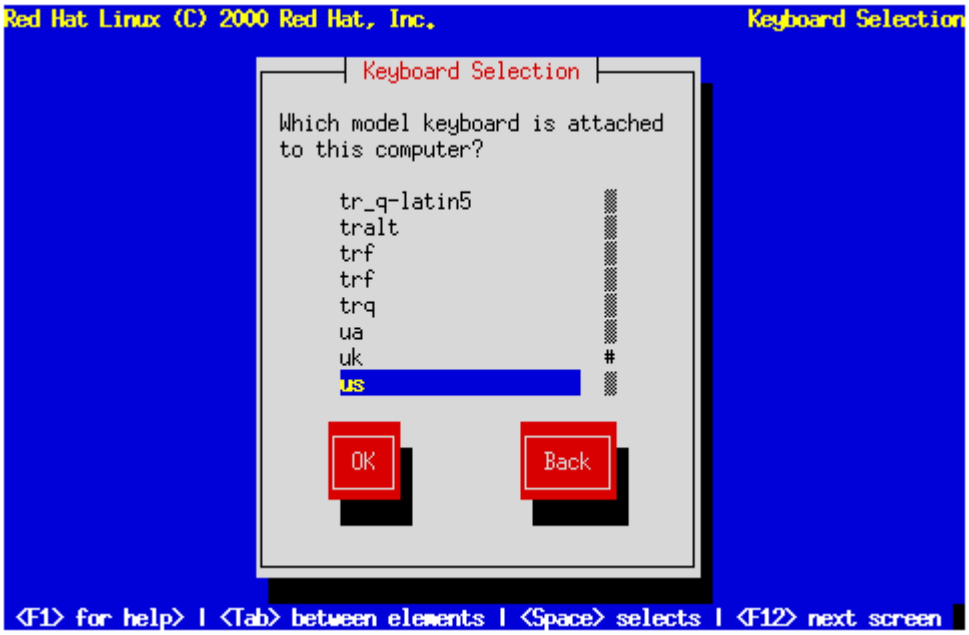
```
boot: text
```

3. Lựa chọn ngôn ngữ

Chọn ngôn ngữ mặc định là English

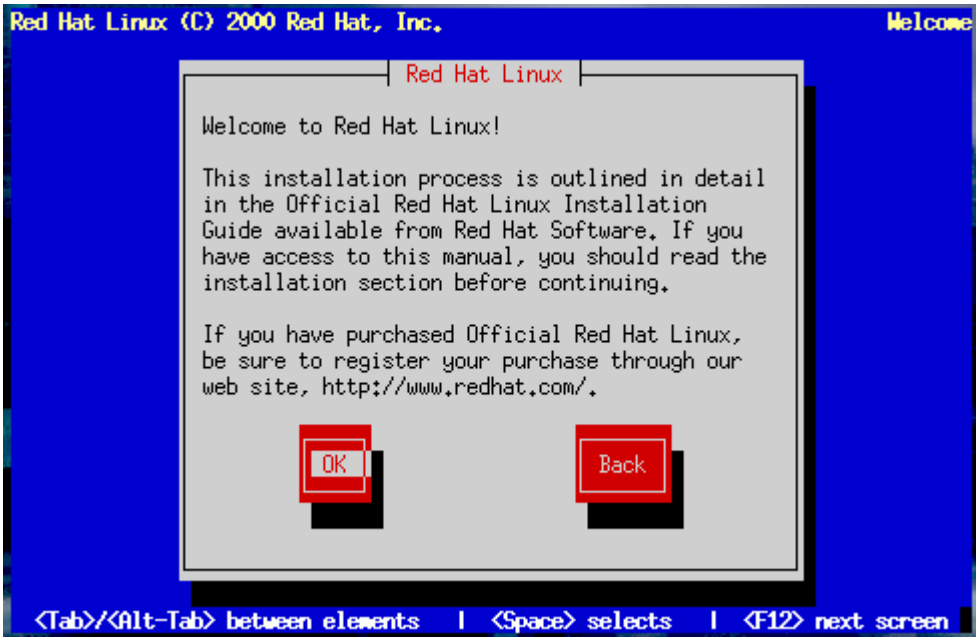
4. Lựa chọn kiểu bàn phím

Lựa chọn kiểu thể hiện bàn phím là **us**.



5. Màn hình chào mừng

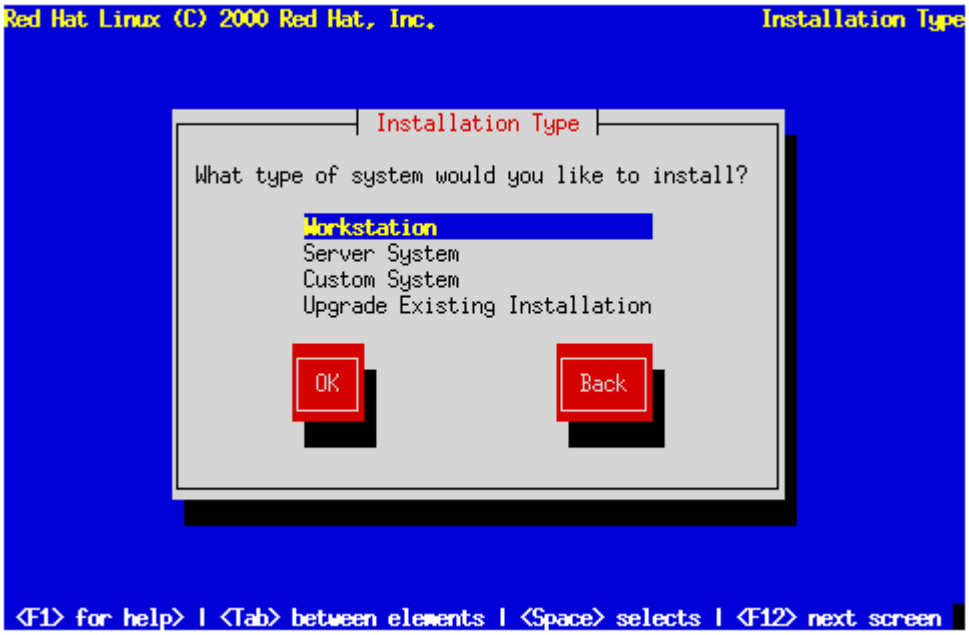
Sau khi đã lựa chọn xong ngôn ngữ cài đặt, bàn phím và phương pháp cài đặt, màn hình chào mừng xuất hiện. Bấm OK để tiếp tục.



6. Chọn kiểu cài đặt

Hộp hội thoại cho phép bạn chọn lựa kiểu cài đặt hệ điều hành Linux RedHat như một Workstation, Server, Custom hay chỉ là nâng cấp phiên bản đã cài đặt.

Chọn kiểu cài đặt là Custom System. Chọn OK để tiếp tục.

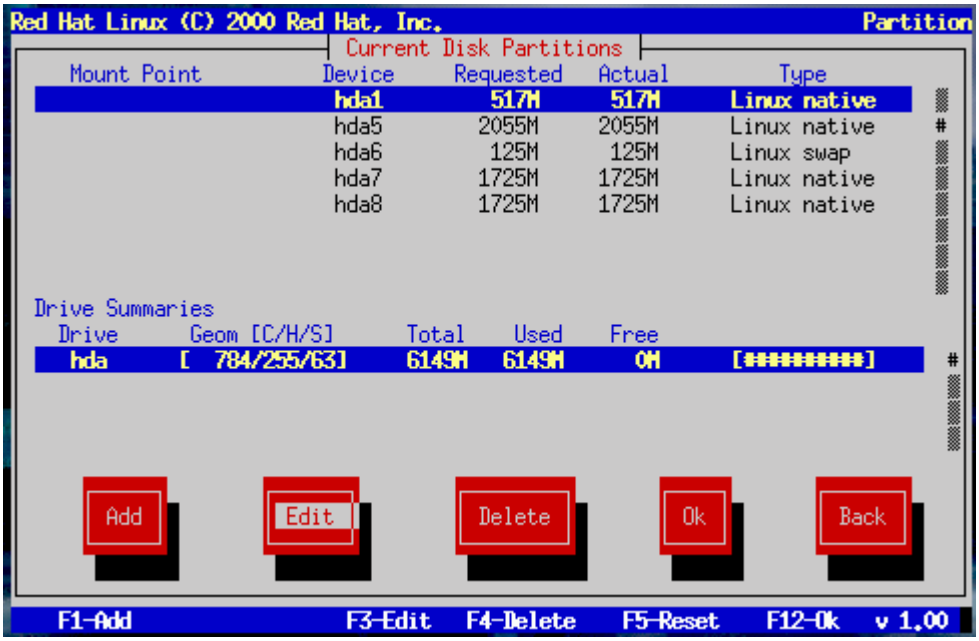


7. Lựa chọn phần mềm phân chia ổ đĩa

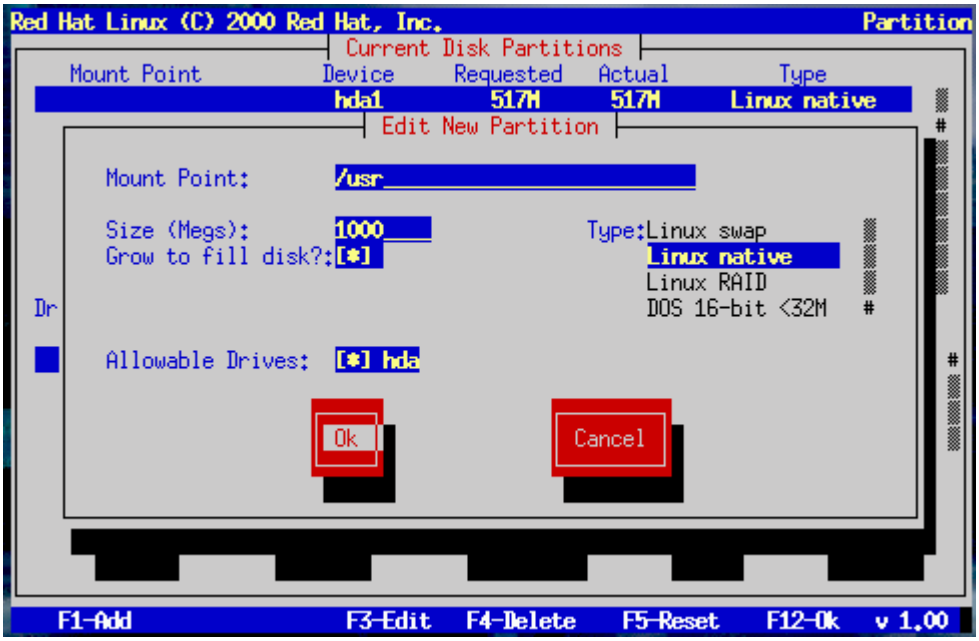
Linux đưa ra cho bạn hai phần mềm để phân chia ổ đĩa dành cho Linux: đó là Disk Druid và fdisk. Chọn Disk Druid để tiếp tục.

Bạn cần tạo 2 partition để install RedHat, nhớ đừng delete những partition có sẵn trong máy bạn (nếu không thì dữ liệu có sẵn sẽ mất, tốt nhất là bạn nên sao lưu dữ liệu trước cho bảo đảm!). Dùng các chức năng **add**, **edit**, **delete** tạo 1 partition với type là **Linux swap**, dung lượng bằng dung lượng RAM của máy. Tiếp theo tạo một partion tên "/" với loại **Linux native**, dung lượng ít nhất là 500Mb (tuỳ theo dung lượng còn trống của đĩa bạn, nếu bạn muốn install trọn gói RedHat thì cần đến

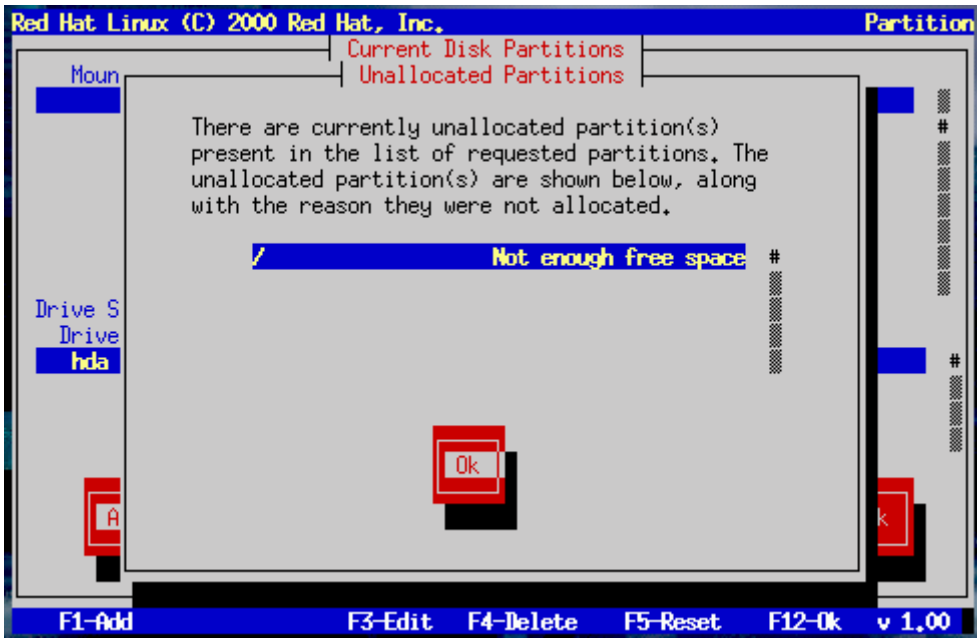
khoảng
2288MB). Hãy yên chí là nếu bạn tạo sai (partition kích thước quá lớn, lớn hơn dung lượng còn trống của đĩa) thì RedHat sẽ không cho bạn đi tiếp. Chỉ cần tạo 2 partition này là đủ rồi. Khi nào bạn click được Next thì *coi như* là thành công!



Để tạo một partition mới, chọn Add. Màn hình **Edit New Partition** xuất hiện

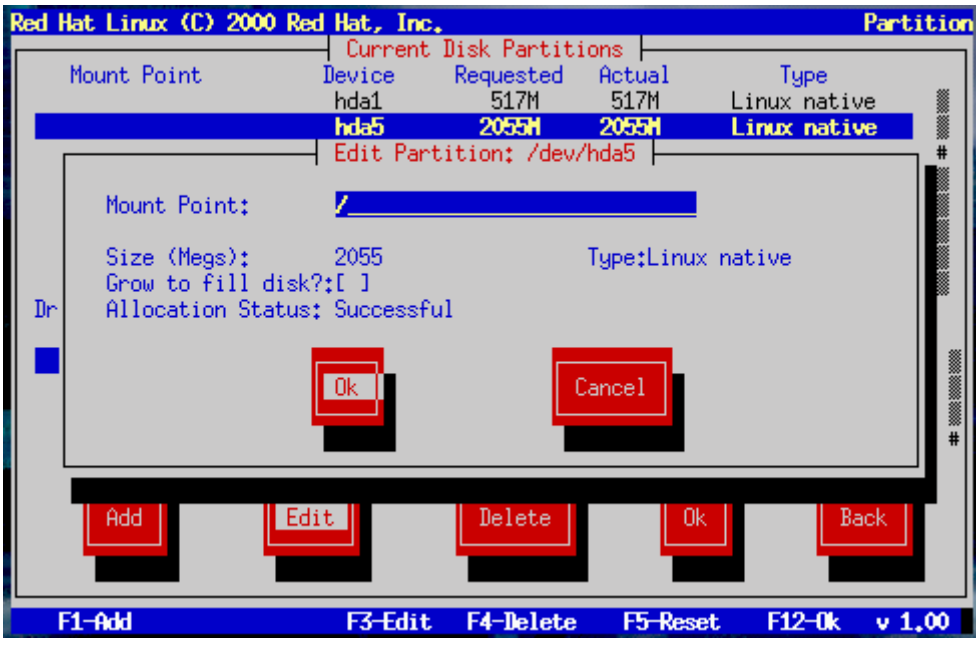


Một số vấn đề có thể xảy ra khi thêm một partition



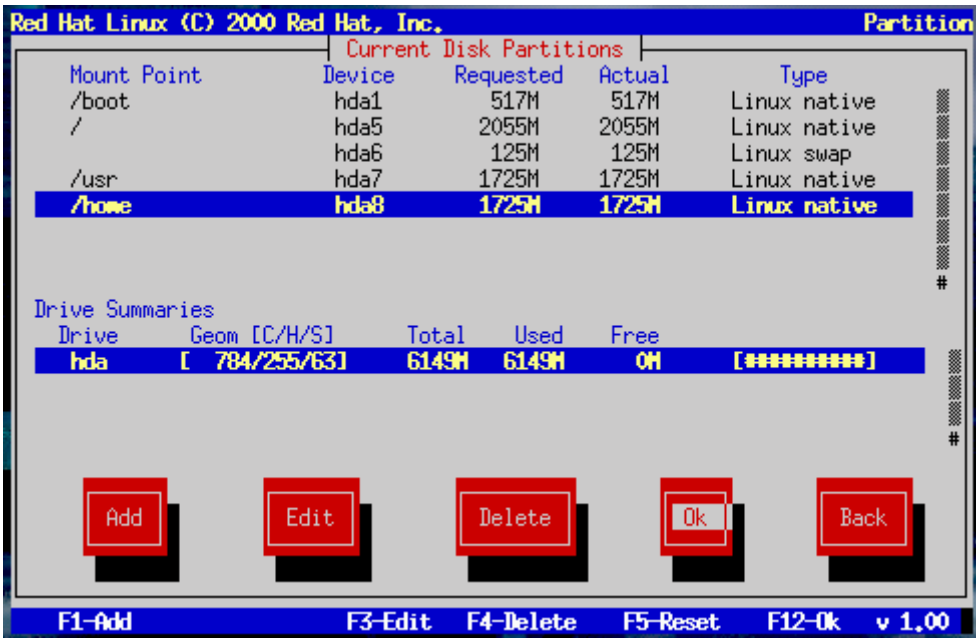
8. Hiệu chỉnh một partition

Chọn một partition cần hiệu chỉnh, nhấn Edit, màn hình mới sẽ cho phép bạn thay đổi các thông số của partition đã chọn như kích thước, kiểu, ...



9. Hoàn thành việc phân chia đĩa

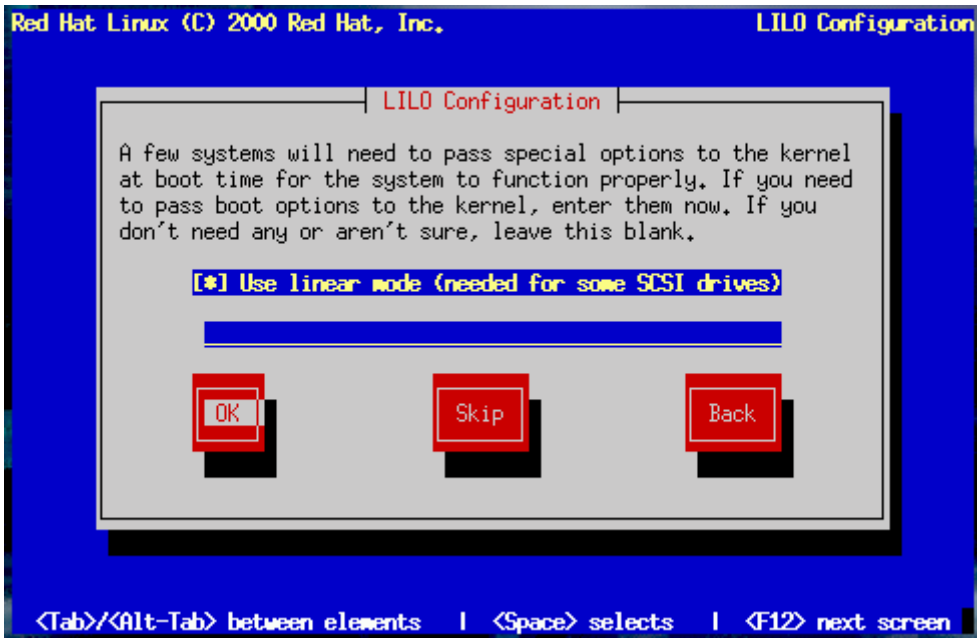
Chương trình cài đặt sẽ yêu cầu bạn format lại phân vùng vừa tạo, chú ý không chọn những phân vùng dữ liệu quan trọng đối với bạn.



10. Khởi tạo LILO

LInux **LO**ader (LILO) cho phép bạn xác định thời gian để khởi tạo Linux hay một hệ điều hành nào khác. Khi khởi tạo cho server, LILO được cấu hình tự động trên Master Boot Record [MBR]. If you are performing a custom-class installation, the **LILO Installation** dialogs let you indicate how or whether to install LILO.

Việc chọn LILO trong cửa sổ **LILO Configuration** cho phép bạn thêm các tuỳ chọn mặc định vào lệnh boot LILO và các tùy chọn này được chuyển cho Linux kernel tại thời điểm boot.



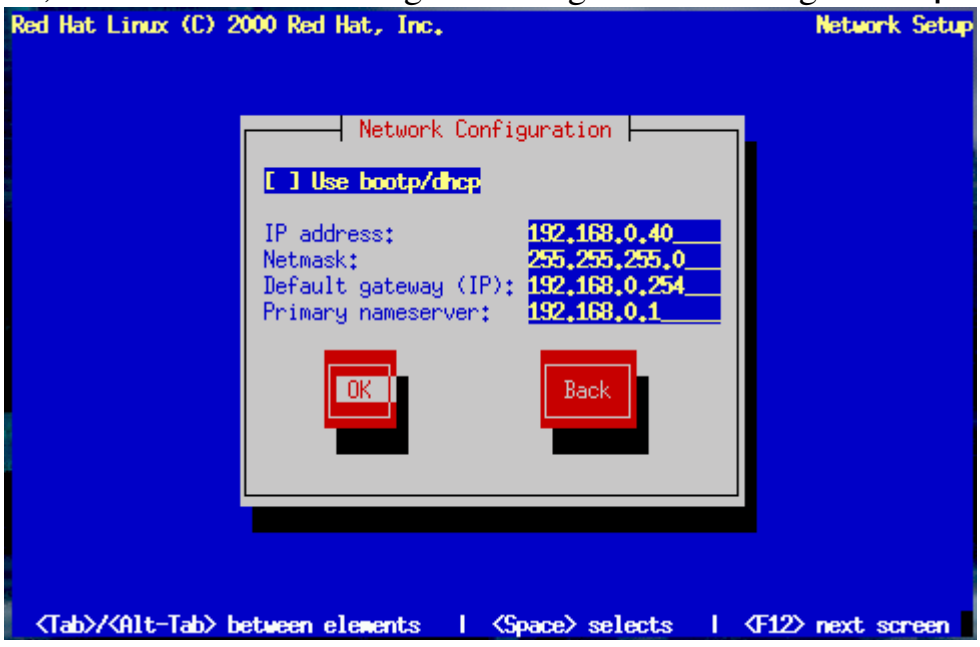
Chú ý rằng nếu bạn chọn **Skip**, bạn sẽ không thể boot hệ thống Red Hat Linux một các trực tiếp mà sẽ phải sử dụng phương pháp boot khác (boot disk chẳng hạn) Bạn chỉ nên lựa chọn cách này khi bạn chắc chắn đã có cách khác để boot hệ thống Red Hat Linux của bạn.
Dùng lựa chọn đặt boot loader tại Master Boot Record để khởi tạo ngay hệ điều hành Linux khi bật máy.

Màn hình này cho phép bạn đặt tên cho máy tính của mình. Bạn có thể thay đổi hostname sau khi đã cài đặt xong bằng lệnh **hostname** ***newname,*** trong đó newname là tên mà bạn muốn đặt.

11. Cấu hình kết nối mạng

Nếu máy không có card mạng, sẽ không nhận được màn hình này. Thực hiện cấu hình mạng cho máy như sau

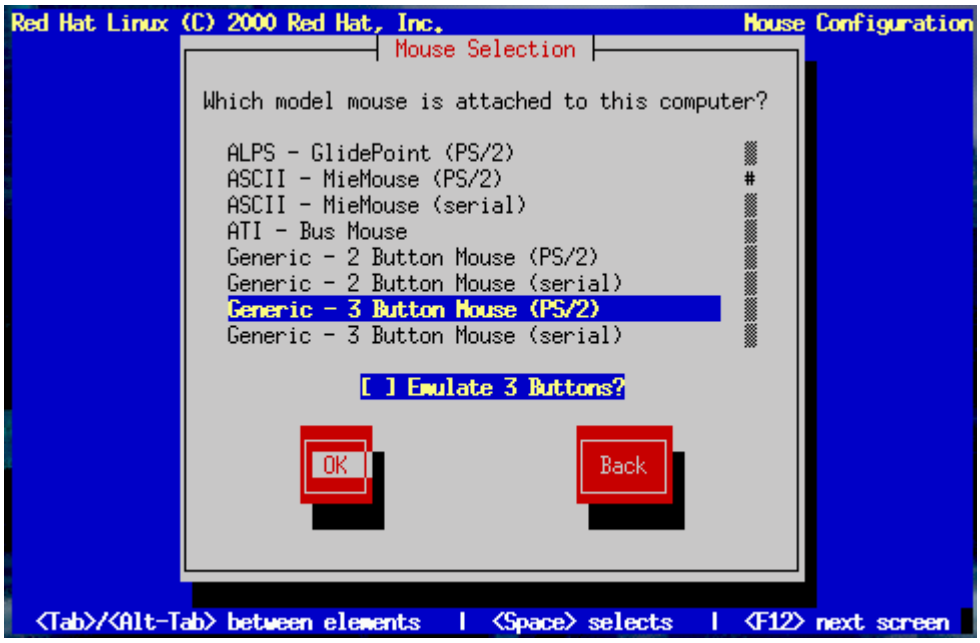
Bỏ lựa chọn ***config using DHCP*** *(chế độ cấp phát địa chỉ IP động)*, nhập địa chỉ IP, subnetmask theo hướng dẫn của giáo viên hướng dẫn thực hành.



12. Cấu hình firewall: chọn ***Medium***

13. Cấu hình chuột

Thông thường thì chương trình cài đặt sẽ tự phát hiện loại chuột của máy bạn. Nếu không, bạn hãy chọn loại chuột phù hợp trong danh sách, và nếu bạn không biết chuột của mình loại gì thì cứ để yên, click **Next** để tiếp tục.



Lựa chọn **Emulate 3 Buttons** cho phép bạn sử dụng chuột của bạn như chuột có 2 nút trong đó dùng nút giữa bằng cách bấm hai nút cùng một lúc. Nếu bạn có chuột hai nút, bạn hãy sử dụng chức năng này vì XWindow trở nên dễ dùng nhất với khi chuột có ba nút.

14. Cấu hình Time Zone

Nếu bạn muốn thiết lập đồng hồ cho CMOS theo giờ GMT (Greenwich Mean Time), chọn **Hardware clock set to GMT**. Tuy nhiên, nếu máy tính của bạn sử dụng một hệ điều hành khác thì việc thiết đặt đồng hồ theo giờ GMT sẽ khiến cho hệ điều hành khác đó hiển thị sai thời gian.

Để đặt giờ VN, chọn Asia/Saigon

Để thay đổi cấu hình về thời gian sau khi bạn đã cài đặt, bạn có thể dùng lệnh **/usr/sbin/timeconfig**

15. Thiết lập mật khẩu root

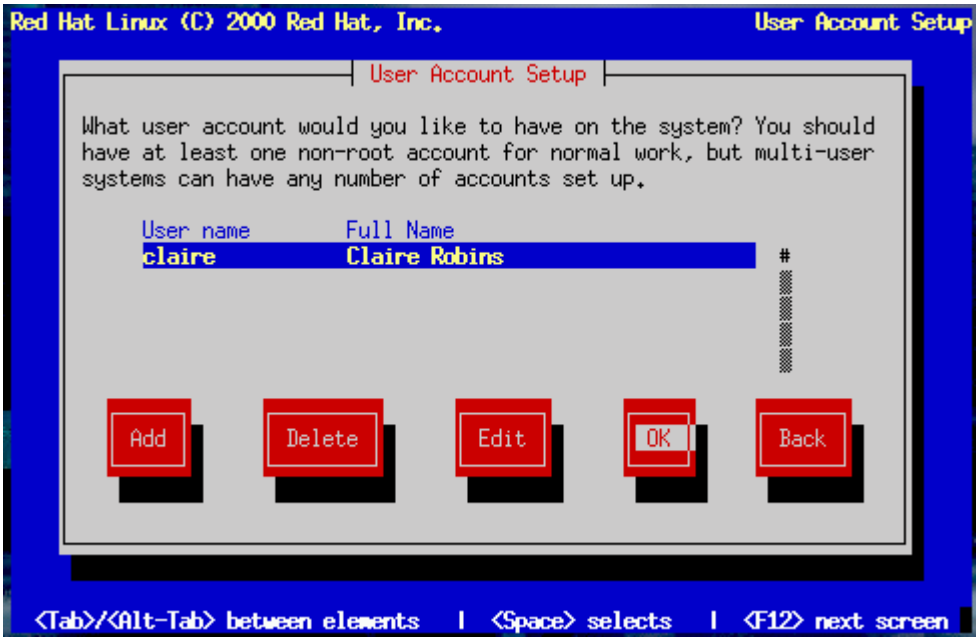
Hộp thoại **Root Password** buộc bạn phải thiết lập một mật khẩu root cho hệ thống của bạn. Bạn sẽ sử dụng mật khẩu này để log vào hệ thống và thực hiện các chức năng quản trị hệ thống của mình.

16. Tạo user

Bạn có thể tạo tài khoản user cho chính mình để sử dụng hàng ngày. User root (*superuser*) có đủ quyền truy nhập vào hệ thống nhưng rất nguy hiểm, chỉ nên sử dụng để bảo dưỡng hay quản trị hệ thống.

Mật khẩu của user có phân biệt chữ hoa chữ thường và ít nhất là 6 ký tự.

17. Bạn có thể tạo tiếp nhiều user theo cửa sổ sau:

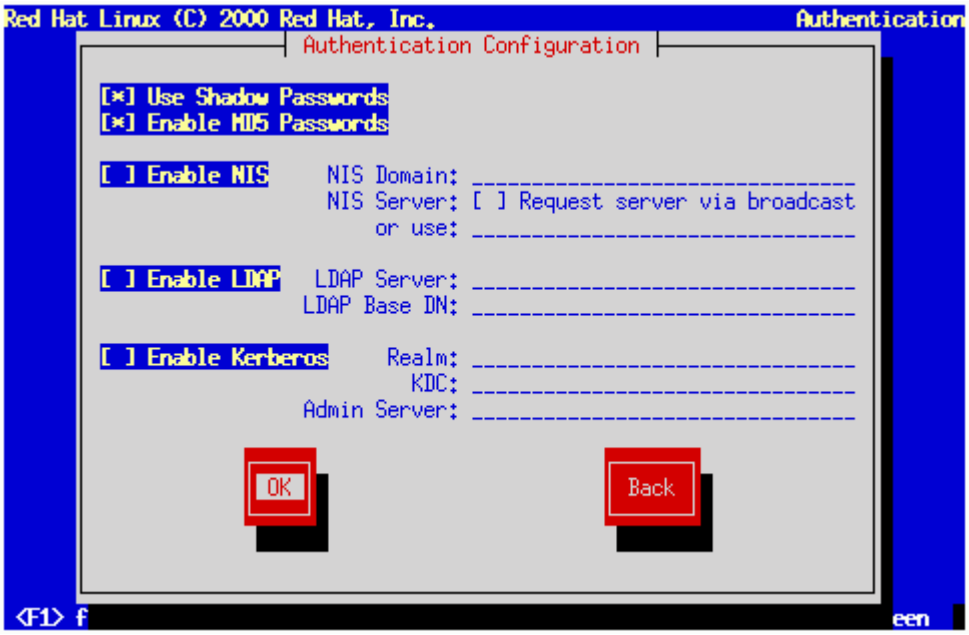


18. Cấu hình xác thực người dùng

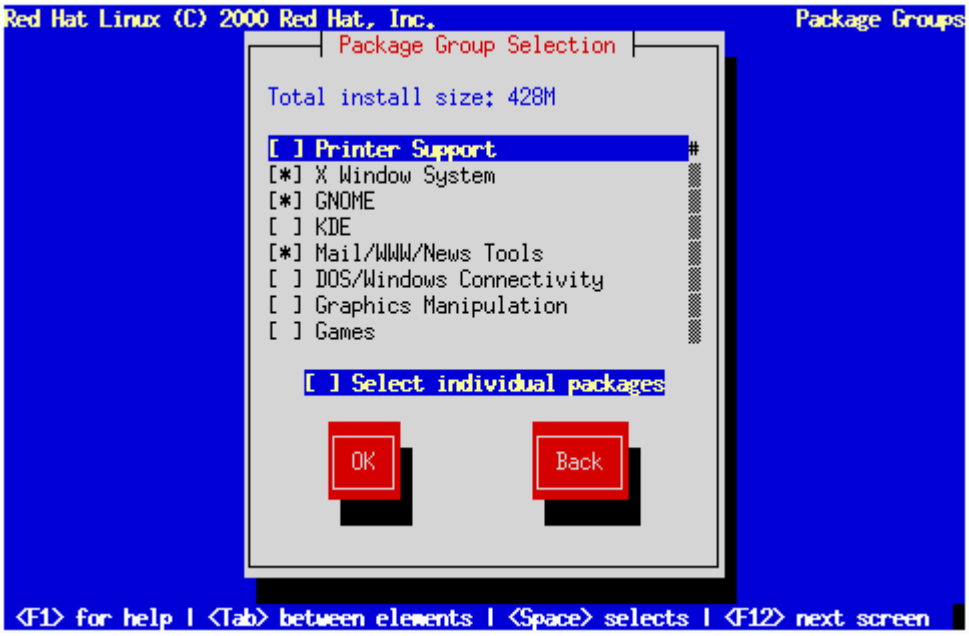
Do bạn khởi tạo theo chế độ custom, bước này cho phép bạn cấu hình cách mà hệ điều hành linux của bạn sử dụng để xác thực mật khẩu.

Lựa chọn **Use Shadow Passwords**: mật khẩu của bạn đáng nhẽ nằm trong tệp/etc/passwd sẽ được thay thế bằng thư mục /etc/shadow và chỉ được truy nhập bởi superuser (root)

Tuỳ chọn **Enable MD5 Passwords** -- cho phép mã hóa mật khẩu theo chuẩn MD5.



19.Tiếp theo, bạn có thể chọn lựa các gói tin để cài đặt. Bạn nên chọn các phần mềm, dịch vụ hay sử dụng nhất để cài đặt sẵn trên máy khi khởi động. Tuy nhiên, tuy nhiên, bạn cũng có thể cài đặt sau này tuỳ theo nhu cầu sử dụng. Các gói tin này nếu được cài đặt sẽ được ghi lại trong tệp /tmp/install.log sau khi khởi tạo lại hệ thống của bạn.

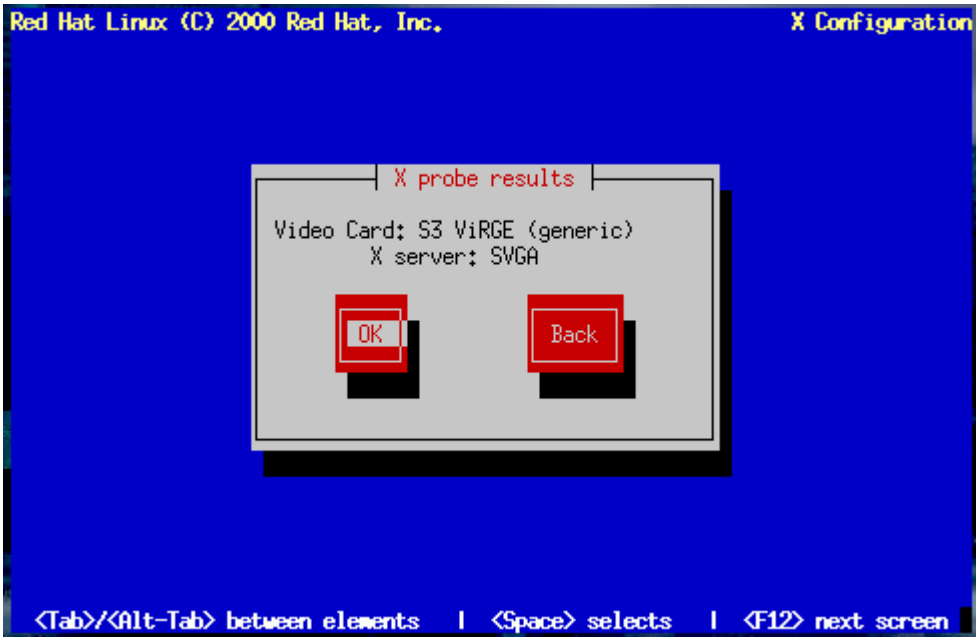


Có thể cài đặt từng gói tin nhỏ hơn bằng cách chọn **Select individual packages** và nhấn OK.

20. Cấu hình Video Adapter

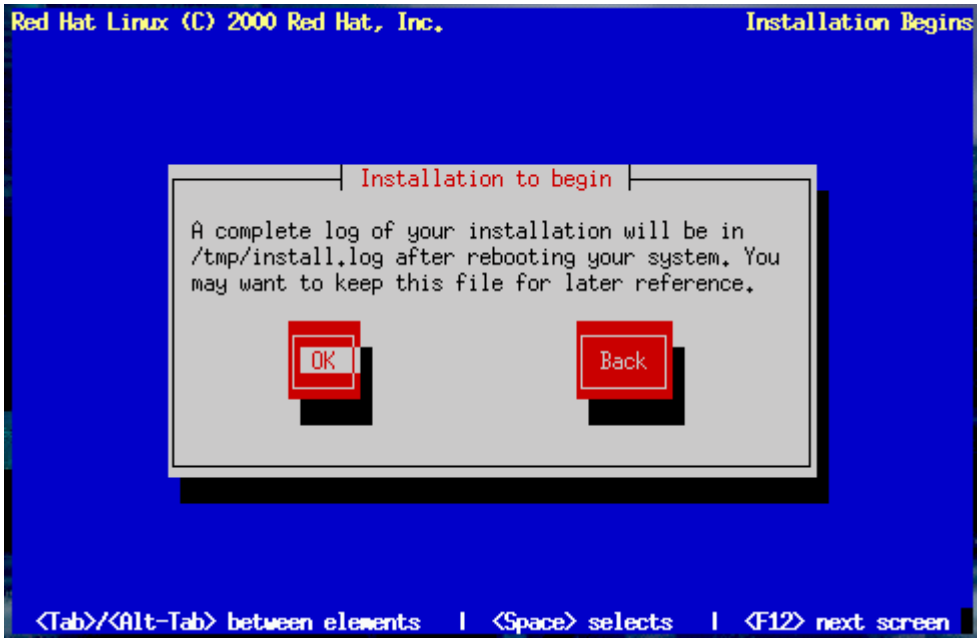
Chương trình cài đặt sẽ tự phát hiện video card khởi tạo. Nhấn OK để tiếp

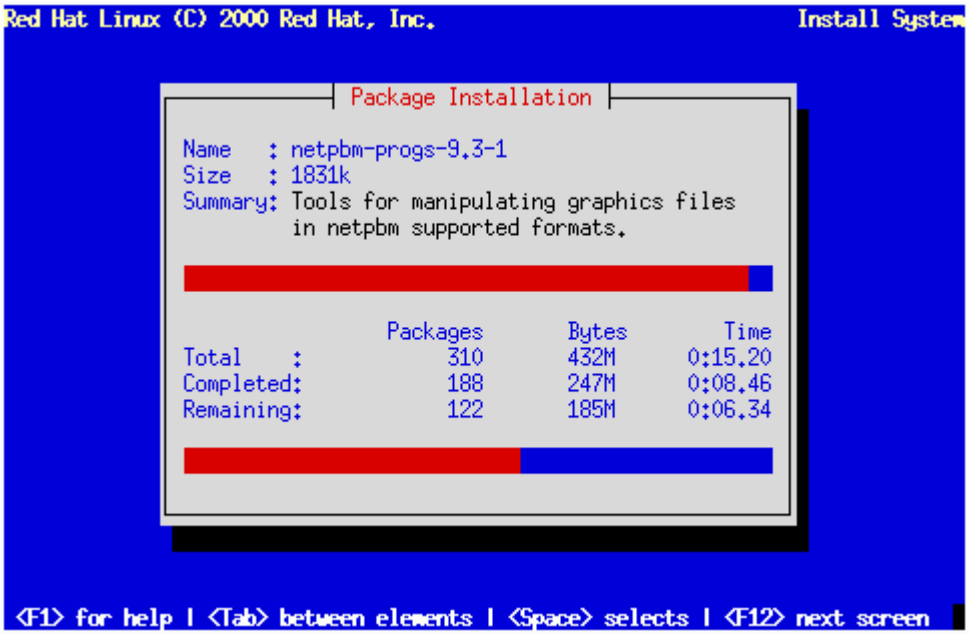
tục.



21. Bắt đầu khởi tạo các gói tin:

Quá trình khởi tạo sẽ được ghi vào tệp /tmp/install.log. Nhấn OK để tiếp tục.




Red Hat Linux (C) 2000 Red Hat, Inc.
Install Syste
Package Installation
Name : netpbm-progs-9.3-1
Size : 1831k
Summary: Tools for manipulating graphics files
in netpbm supported formats.
Packages Bytes Time
Total : 310 432M 0:15.20
Completed: 188 247M 0:08.46
Remaining: 122 185M 0:06.34
<F1> for help | <Tab> between elements | <Space> selects | <F12> next screen

CÂU HỎI CỦNG CỐ BÀI HỌC

1. Hệ điều hành là gì?
2. Có mấy loại hệ điều hành ? Việc phân loại này dựa trên những tiêu chuẩn nào ?
3. Nêu các thành phần chính của hệ điều hành và chức năng của mỗi thành phần này.
4. So sánh các cấu trúc khác nhau của hệ điều hành. Ưu khuyết điểm củ mỗi loại cấu trúc.
5. Quá trình phát triển của hệ điều hành phụ thuộc vào những yếu tố nào.

Bài tập

1. Hệ điều hành là :
a. Một chương trình
b. Một chương trình hay hệ chương trình
c. Một thiết bị
d. ROM-BIOS
2.Một hệ điều hành bao gồm :
a. Hệ thống quản lý ttin, I/O
b. Hệ thống quản lý ttrình, bộ nhớ
c. a và b
d. a, b, c đều sai
3.Hệ điều hành MS-DOS có cấu trúc :
a. Đơn thể
b. Hạt nhân
c. Lớp
d. Máy ảo

CHƯƠNG 1:
ĐIỀU KHIỂN DỮ LIỆU

Mã chương: MH15-02

Giới thiệu:

Người sử dụng thì quan tâm đến cách đặt tên tập tin, các thao tác trên tập tin, cây thư mục...Nhưng đối người cài đặt thì quan tâm đến tập tin và thư mục được lưu trữ như thế nào, vùng nhớ trên đĩa được quản lý như thế nào và làm sao cho toàn bộ hệ thống làm việc hữu hiệu và tin cậy. Hệ thống tập tin được cài đặt trên đĩa. Để gia tăng hiệu quả trong việc truy xuất, mỗi đơn vị dữ liệu được truy xuất gọi là một khối. Một khối dữ liệu bao gồm một hoặc nhiều sector. Bộ phận tổ chức tập tin quản lý việc lưu trữ tập tin trên những khối vật lý bằng cách sử dụng các bảng có cấu trúc. Bài học này giúp chúng ta nắm đặc điểm cũng như ưu và khuyết điểm của các phương pháp tổ chức quản lý tập tin trên đĩa và một số vấn đề liên quan khác nhờ đó có thể hiểu được cách các hệ điều hành cụ thể quản lý tập tin như thế nào.Bài học này đòi hỏi những kiến thức về :mô hình tổ chức các tập tin và thư mục cũng và một số cấu trúc dữ liệu.

Mục Tiêu:

- Nắm được cách thức HĐH tổ chức lưu trữ và tìm kiếm dữ liệu dữ liệu trên hệ thống máy tính
- Nắm được các giai đoạn HĐH thực hiện điều khiển dữ liệu và sự phân công công việc giữa chương trình hệ thống (thuộc HĐH) và chương trình người dùng trong quá trình nhập-xuất dữ liệu.
- Rèn luyện khả năng tư duy, lập luận có tính khoa học

Nội Dung Chính:

1.Các phương pháp tổ chức và truy nhập dữ liệu

Mục tiêu:

- Nắm được cách thức HĐH tổ chức lưu trữ và tìm kiếm dữ liệu dữ liệu trên hệ thống máy tính

1.1.Bản quản lý thư mục tập tin

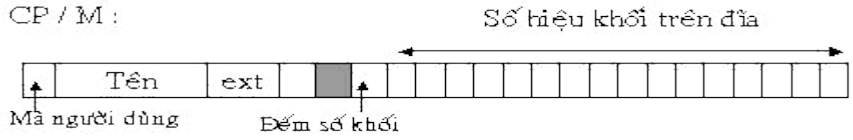
1.1.1.Khái niệm

Trước khi tập tin được đọc, tập tin phải được mở, để mở tập tin hệ thống phải biết đường dẫn do người sử dụng cung cấp và được định vị trong cấu trúc đầu vào thư mục (directory entry). Directory entry cung cấp các thông tin cần thiết để tìm kiếm các khối. Tuỳ thuộc vào mỗi hệ thống, thông tin là địa chỉ trên đĩa của toàn bộ tập tin, số hiệu của khối đầu tiên, hoặc là số I-node.

1.1.2. Cài đặt

Bảng này thường được cài đặt ở phần đầu của đĩa. Bảng là dãy các phần tử có kích thước xác định, mỗi phần tử được gọi là một entry. Mỗi entry sẽ lưu thông tin về tên, thuộc tính, vị trí lưu trữ .... của một tập tin hay thư mục.

Ví dụ quản lý thư mục trong CP/M :



Hình 9.1

1.2.Bản phân phối vùng nhớ

1.2.1. Khái niệm

Bảng này thường được sử dụn phối hợp với bảng quản lý thư mục tập tin, mục tiêu là cho biết vị trí khối vật lý của một tập tin hay thư mục nào đó nói khác đi là lưu giữ dãy các khối trên đĩa cấp phát cho tập tin lưu dữ liệu hay thư mục. Có một số phương pháp được cài đặt.

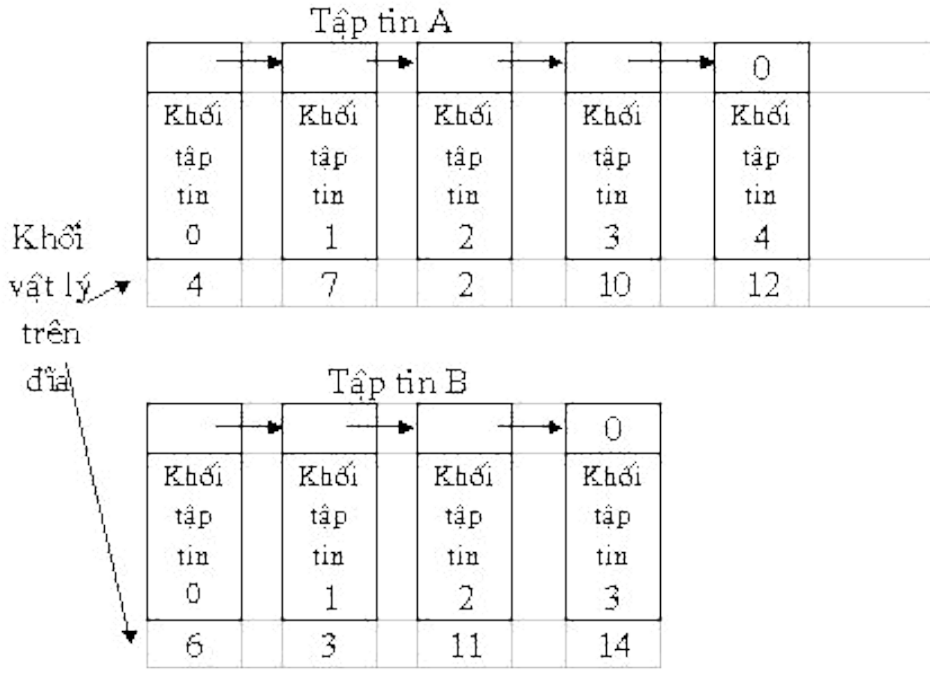
1.2.2 Các phương pháp

Định vị liên tiếp :

Lưu trữ tập tin trên dãy các khối liên tiếp.

Phương pháp này có 2 ưu điểm : thứ nhất, dể dàng cài đặt. Thứ hai, dể dàng thao tác vì toàn bộ tập tin được đọc từ đĩa bằng thao tác đơn giản không cần định vị lại.

Phương pháp này cũng có 2 khuyết điểm : không linh động trừ khi biết trước kích thước tối đa của tập tin. Sự phân mảnh trên đĩa, gây lãng phí lớn.

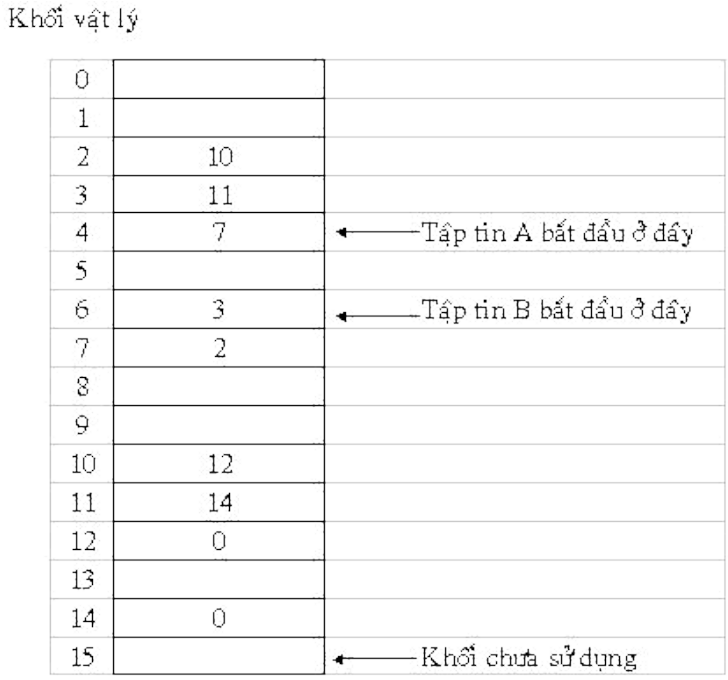


Hình 2.1 Định vị bằng danh sách liên kết

Mọi khối đều được cấp phát, không bị lãng phí trong trường hợp phân mảnh và directory entry chỉ cần chứa địa chỉ của khối đầu tiên.

Tuy nhiên khối dữ liệu bị thu hẹp lại và truy xuất ngẫu nhiên sẽ chậm.

Danh sách liên kết sử dụng index :



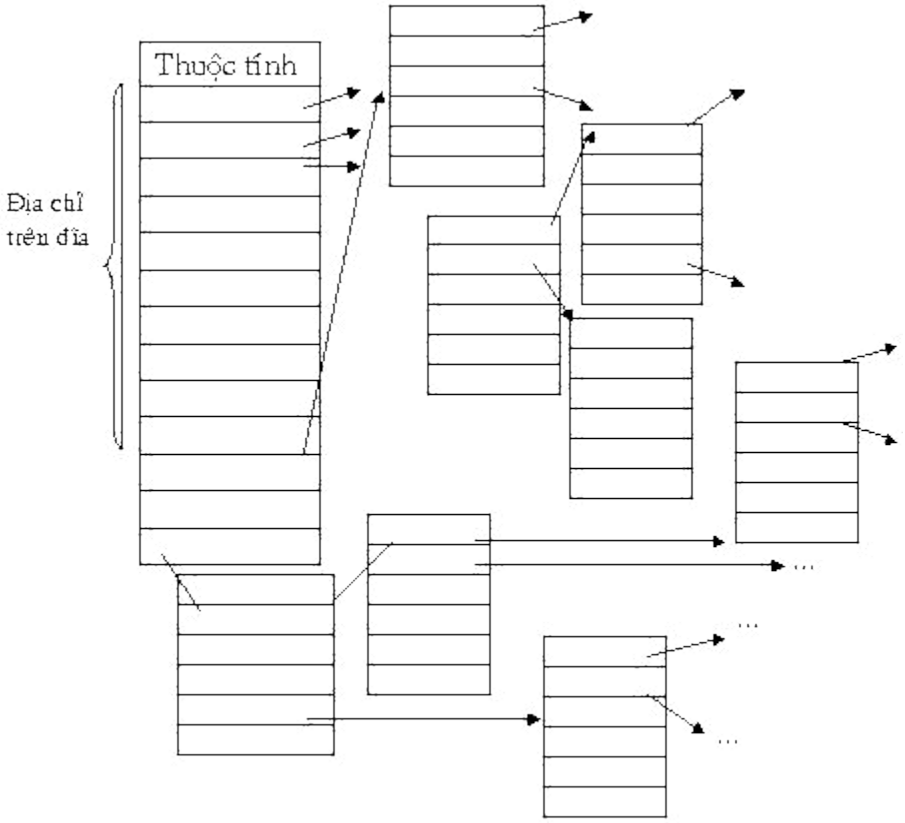
Hình 2.2.Bảng chỉ mục của danh sách liên kết

Tương tự như hai nhưng thay vì dùng con trỏ thì dùng một bảng index. Khi đó toàn bộ khối chỉ chứa dữ liệu. Truy xuất ngẫu nhiên sẽ dễ dàng hơn. Kích thước tập tin được mở rộng hơn. Hạn chế là bản này bị giới hạn bởi kích thước bộ nhớ .

I-nodes :

Một I-node bao gồm hai phần. Phần thứ nhất là thuộc tính của tập tin. Phần này lưu trữ các thông tin liên quan đến tập tin như kiểu, người sở hữu, kích thước, v.v...Phần thứ hai chứa địa chỉ của khối dữ liệu. Phần này chia làm hai phần nhỏ. Phần nhỏ thứ nhất bao gồm 10 phần tử, mỗi phần tử chứa địa chỉ khối dữ liệu của tập tin. Phần tử thứ 11 chứa địa chỉ gián tiếp cấp 1 (single indirect), chứa địa chỉ của một khối, trong khối đó chứa một bảng có thể từ 2^{10} đến 2^{32} phần tử mà mỗi phần tử mới chứa địa chỉ của khối dữ liệu. Phần tử thứ 12 chứa địa chỉ gián tiếp cấp 2 (double indirect), chứa địa chỉ của bảng các khối single indirect. Phần tử thứ 13 chứa địa chỉ gián tiếp cấp 3 (double indirect), chứa địa chỉ của bảng các khối double indirect.

Cách tổ chức này tương đối linh động. Phương pháp này hiệu quả trong trường hợp sử dụng để quán lý những hệ thống tập tin lớn. Hệ điều hành sử dụng phương pháp này là Unix (Ví dụ : BSD Unix)



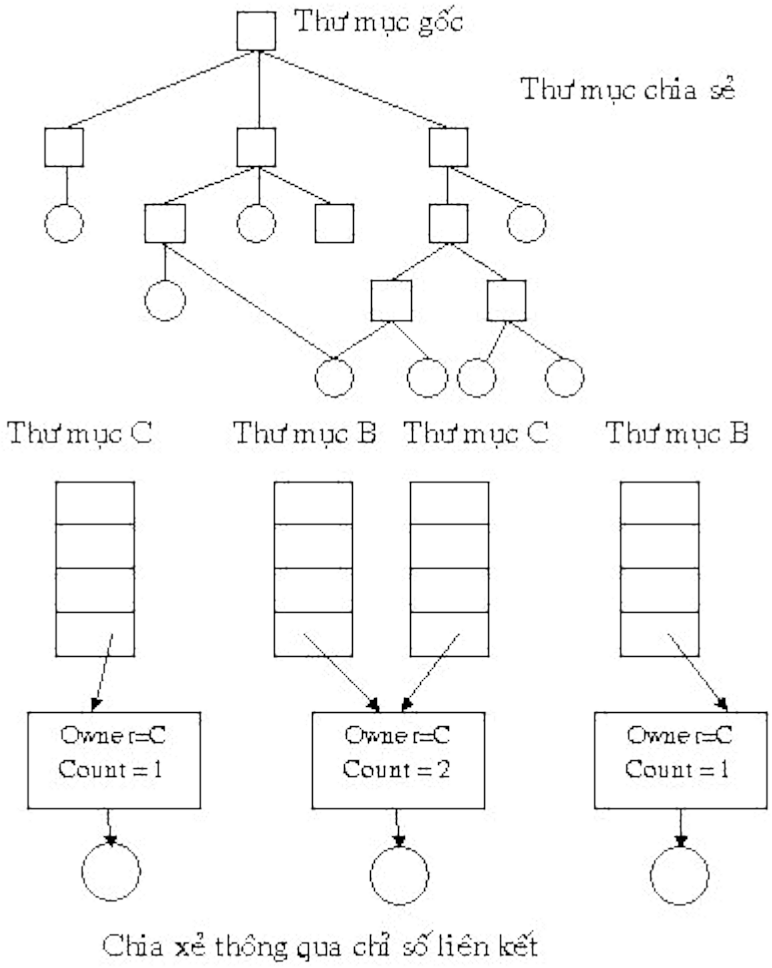
Hình 2.3 Cấu trúc của I-node

1.3.Tập tin chia sẻ

Khi có nhiều người sử dụng cùng làm việc trong một đề án, họ cần ***chia sẻ*** các tập tin. Cách chia sẻ thông thường là tập tin xuất hiện trong các thư mục là như nhau nghĩa là một tập tin có thể liên kết với nhiều thư mục khác nhau.

Để cài đặt được, khối đĩa không được liệt kê trong thư mục mà được thay thế bằng một cấu trúc dữ liệu, thư mục sẽ trỏ tới cấu trúc này. Một cách khác là hệ thống tạo một tập tin mới có kiểu LINK, tập tin mới này chỉ chứa đường dẫn của tập tin được liên kết, khi cần truy xuất sẽ dựa trên tập tin LINK để xác định tập tin cần truy xuất, phương pháp này gọi là liên kết hình thức. Mỗi phương pháp đều có những ưu và khuyết điểm riêng.

Ở phương pháp thứ nhất hệ thống biết được có bao nhiêu thư mục liên kết với tập tin nhờ vào chỉ số liên kết. Ở phương pháp thứ hai khi loại bỏ liên kết hình thức, tập tin không bị ảnh hưởng.



Hình 2.4: tập tin chia sẻ

1.4.Quản lý đĩa

Tập tin được lưu trữ trên đĩa, do đó việc quản trị đĩa là hết sức quan trọng trong việc cài đặt hệ thống tập tin. Có hai phương pháp lưu trữ : một là chứa tuần tự trên n byte liên tiếp, hai là tập tin được chia làm thành từng khối. Cách thứ nhất không hiệu quả khi truy xuất những tập tin có kích thước lớn, do đó hầu hết các hệ thống tập tin đều dùng khối có kích thước cố định.

1.4.1. Kích thước khối

Một vấn đề đặt ra là kích thước khối phải bằng bao nhiêu. Điều này phụ thuộc vào tổ chức của đĩa như số sector, số track, số cylinder. Nếu dùng một cylinder cho một khối cho một tập tin thì theo tính toán sẽ lãng phí đến 97% dung lượng đĩa. Nên thông thường mỗi tập tin thường được lưu trên một số khối. Ví dụ một đĩa có 32768 byte trên một track, thời gian quay là 16.67 msec, thời gian tìm kiếm trung bình là 30 msec thì thời gian tính bằng msec để đọc một khối kích thước k byte là :

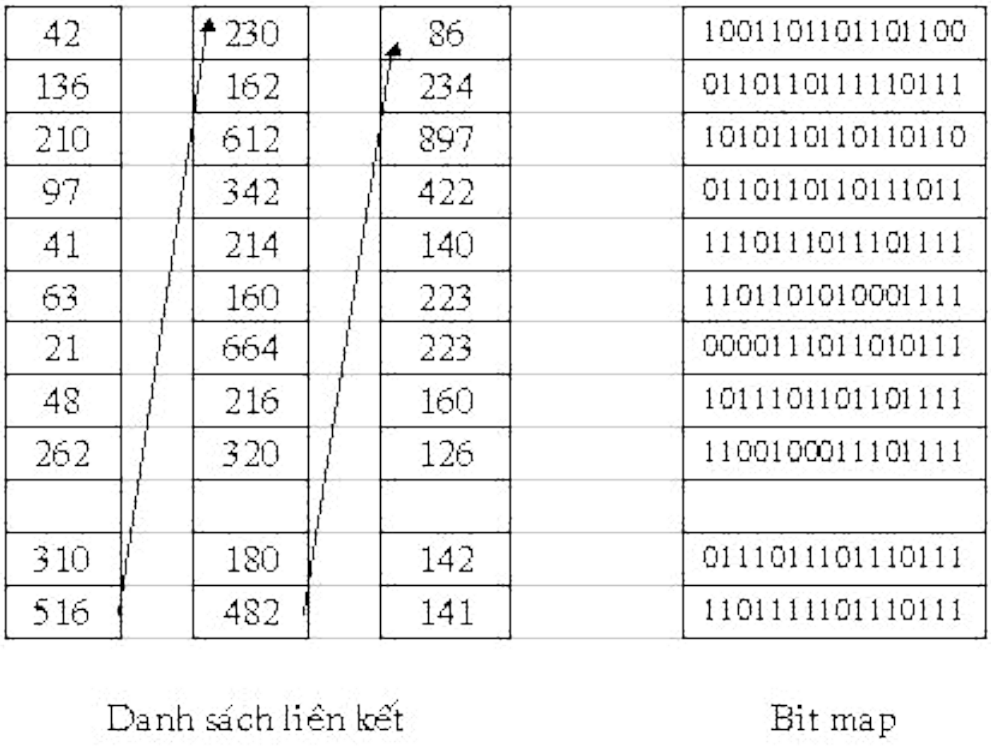
30 + 8.3 + (k/32768) x 16.67

Từ đó thống kê được kích thước khối thích hợp phải < 2K .

Thông thường kích thước khối là 512, 1K hay 2K.

1.4.2. Lưu giữa các khối trống

Có hai phương pháp. Một là sử dụng danh sách liên kết của khối đĩa. Mỗi khối chứa một số các địa chỉ các khối trống. Ví dụ một khối có kích thước 1 K có thể lưu trữ được 511 địa chỉ 16 bit. Một đĩa 20M cần khoảng 40 khối. Hai là, sử dụng bitmap. Một đĩa n khối sẽ được ánh xạ thành n bit với giá trị 1 là còn trống, giá trị 0 là đã lưu dữ liệu. Như vậy một đĩa 20M cần 20K bit để lưu trữ nghĩa là chỉ có khoảng 3 khối. Phương pháp thứ hai này thường được sử dụng hơn.



Hình 2.5: Hai phương pháp lưu giữ khối trống

1.5.Độ an toàn của hệ thống tập tin

Một hệ thống tập tin bị hỏng còn nguy hiểm hơn máy tính bị hỏng vì những hư hỏng trên thiết bị sẽ ít chi phí hơn là hệ thống tập tin vì nó ảnh hưởng đến các phần mềm trên đó. Hơn nữa hệ thống tập tin không thể chống lại được như hư hòng do phần cứng gây ra, vì vậy chúng phải cài đặt một số chức năng để bảo vệ.

1.5.1 Quản lý khối bị hỏng

Đĩa thường có những khối bị hỏng trong quá trình sử dụng đặc biệt đối với đĩa cứng vì khó kiểm tra được hết tất cả.

Có hai giải pháp : phần mềm và phần cứng.

Phần cứng là dùng một sector trên đĩa để lưu giữ danh sách các khối bị hỏng. Khi bộ kiểm soát tực hiện lần đầu tiên, nó đọc những khối bị hỏng

và dùng một khối thừa để lưu giữ. Từ đó không cho truy cập những khối hỏng nữa.

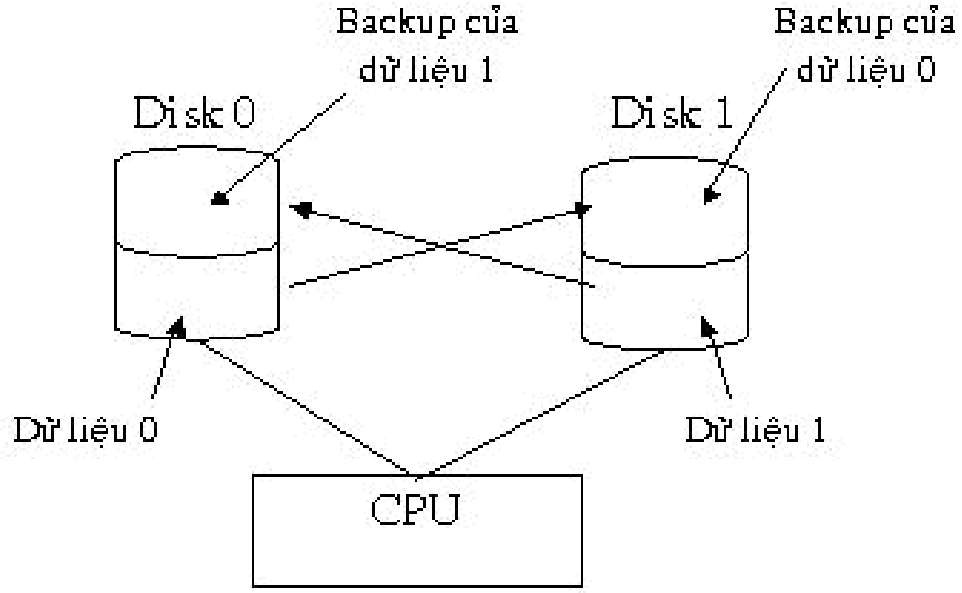
Phần mềm là hệ thống tập tin xây dựng một tập tin chứa các khối hỏng. Kỹ thuật này loại trừ chúng ra khỏi danh sách các khối trống, do đó nó sẽ không được cấp phát cho tập tin.

1.5.2.Backup

Mặc dù có các chiến lược quản lý các khối hỏng, nhưng một công việc hết sức quan trọng là phải backup tập tin thường xuyên.

Tập tin trên đĩa mềm được backup bằng cách chép lại toàn bộ qua một đĩa khác. Dữ liệu trên đĩa cứng nhỏ thì được backup trên các băng từ.

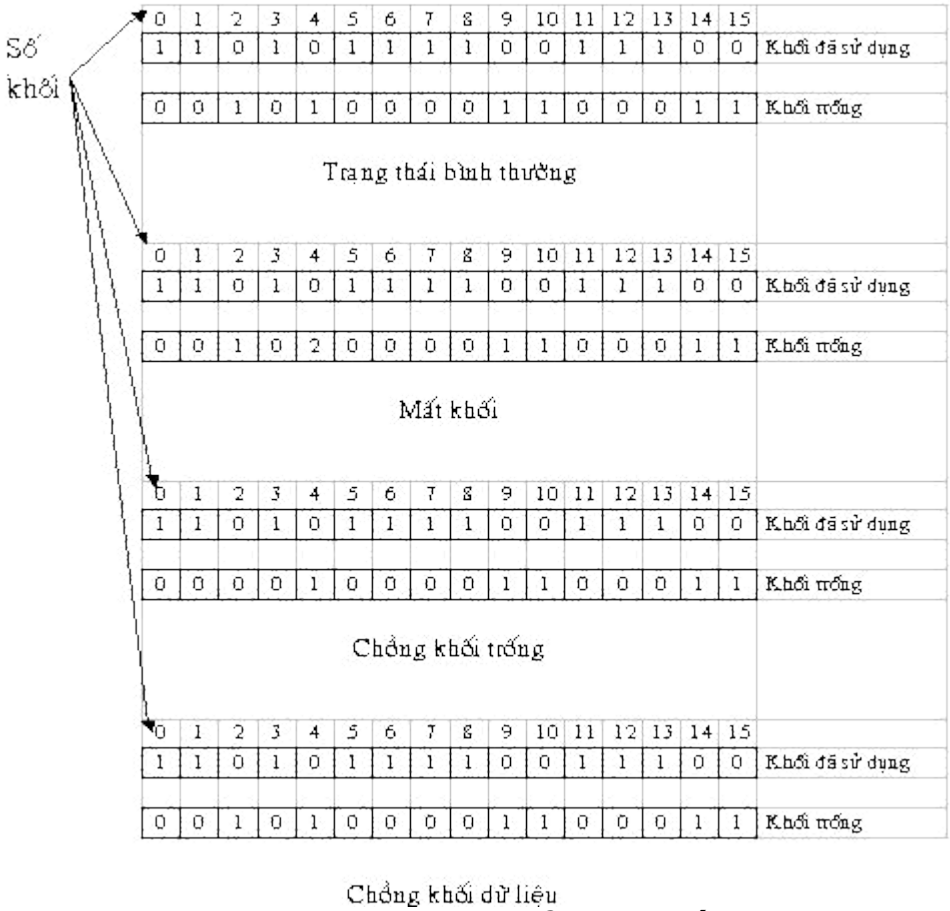
Đối với các đĩa cứng lớn, việc backup thường được tiến hành ngay trên nó. Một chiến lược dể cài đặt nhưng lãng phí một nữa đĩa là chia đĩa cứng làm hai phần một phần dữ liệu và một phần là backup. Mỗi tối, dữ liệu từ phần dữ liệu sẽ được chép sang phần backup.



Hình 2.6: Backup

1.5.3.Tính không đổi của hệ thống tập tin

Một vấn đề nữa về độ an toàn là ***tính không đổi***. Khi truy xuất một tập tin, trong quá trình thực hiện, nếu có xảy ra những sự cố làm hệ thống ngừng hoạt động đột ngột, lúc đó hàng loạt thông tin chưa được cập nhật lên đĩa. Vì vậy mỗi lân khởi động ,hệ thống sẽ thực hiện việc kiểm tra trên hai phần khối và tập tin. Việc kiểm tra thực hiện , khi phát hiện ra lỗi sẽ tiến hành sữa chữa cho các trường hợp cụ thể:



Hình 2.7 Trạng thái của hệ thống tập tin

2.Bản ghi và khối

Mục tiêu:

- phân biệt được bản ghi và khối

2.1.Bản ghi loogic và bản ghi vật lý

2.1.1.Bản ghi loogic và bản ghi vật lý

Một mặt, File được tổ chức thành các đơn vị dữ liệu để chương trình ứng dụng xử lý: đó là các bản ghi loogc(thường gọi tắt là bản ghi). Quy cách và nội dung của bản ghi lôgic được xác định theo chương trình ứng dụng.

Mặt khác, việc lưu trữ File trên vật dẫn ngoài tuân theo các quy tắc làm việc của Hệ điều hành đối với vật dẫn ngoài đó: File được xếp trên bộ nhớ ngoài thành các bản ghi vật lý (phổ biến hơn gọi là khối). Thông thường, khối là đơn vị bộ nhớ ngoài mà hệ điều hành thực hiện việc đọc/ ghi đối với File. Chẳng hạn trong MS-DOS, một cluste chính là một khối trên đĩa từ và File được lưu trữ trên một tập hợp các cluste của đĩa từ.

Một bài toán điển hình liên quan đến các khối trên đĩa từ là bài toán quản lý không gian đĩa để lắm bắt được trạng thái rỗi/bận của các khối trên đĩa để biết được khối nào rỗi (để phân phối cho nhu cầu mới), khối nào bận là khối đã chứa nội dung của một File (để tránh ghi đè lên nó).

Việc đọc/ ghi đối với một File cũng cần có được các thông tin trạng thái như vậy. Tồn tại một phương pháp giải quyết bài toán đó. Một phương pháp đơn giản là sử dụng bảng định vị File (File Allocation Table: FAT) mà MS-DOS sử dụng. Phương pháp phổ dụng hơn là phương pháp Bit map, trong đó người ta dùng một vùng, được gọi là Bit map, để trình bày tình trạng rỗi/bận của tất các các khối trên đĩa. Theo phương pháp này, mỗi khối trên đĩa được tương ứng với một bit trong vùng bit map và tình trạng rỗi/bận của khối đó được xác định bằng giá trị 0/1 của bit tươmg ứng.

2.1.2.Bản ghi theo tổ chức của File: Có ba dạng tổ chức bản ghi lôgic

Thông thường, có ba dạng bản phổ biến là dạng cố định, dạng động và dạng không xác định. Dạng của bản ghi của File dữ liệu sẽ quy định tới cách thức xử lý của hệ điều hành đối với File.

Dạng cố định (F): Mọi bản ghi trong File có độ dài cố định và như nhau (mỗi bản ghi có thể có dấu hiệu điều khiển). Làm việc với các File gồm các bản ghi dạng F rất tiện lợi, từ vị trí của bản ghi đầu tiên và số thứ tự của một bản ghi có thể nhận được vị trí của bản ghi đó. Việc định vị bản ghi theo số hiệu là hoàn toàn xác định. Mặt khác, các công việc chuẩn bị để xử lý các bản ghi dạng F là đơn giản.

Ví dụ: - Các bản gh trong một File có kiểu trong ngôn ngữ lập trình PASCAL thuộc dạng F.

Nếu bỏ phần cấu trúc, các bản ghi File DBF của FOXPRO có thể coi có độ dài cố định và là dạng F.

Dạng động (V): độ dài của bản ghi thay đổi từ bản ghi này cho tới bản ghi khác, song ngay khi xử lý bản ghi thì hệ điều hành đã biết độ dài của bản ghi đó: Trong một phần nội dung của bản ghi đã ghi nhận độ dài của bản ghi. Tùy thuộc vào độ dài mỗi bản ghi có thể chuẩn bị các công việc liên quan để xử lý chúng, chẳng hạn việc tách các bản ghi từ một khối sau khi đọc từ vật dẫn ngoài vào bộ nhớ trong.

Ví dụ: - File chương trình BASIC có thể coi thuộc loại này: mỗi câu lệnh gồm có số liệu lệnh, định vị câu lệnh ngay tiếp theo và nội dung dòng lệnh, từ định vị câu lệnh ngay tiếp theo có ngay độ dài dòng lệnh BASIC.

Dạng không xác định (U): Độ dài bản ghi không thể xác định, cuối mỗi bản ghi mới có dấu hiệu kết thúc bản ghi. Việc xử lý các File mà bản ghi thuộc dạng U nói chung có tính tự động hóa thấp hơn so với File gồm các bản ghi dạng F hay V.

Ví dụ: - File FPT trong FOXPRO chứa nội dung các trường Memory của một tệp DBF. Độ dài bản ghi không đươc biết: Nội dung của trường Memory được kết thúc bởi dấu hiệu kết thúc File (^Z).

2.2.Kết khối và tách khối

Một khối có thể chứa một hoặc một vài bản ghi và ngược lại, một bản ghi có thể được xếp trên một hoặc một số khối. Như vậy tồn tại mối quan hệ giữa khối với bản ghi và điều đó liên quan đến vấn đề xác định bản ghi theo khối.

Việc tổ chức File trên vật dẫn ngoài theo các khối là công việc của Hệ điều hành (do các chương trình của phương pháp truy nhập đảm nhận) và như đã nói là cần đảm bảo tính độc lập với chương trình người dùng cho nên việc đưa một khối vào bộ nhớ trong hoặc đưa dữ liệu lên một khối là do hệ điều hành đảm nhận.Ta có thể gọi quá trình đó là quá trình vào-ra vật lý.

Sau khi hệ điều hành đã đưa một khối vào bộ nhớ trong, cần phải xá định bản ghi hiện thời để chương trình người dùng xử lý. Đó là quá trình tách khối.

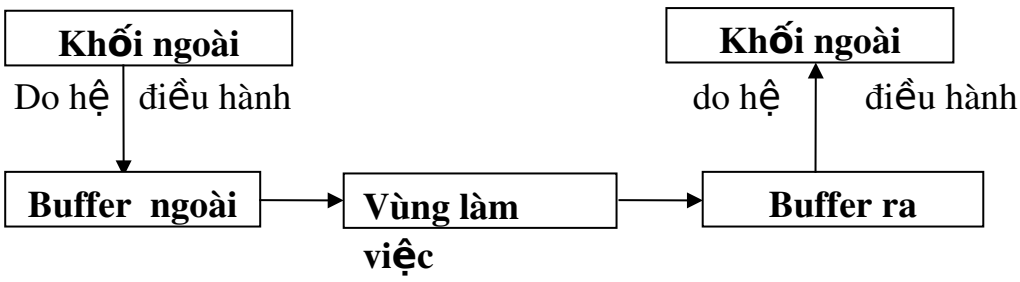
Tách khối là quá trình từ các khối đưa ra được các bản ghi cần tìm có liên quan đến khối đó. Quá trình này diễn ra ngay sau khi hệ điều hành đã đọc một khối vào bộ nhớ trong và trước khi chương trình người dùng xử lý bản ghi.

Tùy thuộc vào phương pháp truy nhập dữ liệu mà tách khối hoặc do hệ điều hành hoặc do chính chương trình người dùng đảm nhận.

Sau khi chương trình người dùng chuẩn bị xong nội dung bản ghi, thông tin trên bản ghi đó đã đúng như yêu cầu của người dùng, cần đưa nó lên vật dẫn ngoài để lưu trữ nâu dài. Như đã biết, hệ điều hành ghi thông tin lên vật dẫn ngoài theo đơn vị là khối, vì vậy bản ghi nói trên phải được xếp vào một khối tương ứng (quá trình đó gọi là kết khối). Khi khối đã đầy đủ thông tin được xử lý thì hệ điều hành cần đặt đúng khối đã có vào vị trí đã dành cho nó trên vật dẫn ngoài.

Về hình thức, kết khối là quá trình ngược lại với quá trình tách khối. Kết khối diễn ra sau khi chương trình người dùng chuẩn bị xong nội dung bản ghi và đưa bản ghi đó vào khối để đưa ra vật dẫn ngoài.

Chương trình người dũng xử lý dữ liệu những vùng bộ nhớ theo quy định của chương trình, được gọi là vùng làm việc. Hệ điều hành đọc khối vào các vùng nhớ trung giang được gọi là vùng đệm vào (buffer vào) trước khi dữ liệu được chương trình xử lý. Sau khi chương trình xử lý dữ liệu xong, bản ghi đã hoàn thiện được kết khối vào các vùng nhớ đệm ra (buffer ra) trước khi được hệ điều hành đưa ra vật dẫn ngoài.



Bộ nhớ trong

Hình 2.8.Sơ đồ tách khối/kết khối

Sơ đồ trong hình 2.8 diễn tả sơ lược về hai quá trình trên. Trong sơ đồ này, giai đoạn đọc vật lý (khi vào) và ghi vật lý (khi ra) do chương trình của phương pháp truy nhập phải đảm nhận. Giai đoạn tách khối và kết khối hoặc do hệ điều hành đảm nhận hoặc do chương trình người dùng đảm nhận tùy thuộc vào File dữ liệu nói trên được mở làm việc theo phương pháp truy nhập vào. Tùy thuộc vào phương pháp truy nhập mà các quá trình nói trên được thực hiện theo các cách thức khác nhau như trình bày ở các mục sau.

Theo sơ đồ trên đây, ta có thể nhận thấy rằng mỗi phương pháp tổ chức và truy nhập dữ liệu bao gồm một số thành phần cơ bản như sau (mô đun chương trình có thể được phát triển thành nhóm mô đun chương trình):

- Mô đun chương trình đảm bảo chức năng tổ chức lưu trữ và định vị trên vật dẫn ngoài;
- Mô đun chương trình đảm bảo vào/ra mỗi khối (bản ghi vvatj lý) đối với mỗi khối xác định;
- Mô đun chương trình đảm bảo tách/kết khối theo bản ghi đối với File xác định.

3. Điều khiển buffer(điều khiển phòng đệm)

Mục tiêu:

- Nắm được các giai đoạn HĐH thực hiện điều khiển dữ liệu và sự phân công công việc giữa chương trình hệ thống (thuộc HĐH) .

Đặc trưng cơ bản của thiết bị ngoại vi là tốc độ hoạt động nhỏ hơn nhiều lần so với tốc độ hoạt động của processor. Để thực hiện một phép vào ra hệ thống phải kích hoạt thiết bị, chờ đợi thiết bị đạt trạng thái thích hợp (Ví dụ như máy in phải chờ nóng) và sau đó chờ đợi công việc được thực hiện. Chính vì vậy phần lớn các thiết bị vào ra làm việc với từng khối dữ liệu chứ không phải từng byte riêng lẻ. Để đảm bảo năng suất, hệ thống cần phải

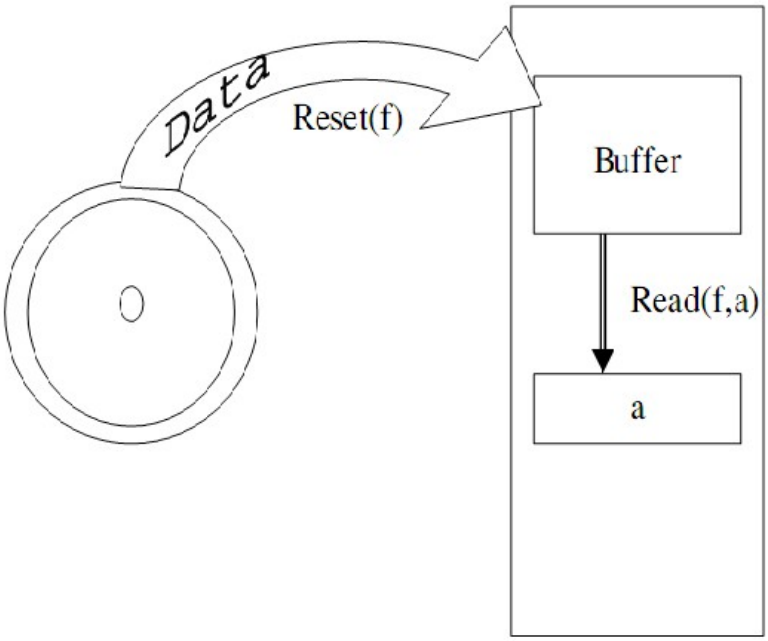
+ Cố gắng thực hiện song song công việc vào ra với các phép xử lý thông tin khác
+ Giảm số lượng các phép trao đổi vào ra vật lý
+ Thực hiện trước các phép nhập dữ liệu
- Như vậy người ta phải sử dụng phòng đệm để nâng cao năng suất
+ Phòng đệm của hệ điều hành là một vùng nhớ dùng để lưu trữ tạm thời

các thông tin phục cho các phép vào ra.

+ Ngoài ra còn có phòng đệm của thiết bị không phụ thuộc vào hệ điều
hành gọi là phòng đệm kỹ thuật. Ví dụ phòng đệm của máy in.

- Ví dụ

```
Assign(f,'f1.txt');
Reset(f);
Read(f,a);
```



Hình 2.9.Phòng đệp của máy in

Khi thực hiện Reset(f) thì hệ thống đã đưa dữ liệu từ đĩa lên vùng đệm.

Khi chương trình muốn đọc dữ liệu từ tệp vào biến a thì hệ thống chỉ cần lấy dữ liệu từ vùng đệm thay cho việc đọc tệp.

Giả thiết mỗi lần truy nhập đĩa mất 0,01 giây, kích thước vùng đệm là 512 bytes và thời gian truy nhập vào bộ nhớ là rất nhỏ (so với 0,01)

Số byte cần đọc	**Không có vùng đệm**	**Có vùng đệm**
1B	0,01''	0,01''
512B	5'' = 512x0.01	0,01''
5KB	50'' = 10x5	0.1'' = 10x0.01
50KB	8' = 10x50	1'' = 10x01

- Phân loại phòng đệm

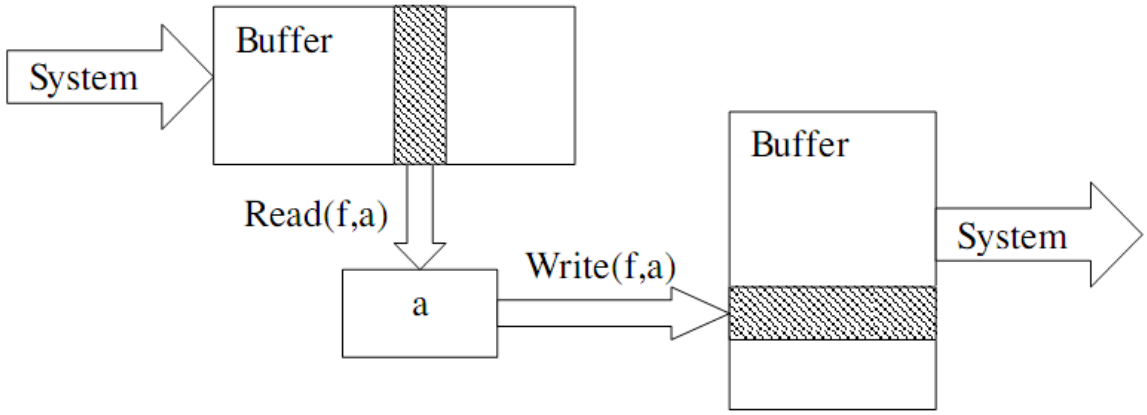
3.1.Phòng đệm trung chuyển

- Là phòng đệm thuần tuý lưu trữ tạm thời các phép phục vụ vào ra.
- Phòng đệm này có hai loại

+ Phòng đệm vào là phòng đệm chỉ dùng để nhập thông tin. Trong hệ thống sẽ có lệnh để đ-a thông tin vào phòng đệm (đọc vật lý).
Khi gặp chỉ thị đọc (READ), thông tin sẽ được tách và chuyển từ phòng đệm vào các địa chỉ tương ứng trong chương trình ứng dụng. Như vậy, mỗi giá trị được lưu trữ ở hai nơi trong bộ nhớ (một ở phòng đệm và một ở vùng bộ nhớ trong chương trình ứng dụng). Khi giá trị cuối cùng của phòng đệm vào được lấy ra thì phòng đệm được giải phóng (rỗng) và hệ thống đưa thông tin mới vào phòng đệm trong thời gian ngắn nhất có thể.

Để giảm thời gian chờ đợi, hệ thống có thể tổ chức nhiều phòng đệm vào, khi hết thông tin ở một phòng đệm, hệ thống sẽ chuyển sang phòng đệm khác.

+ Phòng đệm ra là phòng đệm để ghi thông tin. Trong hệ thống có lệnh để giải phóng phòng đệm (ghi vật lý). Khi có chỉ thị ghi (WRITE), thông tin
được đưa vào phòng đệm. Khi phòng đệm ra đầy, hệ thống sẽ đưa thông tin ra thiết bị ngoại vi. Hệ thống cũng có thể tổ chức nhiều phòng đệm ra.



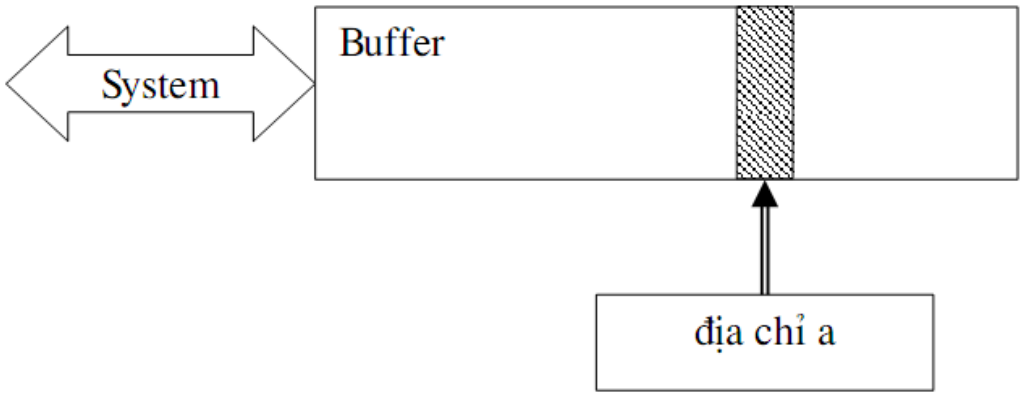
Hình 2.10.Phòng đệp trung chuyển

- ưu điểm:
  + Đơn giản
  + Có hệ số song song cao vì tốc độ giải phóng vùng đệm lớn
  + Có tính chất vạn năng, thích ứng với mọi phương pháp truy nhập
- Nhược điểm:
  + Tốn bộ nhớ
  + Tốn thời gian để trao đổi thông tin trong bộ nhớ

3.2.Phòng đệm xử lý

Thông tin được xử lý ngay trong phòng đệm không ghi lại vào nơi khác
trong bộ nhớ. Chỉ thị đọc xác định địa chỉ thông tin chứ không cung cấp thông

tin chứ không cung cấp giá trị.

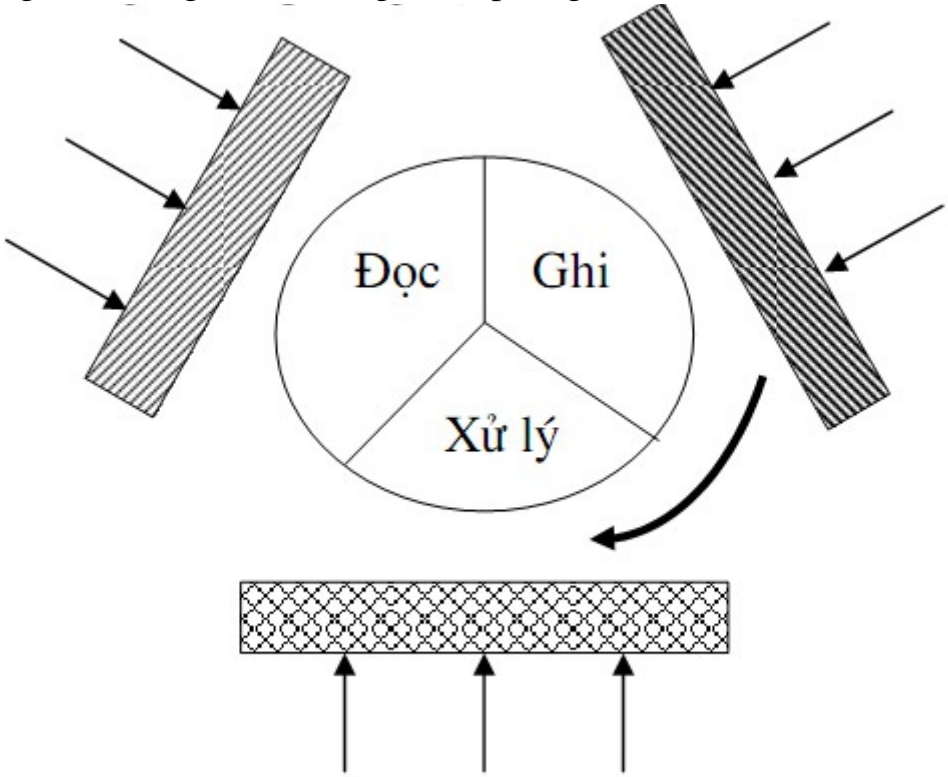


Hình 2.11.Phòng đệp xử lý

- ưu điểm:
  + Tiết kiệm bộ nhớ
  + Không mất thời gian chuyển thông tin ở bộ nhớ trong, thích hợp khi cần kích thước bản ghi dữ liệu lớn.
- Nhược điểm:
  + Tính vạn năng không cao
  + Hệ số song song thấp

3.3.Phòng đệm vòng tròn

Phòng đệm vòng tròn thường có ba phòng đệm



Hình 2.12.Phòng đệp vòng tròn

- Sau một khoảng thời gian vai trò của ba phòng đệm được thay đổi cho nhau.
- với ưu điểm:

+ Có sự đồng bộ giữa đọc, ghi và xử lý (ba quá trình được thực hiện song song).

+ Thường áp dụng cho hệ cơ sở dữ liệu và hữu dụng nhất khi lượng thông tin vào bằng lượng thông tin ra.

4. Quy trình chung điều khiển nhập-xuất

Mục Tiêu:

- Nắm được sự phân công công việc giữa chương trình hệ thống (thuộc HĐH) và chương trình người dùng trong quá trình nhập-xuất dữ liệu.

Là cấp thấp nhất chứa các trình điều khiển thiết bị và các bộ quản lý ngắt để chuyển thông tin giữa bộ nhớ chính và hệ thống đĩa. Trình điều khiển thiết bị thường viết các mẫu bit xác định tới các vị trí trong bộ nhớ của bộ điều khiển nhập/xuất để báo với bộ điều khiển vị trí trên thiết bị nào và hoạt động gì xảy ra.

Hệ thống quản lý nhập/xuất được tổ chức theo từng lớp, mỗi lớp có một chức năng nhất định và các lớp có giao tiếp với nhau như sơ đồ sau :

CÁC LỚP	CHỨC NĂNG NHẬP/XUẤT
Xử lý của người dùng	Tạo lời gọi nhập/xuất, định dạng nhập/xuất
Phần mềm độc lập thiết bị	Đặt tên, bảo vệ, tổ chức khối, bộ đệm, định vị
Điều khiển thiết bị	Thiết lập thanh ghi thiết bị, kiểm tra trạng thái
Kiểm soát ngắt	Báo cho driver khi nhập/xuất hoàn tất
Phần cứng	Thực hiện thao tác nhập/xuất

Ví dụ: Trong một chương trình ứng dụng, người dùng muốn đọc một khối từ một tập tin, hệ điều hành được kích hoạt để thực hiện yêu cầu này. Phần mềm độc lập thiết bị tìm kiếm trong cache, nếu khối cần đọc không có sẵn, nó sẽ gọi chương trình điều khiển thiết bị gửi yêu cầu đến phần cứng. Tiến trình bị ngưng lại cho đến khi thao tác đĩa hoàn tất. Khi

thao tác này hoàn tất, phần cứng phát sinh một ngắt. Bộ phận kiểm soát ngắt kiểm tra biến cố này, ghi nhận trạng thái của thiết bị và đánh thức tiến trình bị ngưng để chấm dứt yêu cầu I/O và cho tiến trình của người sử dụng tiếp tục thực hiện.[TAN]

4.1.Phần cứng nhập/xuất

Có nhiều cách nhìn khác nhau về phần cứng nhập/xuất. Các kỹ sư điện tử thì nhìn dưới góc độ là các thiết bị như IC, dây dẫn, bộ nguồn, motor v.v….Các lập trình viên thì nhìn chúng dưới góc độ phần mềm - những lệnh nào thiết bị chấp nhận, chúng sẽ thực hiện những chức năng nào, và thông báo lỗi của chúng bao gồm những gì, nghĩa là chúng ta quan tâm đến lập trình thiết bị chứ không phải các thiết bị này hoạt động như thế nào mặc dù khía cạnh này có liên quan mật thiết với các thao tác bên trong của chúng. Phần này chúng ta đề cập đến một số khái niệm về phần cứng I/O liên quan đến khía cạnh lập trình.

4.1.1.Thiết bị I/O

Các thiết bị nhập xuất có thể chia tương đối thành hai loại là thiết bị khối và thiết bị tuần tự.

Thiết bị khối là thiết bị mà thông tin được lưu trữ trong những khối có kích thước cố định và được định vị bởi địa chỉ. Kích thước thông thường của một khối là khoảng từ 128 bytes đến 1024 bytes. Đặc điểm của thiết bị khối là chúng có thể được truy xuất (đọc hoặc ghi) từng khối riêng biệt, và chương trình có thể truy xuất một khối bất kỳ nào đó. Đĩa là một ví dụ cho loại thiết bị khối.

Một dạng thiết bị thứ hai là thiết bị tuần tự. Ở dạng thiết bị này, việc gửi và nhận thông tin dựa trên là chuỗi các bits, không có xác định địa chỉ và không thể thực hiện thao tác seek được. Màn hình, bàn phím, máy in, card mạng, chuột, và các loại thiết bị khác không phải dạng đĩa là thiết bị tuần tự.

Việc phân chia các lớp như trên không hoàn toàn tối ưu, một số các thiết bị không phù hợp với hai lớp trên, ví dụ : đồng hồ, bộ nhớ màn hình v.v...không thực hiện theo cơ chế tuần tự các bits. Ngoài ra, người ta còn phân loại các thiết bị I/O dưới một tiêu chuẩn khác :

Thiết bị tương tác được với con người : dùng để giao tiếp giữa người và máy. Ví dụ : màn hình, bàn phím, chuột, máy in ...

Thiết bị tương tác trong hệ thống máy tính là các thiết bị giao tiếp với nhau. Ví dụ : đĩa, băng từ, card giao tiếp...

Thiết bị truyền thông : như modem...

Những điểm khác nhau giữa các thiết bị I/O gồm :

Tốc độ truyền dữ liệu , ví dụ bàn phím : 0.01 KB/s, chuột 0.02 KB/s ...

Công dụng.

Đơn vị truyền dữ liệu (khối hoặc ký tự).
Biểu diễn dữ liệu, điều này tùy thuộc vào từng thiết bị cụ thể.
Tình trạng lỗi : nguyên nhân gây ra lỗi, cách mà chúng báo về...

4.1.2.Tổ chức của chức năng I/O

Có ba cách để thực hiện I/O :

Một là, bộ xử lý phát sinh một lệnh I/O đến các đơn vị I/O, sau đó, nó chờ trong trạng thái "busy" cho đến khi thao tác này hoàn tất trước khi tiếp tục xử lý.

Hai là, bộ xử lý phát sinh một lệnh I/O đến các đơn vị I/O, sau đó, nó tiếp tục việc xử lý cho tới khi nhận được một ngắt từ đơn vị I/O báo là đã hoàn tất, nó tạm ngưng việc xử lý hiện tại để chuyển qua xử lý ngắt.

Ba là, sử dụng cơ chế DMA (như được đề cập ở sau)

Các bước tiến hóa của chức năng I/O :

Bộ xử lý kiểm soát trực tiếp các thiết bị ngoại vi.

Hệ thống có thêm bộ điều khiển thiết bị. Bộ xử lý sử dụng cách thực hiện nhập xuất thứ nhất. Theo cách này bộ xử lý được tách rời khỏi các mô tả chi tiết của các thiết bị ngoại vi.

Bộ xử lý sử dụng thêm cơ chế ngắt.

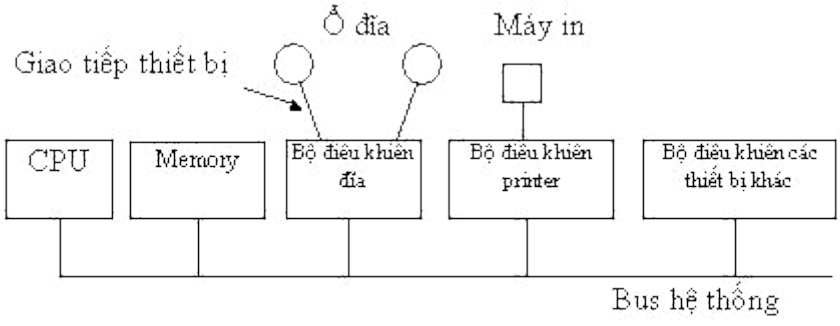
Sử dụng cơ chế DMA, bộ xử lý truy xuất những dữ liệu I/O trực tiếp trong bộ nhớ chính.

4.1.3.Bộ điều khiển thiết bị

Một đơn vị bị nhập xuất thường được chia làm hai thành phần chính là thành phần cơ và thành phần điện tử. Thành phần điện tử được gọi là bộ phận điều khiển thiết bị hay bộ tương thích, trong các máy vi tính thường được gọi là card giao tiếp. Thành phần cơ chính là bản thân thiết bị.

Một bộ phận điều khiển thường có bộ phận kết nối trên chúng để có thể gắn thiết bị lên đó. Một bộ phận điều khiển có thể quản lý được hai, bốn hay thậm chí tám thiết bị khác nhau. Nếu giao tiếp giữa thiết bị và bộ phận điều khiển là các chuẩn như ANSI, IEEE hay ISO thì nhà sản xuất thiết bị và bộ điều khiển phải tuân theo chuẩn đó, ví dụ : bộ điều khiển đĩa được theo chuẩn giao tiếp của IBM.

Giao tiếp giữa bộ điều khiển và thiết bị là giao tiếp ở mức thấp.



Hình 2.13:Sự kết nối giữa CPU, bộ nhớ, bộ điều khiển và các thiết bị nhập xuất

Chức năng của bộ điều khiển là giao tiếp với hệ điều hành vì hệ điều hành không thể truy xuất trực tiếp với thiết bị. Việc thông tin thông qua hệ thống đường truyền gọi là bus.

Công việc của bộ điều khiển là chuyển đổi dãy các bit tuần tự trong một khối các byte và thực hiện sửa chữa nếu cần thiết. Thông thường khối các byte được tổ chức thành từng bit và đặt trong buffer của bộ điều khiển. Sau khi thực hiện checksum nội dung của buffer sẽ được chuyển vào bộ nhớ chính. Ví dụ : bộ điều khiển cho màn hình đọc các byte của ký tự để hiển thị trong bộ nhớ và tổ chức các tín hiệu để điều khiển các tia của CRT để xuất trên màn ảnh bằng cách quét các tia dọc và ngang. Nếu không có bộ điều khiển, lập trình viên hệ điều hành phải tạo thêm chương trình điều khiển tín hiệu analog cho đèn hình. Với bộ điều khiển , hệ điều hành chỉ cần khởi động chúng với một số tham số như số ký tự trên một dòng, số dòng trên màn hình và bộ điều khiển sẽ thực hiện điều khiển các tia.

Mỗi bộ điều khiển có một số thanh ghi để liên lạc với CPU. Trên một số máy tính, các thanh ghi này là một phần của bộ nhớ chính tại một địa chỉ xác định gọi là ánh xạ bộ nhớ nhập xuất. Hệ máy PC dành ra một vùng địa chỉ đặc biệt gọi là địa chỉ nhập xuất và trong đó được chia làm nhiều đoạn, mỗi đoạn cho một loại thiết bị như sau :

Bộ điều khiển nhập/xuất	Địa chỉ nhập/xuất	Vectơ ngắt
Đồng hồ	040 - 043	8
Bàn phím	060 - 063	9
RS232 phụ	2F8 - 2FF	11
Đĩa cứng	320 - 32F	13
Máy in	378 - 37F	15
Màn hình mono	380 - 3BF	-
Màn hình màu	3D0 - 3DF	-
Đĩa mềm	3F0 - 3F7	14

RS232 chính	3F8 - 3FF	12

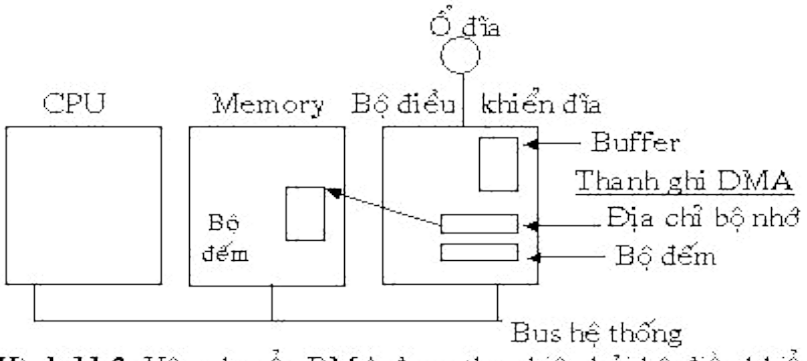
Hệ điều hành thực hiện nhập xuất bằng cách ghi lệnh lên các thanh ghi của bộ điều khiển. Ví dụ : bộ điều khiển đĩa mềm của IBMPC chấp nhận 15 lệnh khác nhau như : READ, WRITE, SEEK, FORMAT, RECALIBRATE, một số lệnh có tham số và các tham số cũng được nạp vào thanh ghi. Khi một lệnh đã được chấp nhận, CPU sẽ rời bộ điều khiển để thực hiện công việc khác. Sau khi thực hiện xong, bộ điều khiển phát sinh một ngắt để báo hiệu cho CPU biết và đến lấy kết quả được lưu giữ trong các thanh ghi.

4.1.4.DMA (Direct Memory Access)

Đa số các loại thiết bị, đặc biệt là các thiết bị dạng khối, hỗ trợ cơ chế DMA (direct memory access). Để hiểu về cơ chế này, trước hết phải xem xét quá trình đọc đĩa mà không có DMA. Trước tiên, bộ điều khiển đọc tuần tự các khối trên đĩa, từng bit từng bit cho tới khi toàn bộ khối được đưa vào buffer của bộ điều khiển. Sau đó máy tính thực hiện checksum để đảm bảo không có lỗi xảy ra. Tiếp theo bộ điều khiển tạo ra một ngắt để báo cho CPU biết. CPU đến lấy dữ liệu trong buffer chuyển về bộ nhớ chính bằng cách tạo một vòng lặp đọc lần lượt từng byte. Thao tác này làm lãng phí thời gian của CPU. Do đó để tối ưu, người ta đưa ra cơ chế DMA.

Cơ chế DMA giúp cho CPU không bị lãng phí thời gian. Khi sử dụng, CPU gửi cho bộ điều khiển một số các thông số như địa chỉ trên đĩa của khối, địa chỉ trong bộ nhớ nơi định vị khối, số lượng byte dữ liệu để chuyển.

Sau khi bộ điều khiển đã đọc toàn bộ dữ liệu từ thiết bị vào buffer của nó và kiểm tra checksum. Bộ điều khiển chuyển byte đầu tiên vào bộ nhớ chính tại địa chỉ được mô tả bởi địa chỉ bộ nhớ DMA. Sau đó nó tăng địa chỉ DMA và giảm số bytes phải chuyển. Quá trình này lập cho tới khi số bytes phải chuyển bằng 0, và bộ điều khiển tạo một ngắt. Như vậy không cần phải copy khối vào trong bộ nhớ, nó đã hiện hữu trong bộ nhớ.



Hình 11.2 Vận chuyển DMA được thực hiện bởi bộ điều khiển

Hình 2.14: Vận chuyển DMA được thực hiện bởi bộ điều khiển

4.2.Phần mềm nhập/xuất

Mục tiêu chung của thiết bị logic là để biểu diễn. Thiết bị logic được tổ chức thành nhiều lớp. Lớp dưới cùng giao tiếp với phần cứng, lớp trên cùng giao tiếp tốt, thân thiện với người sử dụng. Khái niệm then chốt của thiết bị logic là độc lập thiết bị, ví dụ : có thể viết chương trình truy xuất file trên đĩa mềm hay đĩa cứng mà không cần phải mô tả lại chương trình cho từng loại thiết bị. Ngoài ra, thiết bị logic phải có khả năng kiểm soát lỗi. Thiết bị logic được tổ chức thành bốn lớp : Kiểm soát lỗi, điều khiển thiết bị, phần mềm hệ điều hành độc lập thiết bị, phần mềm mức người sử dụng.

4.2.1 Kiểm soát ngắt

Ngắt là một hiện tượng phức tạp. Nó phải cần được che dấu sâu trong hệ điều hành, và một phần ít của hệ thống biết về chúng. Cách tốt nhất để che dấu chúng là hệ điều hành có mọi tiến trình thực hiện thao tác nhập xuất cho tới khi hoàn tất mới tạo ra một ngắt. Tiến trình có thể tự khóa lại bằng cách thực hiện lệnh WAIT theo một biến điều kiện hoặc RECEIVE theo một thông điệp.

Khi một ngắt xảy ra, hàm xử lý ngắt khởi tạo một tiến trình mới để xử lý ngắt. Nó sẽ thực hiện một tín hiệu trên biến điều kiện và gửi những thông điệp đến cho các tiến trình bị khóa. Tổng quát, chức năng của ngắt là làm cho một tiến trình đang bị khóa được thi hành trở lại.

4.2.2 Điều khiển thiết bị (device drivers)

Tất cả các đoạn mã độc lập thiết bị đều được chuyển đến device drivers. Mỗi device drivers kiểm soát mỗi loại thiết bị, nhưng cũng có khi là một tập hợp các thiết bị liên quan mật thiết với nhau.

Device drivers phát ra các chỉ thị và kiểm tra xem chỉ thị đó có được thực hiện chính xác không. Ví dụ, driver của đĩa là phần duy nhất của hệ điều hành kiểm soát bộ điều khiển đĩa. Nó quản lý sectors, tracks, cylinders, head, chuyển động, interleave, và các thành phần khác giúp cho các thao tác đĩa được thực hiện tốt.

Chức năng của device drivers là nhận những yêu cầu trừu tượng từ phần mềm nhập/xuất độc lập thiết bị ở lớp trên, và giám sát yêu cầu này thực hiện. Nếu driver đang rảnh, nó sẽ thực hiện ngay yêu cầu, ngược lại, yêu cầu đó sẽ được đưa vào hàng đợi.

Ví dụ, bước đầu tiên của yêu cầu nhập/xuất đĩa là chuyển từ trừu tượng thành cụ thể. Driver của đĩa phải biết khối nào cần đọc, kiểm tra sự hoạt động của motor đĩa, xác định vị trí của đầu đọc đã đúng chưa v.v…

Nghĩa là device drivers phải xác định được những thao tác nào của bộ điều khiển phải thi hành và theo trình tự nào. Một khi đã xác định được chỉ

thị cho bộ điều khiển, nó bắt đầu thực hiện bằng cách chuyển lệnh vào thanh ghi của bộ điều khiển thiết bị. Bộ điều khiển có thể nhận một hay nhiều chỉ thị liên tiếp và sau đó tự nó thực hiện không cần sự trợ giúp của hệ điều hành. Trong khi lệnh thực hiện. Có hai trường hợp xảy ra : Một là device drivers phải chờ cho tới khi bộ điều khiển thực hiện xong bằng cách tự khóa lại cho tới khi một ngắt phát sinh mở khóa cho nó. Hai là, hệ điều hành chấm dứt mà không chờ, vì vậy driver không cần thiết phải khóa.

Sau khi hệ điều hành hoàn tất việc kiểm tra lỗi và nếu mọi thứ đều ổn driver sẽ chuyển dữ liệu cho phần mềm độc lập thiết bị. Cuối cùng nó sẽ trả về thông tin về trạng thái hay lỗi cho nơi gọi và nếu có một yêu cầu khác ở hàng đợi, nó sẽ thực hiện tiếp, nếu không nó sẽ khóa lại chờ đến yêu cầu tiếp theo.

4.2.3 Phần mềm nhập/xuất độc lập thiết bị

Mặc dù một số phần mềm nhập/xuất mô tả thiết bị nhưng phần lớn chúng là độc lập với thiết bị. Ranh giới chính xác giữa drivers và phần mềm độc lập thiết bị là độc lập về mặt hệ thống, bởi vì một số hàm mà được thi hành theo kiểu độc lập thiết bị có thể được thi hành trên drivers vì lý do hiệu quả hay những lý dó khác nào đó.

Giao tiếp đồng nhất cho device drivers
Đặt tên thiết bị
Bảo vệ thiết bị
Cung cấp khối độc lập thiết bị
Tổ chức buffer
Định vị lưu trữ trên thiết bị khối
Cấp phát và giải phóng thiết bị tận hiến
Báo lỗi

Chức năng cơ bản của phần mềm nhập/xuất độc lập thiết bị là những chức năng chung cho tất cả các thiết bị và cung cấp một giao tiếp đồng nhất cho phần mềm phạm vi người sử dụng.

Trước tiên nó phải có chức năng tạo một ánh xạ giữa thiết bị và một tên hình thức. Ví dụ đối với UNIX, tên /dev/tty0 dành riêng để mô tả I-node cho một file đặc biệt, và I-node này chứa chứa số thiết bị chính, được dùng để xác định driver thích hợp và số thiết bị phụ, được dùng để xác định các tham số cho driver để cho biết là đọc hay ghi.

Thứ hai là bảo vệ thiết bị, là cho phép hay không cho phép người sử dụng truy xuất thiết bị. Các hệ điều hành có thể có hay không có chức năng này.

Thứ ba là cung cấp khối dữ liệu độc lập thiết bị vì ví dụ những đĩa khác nhau sẽ có kích thước sector khác nhau và điều này sẽ gây khó khăn cho các phần mềm người sử dụng ở lớp trên. Chức năng này cung cấp các khối dữ liệu logic độc lập với kích thước sector vật lý.

Thứ tư là cung cấp buffer để hỗ trợ cho đồng bộ hóa quá trình hoạt động của hệ thống. Ví dụ buffer cho bàn phím.

Thứ năm là định vị lưu trữ trên các thiết bị khối.

Thứ sáu là cấp phát và giải phóng các thiết bị tận hiến.

Cuối cùng là thông báo lỗi cho lớp bên trên từ các lỗi do device driver báo về.

4.2.4 Phần mềm nhập/xuất phạm vi người sử dụng

Hầu hết các phần mềm nhập/xuất đều ở bên trong của hệ điều hành và một phần nhỏ của chúng chứa các thư viện liên kết với chương trình của người sử dụng ngay cả những chương trình thi hành bên ngoài hạt nhân.

Lời gọi hệ thống, bao gồm lời gọi hệ thống nhập/xuất thường được thực hiện bởi các hàm thư viện. Ví dụ khi trong chương trình C có lệnh

count = write(fd, buffer, nbytes) ;

Hàm thư viện write được địch và liên kết dưới dạng nhị phân và nằm trong bộ nhớ khi thi hành. Tập hợp tất cả những hàm thư viện này rõ ràng là một phần của hệ thống nhập/xuất.

Không phải tất cả các phần mềm nhập/xuất đều chứa hàm thư viện, có một loại quan trọng khác gọi là hệ thống spooling dùng để khai thác tối đa thiết bị nhập/xuất trong hệ thống đa chương.

Các hàm thư viện chuyển các tham số thích hợp cho lời gọi hệ thống và hàm thư viện thực hiện việc định dạng cho nhập và xuất như lệnh printf trong C. Thư viện nhập/xuất chuẩn chứa một số hàm có chức năng nhập/xuất và tất cả chạy như chương trình người dùng.

Chức năng của spooling là tránh trường hợp một tiến trình đang truy xuất thiết bị, chiếm giữ thiết bị nhưng sau đó không làm gì cả trong một khoảng thời gian và như vậy các tiến trình khác bị ảnh hưởng vì không thể

truy xuất thiết bị đó. Một ví dụ của spooling device là line printer. Spooling còn được sử dụng trong hệ thống mạng như hệ thống e-mail chẳng hạn.

5. Tổ chức lưu trữ dữ liệu trên đĩa từ

Mục tiêu:

- *Lưu trữ được dữ liệu trên đĩa*

Về nguyên tắc hệ thống file trên CD_ROM đơn giản hơn so với những hệ thống file khác, vì các đĩa CD_ROM chỉ được ghi một lần (write-once media), do đó các file ghi trên nó không thể xóa bỏ hay thay đổi sau khi đĩa đã được chế tạo, chính vì vậy thành phần quản lý File/đĩa của hệ điều hành sẽ không lo đến việc quản lý các Block còn tự do trên đĩa cũng như việc cấp phát và thu hồi các Block cho các file, trong trường hợp các file được lưu trữ trên đĩa CD_ROM.

Sau đây chúng ta xem xét hệ thống file chính trên CD_ROM và 2 hệ thống mở rộng của chúng:

Hệ thống file ISO 9660: Đây là chuẩn phổ biến nhất đối với các hệ thống file CD_ROM và đã được chấp nhận như một chuẩn quốc tế vào năm 1988 với cái tên ISO 9660. Một trong những mục đích của chuẩn này là làm cho tất cả các CD_ROM đều có thể đọc được trên các máy tính khác nhau, nó không phụ thuộc vào thứ tự byte cũng như hệ điều hành đang sử dụng, kể cả hệ điều hành yếu nhất như MS_DOS.

Trên CD_ROM không có track, cylinder như trên các đĩa từ, nó chỉ có một đường xoắn ốc đi từ tâm đĩa ra bên ngoài, đường xoắn ốc này được chia thành các khối (block) logic có kích thước bằng nhau và bằng 2352 byte, đôi khi cũng được gọi là các sector logic. Một vài byte trong khối dành cho phần mở đầu, sửa chữa lỗi, và những việc khác. Phần chính của mỗi khối logic còn lại khoảng 2048 byte.

ISO 9660 hỗ trợ cho một tập đĩa CD_ROM với một tập gồm 2^{16}-1 đĩa, một CD_ROM riêng lẽ có thể được chia thành nhiều partition. Trong phần này chúng ta chỉ tìm hiểu chuẩn ISO 9660 với một CD_ROM và không được chia thành các Partition.

Mỗi CD_ROM đều có phần đầu của đĩa, dài 16 block, chức năng của phần này không được định nghĩa trong chuẩn ISO 9600. Các nhà sản xuất CD_ROM có thể sử dụng phần đầu này để ghi vào đó chương trình BootStrap cho phép máy tính có thể khởi động được từ đĩa CD_ROM, hoặc dùng cho những mục đích khác.

Phần tiếp theo là 1 block chứa bộ mô tả Volume chính, bộ mô tả này chứa một số thông tin chung về CD_ROM, bao gồm: định danh hệ thống (32byte), định danh volume (32byte), định danh nhà sản xuất (128byte) và định danh dữ liệu (128byte). Khi chế tạo có thể lấp đầy những trường trên theo ý muốn. Trong phần này còn chứa phần giới thiệu, bản quyền tác giả,

thông tin thư mục, kích thước của một khối logic (2048, 4096, 8192, ...), số các block trên CD_ROM, và thời gian tạo và kết thúc của CD_ROM. Cuối cùng, trong bộ mô tả Volume chính còn chứa một tập các mục vào (directory entry) cho thư mục gốc, tại đây chứa địa chỉ của block bắt đầu của thư mục gốc trên CD_ROM. Trên CD_ROM có 2 bộ mô tả volume chính, có nội dung hoàn toàn giống nhau, sử dụng một bộ và một bộ để dự phòng.

Sau các phần trên là phần bắt đầu của CD_ROM dùng để chứa các file đang được ghi trên đĩa.

Thư mục gốc và tất cả các thư mục khác, chỉ gồm một số mục vào, phần cuối của chúng chứa một bít đánh dấu (mark). Mỗi mục vào chứa từ 10 đến 12 trường, trong đó có một số thuộc ASCII và số khác là những trường số thuộc số nhị phân.

Mở rộng Rock Ridge: Các chuyên viên thiết kế của UNIX nhận thấy ISO 9660 còn một vài hạn chế, do đó họ đã mở rộng ISO 9660 với mục đích là cho nó có thể thay thế cho hệ thống file của UNIX trên các đĩa CD_ROM và các file được tạo từ UNIX có thể được sao chép sang CD_ROM và ngược lại, chuẩn mở rộng này được gọi là Rock Ridge.

Rock Ridge giữ lại tất cả các trường của ISO 9660, và sử dụng trường System để đưa thêm vào các trường mới, các hệ thống file khác không nhận biết các trường này và xem CD_ROM như một đĩa CD_ROM thông thường. Rock Ridge bổ sung thêm các trường, theo thứ tự là: PX: Posix Attributes, PN: Major and miror device number, SL: Symbolic link, NM: Alternative name, CL: Child location, PL: Parent location, RE: Relocaltion, TF: Times stamps, trong đó trường quan trọng nhất là NM, trường này cho phép sử dụng 2 tên file cho một file, một tên file trong mục vào của thư mục và một tên file kết hợp, tên này không phụ vào tập kí tự hoặc giới hạn chiều dài của chuẩn ISO 9660.

Mở rộng Joliet: Cũng như các chuyên viên thiết kế của UNIX, các chuyên viên thiết kế của Microsoft muốn mở rộng ISO 9660 sao cho các file được tạo từ Windows có thể được sao chép sang CD_ROM và ngược lại và họ đã thành công với mở rộng Joliet. Mở rộng Joliet cho phép: Tên file dài đến 64 kí tự; Sử dụng tập kí tự Unicode nên tên file có thể dài đến 128 kí tự; Có nhiều hơn 8 cấp thư mục lồng nhau; Sử dụng tên thư mục với phần mở rộng.

CÂU HỎI CỦNG CỐ BÀI HỌC

1.Trình bày các phương pháp tổ chức và truy nhập dữ liệu.

2.Mô tả dạng sơ lược nhất sơ đồ khối của thuật toán tìm kiếm một bản ghi có chỉ số k trong File được tổ chức kiểu chỉ số kế tiếp có sử dụng các vùng chống tràn.

3.Trên một đĩa từ có dung lượng cần quản lý là 100MB, với mỗi khối dữ liệu là 1 KB. Nếu sử dụng phương pháp bit map để quản lý dung lượng đĩa đã cho thì đòi hỏi vùng bit map cần có dung lượng là bao nhiêu byte.

4.Trình bày quy trình chung của điều khiển nhập xuất.

CHƯƠNG 2 :
ĐIỀU KHIỂN BỘ NHỚ

Mã chương: MH15-03

Giới thiệu:

Quản lý bộ nhớ là một trong những nhiệm vụ quan trọng và phức tạp nhất của hệ điều hành. Bộ phận quản lý bộ nhớ xem bộ nhớ chính như là một tài nguyên của hệ thống dùng để cấp phát và chia sẻ cho nhiều tiến trình đang ở trong trạng thái active. Các hệ điều hành đều mong muốn có nhiều hơn các tiến trình trên bộ nhớ chính. Công cụ cơ bản của quản lý bộ nhớ là sự phân trang (paging) và sự phân đoạn (segmentation). Với sự phân trang mỗi tiến trình được chia thành nhiều phần nhỏ có quan hệ với nhau, với kích thước của trang là cố định. Sự phân đoạn cung cấp cho chươngtrình người sử dụng các khối nhớ có kích thước khác nhau. Hệ điều hành cũng có thể kết hợp giữa phân trang và phân đoạn để có được một chiến lược quản lý bộ nhớ linh hoạt hơn.

Mục Tiêu:

- Nắm được nguyên lý điều khiển bộ nhớ của hệ điều hành, phương thức tối ưu hóa việc phân phối bội nhớ, tránh lãng phí và chia sẻ tài nguyên bộ nhớ.
- Rèn luyện khả năng tư duy, lập luận có tính khoa học
- Tinh thần hỗ trợ nhau trong học tập.

Nội Dung Chính:

1.Quản lý và bảo vệ bộ nhớ

Mục tiêu:

- Nắm được nguyên lý điều khiển bộ nhớ của hệ điều hành.

Nhiệm vụ của quản lý bộ nhớ

Trong các hệ thống đơn chương trình (uniprogramming), trên bộ nhớ chính ngoài hệ điều hành, chỉ có một chương trình đang thực hiện. Trong các hệ thống đa chương (multiprogramming) trên bộ nhớ chính ngoài hệ điều hành, có thể có nhiều tiến trình đang hoạt động. Do đó nhiệm vụ quản lý bộ nhớ của hệ điều hành trong hệ thống đa chương trình sẽ phức tạp hơn nhiều so với trong hệ thống đơn chương trình. Trong hệ thống đa chương bộ phận quản lý bộ nhớ phải có nhiệm vụ đưa bất kỳ một tiến trình nào đó vào bộ nhớ khi nó có yêu cầu, kể cả khi trên bộ nhớ không còn không gian trống, ngoài ra nó phải bảo vệ chính hệ điều hành và các tiến trình trên bộ nhớ tránh các trường hợp truy xuất bất hợp lệ xảy ra. Như vậy việc quản lý bộ nhớ trong các hệ thống đa chương là quan trọng và cần thiết. Bộ phận quản lý bộ nhớ phải thực hiện các nhiệm vụ sau đây:

- **Sự tái định vị** (Relocation): Trong các hệ thống đa chương,

không gian bộ nhớ chính thường được chia sẻ cho nhiều tiến trình khác nhau và yêu cầu bộ nhớ của các tiến trình luôn lớn hơn không gian bộ nhớ vật lý mà hệ thống có được. Do dó, một chương trình đang hoạt động trên bộ nhớ cũng có thể bị đưa ra đĩa (swap-out) và nó sẽ được đưa vào lại (swap-in) bộ nhớ tại một thời điểm thích hợp nào đó sau này. Vấn đề đặt ra là khi đưa một chương trình vào lại bộ nhớ thì hệ điều hành phải định vị nó vào đúng vị trí mà nó đã được nạp trước đó. Để thực hiện được điều này hệ điều hành phải có các cơ chế để ghi lại tất cả các thông tin liên quan đến một chương trình bị swap-out, các thông tin này là cơ sở để hệ điều hành swap-in chương trình vào lại bộ nhớ chính và cho nó tiếp tục hoạt động. Hệ điều hành buộc phải swap-out một chương trình vì nó còn không gian bộ nhớ chính để nạp tiến trình khác, do dó sau khi swap-out một chương trình hệ điều hành phải tổ chức lại bộ nhớ để chuẩn bị nạp tiến trình vừa có yêu cầu. Các nhiệm vụ trên do bộ phần quản lý bộ nhớ của hệ điều hành thực hiện. Ngoài ra trong nhiệm vụ này hệ điều hành phải có khả năng chuyển đổi các địa chỉ bộ nhớ được ghi trong code của chương trình thành các địa chỉ vật lý thực tế trên bộ nhớ chính khi chương trình thực hiện các thao tác truy xuất trên bộ nhớ, bởi vì người lập trình không hề biết trước hiện trạng của bộ nhớ chính và vị trí mà chương trình được nạp khi chương trình của họ hoạt động. Trong một số trường hợp khác các chương trình bị swap-out có thể được swap-in vào lại bộ nhớ tại vị trí khác với vị trí mà nó được nạp trước đó.

- **Bảo vệ bộ nhớ** (Protection): Mỗi tiến trình phải được bảo vệ để chống lại sự truy xuất bất hợp lệ vô tình hay có chủ ý của các tiến trình khác. Vì thế các tiến trình trong các chương trình khác không thể tham chiếu đến các vùng nhớ đã dành cho một tiến trình khác để thực hiện các thao tác đọc/ghi mà không được phép (permission), mà nó chỉ có thể truy xuất đến không gian địa chỉ bộ nhớ mà hệ điều hành đã cấp cho tiến trình đó. Để thực hiện điều này hệ thống quản lý bộ nhớ phải biết được không gian địa chỉ của các tiến trình khác trên bộ nhớ và phải kiểm tra tất cả các yêu cầu truy xuất bộ nhớ của mỗi tiến trình khi tiến trình đưa ra địa chỉ truy xuất. Điều này khó thực hiện vì không thể xác định địa chỉ của các chương trình trong bộ nhớ chính trong quá trình biên dịch mà phải thực hiện việc tính toán địa chỉ tại thời điểm chạy chương trình. Hệ điều hành có nhiều chiến lược khác nhau để thực hiện điều này.

  Điều quan trọng nhất mà hệ thống quản lý bộ nhớ phải thực hiện là không cho phép các tiến trình của người sử dụng truy cập đến bất kỳ một vị trí nào của chính hệ điều hành, ngoại trừ vùng dữ liệu và các rountine mà hệ điều hành cung cấp cho chương trình người sử dụng.

- **Chia sẻ bộ nhớ** (Sharing): Bất kỳ một chiến lược nào được

cài đặt đều phải có tính mềm dẻo để cho phép nhiều tiến trình có thể truy cập đến cùng một địa chỉ trên bộ nhớ chính. Ví dụ, khi có nhiều tiến trình cùng thực hiện một chương trình thì việc cho phép mỗi tiến trình cùng truy cập đến một bản copy của chương trình sẽ thuận lợi hơn khi cho phép mỗi tiến trình truy cập đến một bản copy sở hữu riêng. Các tiến trình đồng thực hiện (co-operating) trên một vài tác vụ có thể cần để chia sẻ truy cập đến cùng một cấu trúc dữ liệu. Hệ thống quản lý bộ nhớ phải điều khiển việc truy cập đến không gian bộ nhớ được chia sẻ mà không vi phạm đến các yêu cầu bảo vệ bộ nhớ. Ngoài ra, trong môi trường hệ điều hành đa nhiệm hệ điều hành phải chia sẻ không gian nhớ cho các tiến trình để hệ điều hành có thể nạp được nhiều tiến trình vào bộ nhớ để các tiến trình này có thể hoạt động đồng thời với nhau.

- **Tổ chức bộ nhớ logic** (Logical organization): Bộ nhớ chính của hệ thống máy tính được tổ chức như là một dòng hoặc một mảng, không gian địa chỉ bao gồm một dãy có thứ tự các byte hoặc các word. Bộ nhớ phụ cũng được tổ chức tương tự. Mặc dù việc tổ chức này có sự kết hợp chặt chẽ với phần cứng thực tế của máy nhưng nó không phù hợp với các chương trình. Đa số các chương trình đều được chia thành các modun, một vài trong số đó là không thể thay đổi (read only, execute only) và một vài trong số đó chứa dữ liệu là có thể thay đổi. Nếu hệ điều hành và phần cứng máy tính có thể giao dịch một cách hiệu quả với các chương trình của người sử dụng và dữ liệu trong các modun thì một số thuận lợi có thể thấy rõ sau đây:

  Các modun có thể được viết và biên dịch độc lập, với tất cả các tham chiếu từ một modun đến modun khác được giải quyết bởi hệ thống tại thời điểm chạy.

  Các mức độ khác nhau của sự bảo vệ, read-only, execute-only, có thể cho ra các modun khác nhau.

  Nó có thể đưa ra các cơ chế để các modun có thể được chia sẻ giữa các tiến trình.

  Công cụ đáp ứng cho yêu cầu này là sự phân đoạn (segmentation), đây là một trong những kỹ thuật quản lý bộ nhớ được trình bày trong chương này.

- **Tổ chức bộ nhớ vật lý** (Physical organization): Như chúng ta đã biết bộ nhớ máy tính được tổ chức theo 2 cấp: bộ nhớ chính và bộ nhớ phụ. Bộ nhớ chính cung cấp một tốc độ truy cập dữ liệu cao, nhưng dữ liệu trên nó phải được làm tươi thường xuyên và không thể tồn tại lâu dài trên nó. Bộ nhớ phụ có tốc độ truy xuất chậm và rẻ tiền hơn so với bộ nhớ chính nhưng nó không cần làm tươi thường xuyên. Vì thế bộ nhớ phụ có khả năng lưu trữ lớn và cho phép lưu trữ dữ liệu và chương trình trong một

khoảng thời gian dài, trong khi đó bộ nhớ chính chỉ để giữ (hold) một khối lượng nhỏ các chương trình và dữ liệu đang được sử dụng tại thời điểm hiện tại.

Trong giản đồ 2 cấp này, việc tổ chức luồng thông tin giữa bộ nhớ chính và bộ nhớ phụ là một nhiệm vụ quan trọng của hệ thống. Sự chịu trách nhiệm cho luồng này có thể được gán cho từng người lập trình riêng, nhưng điều này là không hợp lý và có thể gây rắc rối, là do hai nguyên nhân:

Không gian bộ nhớ chính dành cho các chương trình cùng với dữ liệu của nó thường là không đủ, trong trường hợp này, người lập trình phải tiến hành một thao tác được hiểu như là Overlaying, theo đó chương trình và dữ liệu được tổ chức thành các modun khác nhau có thể được gán trong cùng một vùng của bộ nhớ, trong đó có một chương trình chính chịu trách nhiệm chuyển các modun vào và ra khi cần.

Trong môi trường đa chương trình, người lập trình không thể biết tại một thời điểm xác định có bao nhiêu không gian nhớ còn trống hoặc khi nào thì không gian nhớ sẽ trống. Như vậy nhiệm vụ di chuyển thông tin giữa 2 cấp bộ nhớ phải do hệ thống thực hiện. Đây là nhiệm vụ cơ bản mà thành phần quản lý bộ nhớ phải thực hiện.

2.Điều khiển bộ nhớ liên tục

Mục tiêu:

- *Nắm được phương thức tối ưu hóa việc phân phối bộ nhớ, tránh lãng phí và chia sẻ tài nguyên bộ nhớ.*

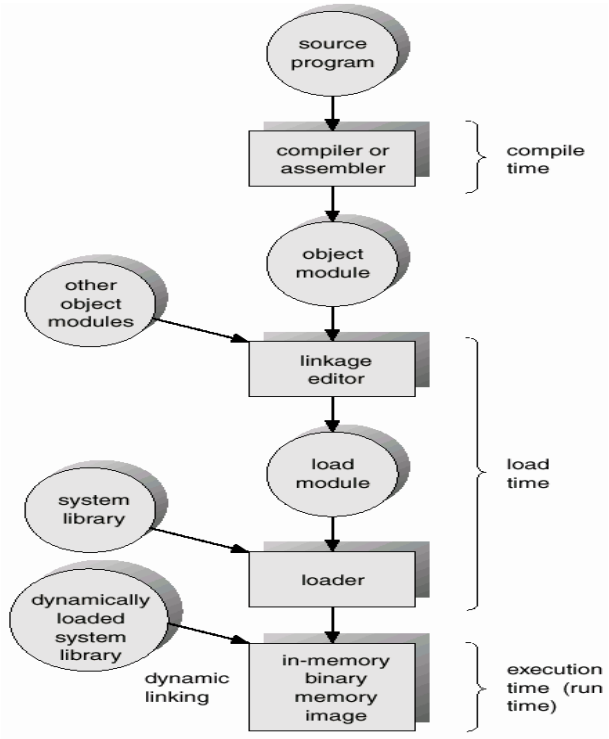
2.1.Giới thiệu

Thông thường, một chương trình nằm trên đĩa như một tập tin có thể thực thi dạng nhị phân. Chương trình này được mang vào trong bộ nhớ và được đặt trong một quá trình để nó được thực thi. Phụ thuộc vào việc quản lý bộ nhớ đang dùng, quá trình có thể được di chuyển giữa đĩa và bộ nhớ trong khi thực thi. Tập hợp các quá trình trên đĩa đang chờ được mang vào bộ nhớ để thực thi hình thành một **hàng đợi nhập** (input queue).

Thủ tục thông thường là chọn một trong những quá trình trong hàng đợi nhập và nạp quá trình đó vào trong bộ nhớ. Khi một quá trình được thực thi, nó truy xuất các chỉ thị và dữ liệu từ bộ nhớ. Cuối cùng, một quá trình kết thúc và không gian bộ nhớ của nó được xác định là trống.

Hầu hết các hệ thống cho phép một quá trình người dùng nằm ở bất cứ phần nào của bộ nhớ vật lý. Do đó, mặc dù không gian địa chỉ của máy tính bắt đầu tại 00000, nhưng địa chỉ đầu tiên của quá trình người dùng không cần tại 00000. Sắp xếp này ảnh hưởng đến địa chỉ mà chương trình người dùng có thể dùng. Trong hầu hết các trường hợp, một chương trình người dùng sẽ đi qua một số bước- một vài trong chúng có thể là tuỳ chọn-trước khi được thực thi (hình VII-1). Các địa chỉ có thể được hiện

diện trong những cách khác trong những bước này. Các địa chỉ trong chương trình nguồn thường là những danh biểu. Một trình biên dịch sẽ liên kết các địa chỉ danh biểu tới các địa chỉ có thể tái định vị (chẳng hạn như 14 bytes từ vị trí bắt đầu của module này). Bộ soạn thảo liên kết hay bộ nạp sẽ liên kết các địa chỉ có thể tái định vị tới địa chỉ tuyệt đối (chẳng hạn 74014). Mỗi liên kết là một ánh xạ từ một không gian địa chỉ này tới một không gian địa chỉ khác.



Hình 3.1: Xử lý nhiều bước của chương trình người dùng

Về truyền thống, liên kết các chỉ thị và dữ liệu tới các địa chỉ có thể được thực hiện tại bất cứ bước nào theo cách sau đây:

• **Thời gian biên dịch**: nếu tại thời điểm biên dịch có thể biết quá trình nằm ở đâu trong bộ nhớ thì mã tuyệt đối có thể được phát sinh. Thí dụ, nếu biết trước quá trình người dùng nằm tại vị trí R thì mã trình biên dịch được
phát sinh sẽ bắt đầu tại vị trí đó và mở rộng từ đó. Nếu tại thời điểm sau đó, vị trí bắt đầu thay đổi thì sẽ cần biên dịch lại mã này. Các chương trình định dạng .COM của MS-DOS là mã tuyệt đối giới hạn tại thời điểm biên dịch.

• **Thời điểm nạp**: nếu tại thời điểm biên dịch chưa biết nơi quá trình sẽ nằm ở đâu trong bộ nhớ thì trình biên dịch phải phát sinh mã có thể tái định vị. Trong trường hợp này, liên kết cuối cùng được trì hoãn cho tới thời điểm.

2.2.Cấp phát tĩnh

Trong kỹ thuật này không gian địa chỉ của bộ nhớ chính được chia thành 2 phần cố định, phần nằm ở vùng địa chỉ thấp dùng để chứa chính hệ điều hành, phần còn lại, tạm gọi là phần user program, là sẵn sàng cho việc sử dụng của các tiến trình khi các tiến trình được nạp vào bộ nhớ chính.

Trong các hệ thống đơn chương, phần user program được dùng để cấp cho chỉ một chương trình duy nhất, do đó nhiệm vụ quản lý bộ nhớ của hệ điều hành trong trường hợp này sẽ đơn giản hơn, hệ điều hành chỉ kiểm soát sự truy xuất bộ nhớ của chương trình người sử dụng, không cho nó truy xuất lên vùng nhớ của hệ điều hành. Để thực hiện việc này hệ điều hành sử dụng một thanh ghi giới hạn để ghi địa chỉ ranh giới giữa hệ điều hành và chương trình của người sử dụng, theo đó khi chương trình người sử dụng cần truy xuất một địa chỉ nào đó thì hệ điều hành sẽ so sánh địa chỉ này với giá trị địa chỉ được ghi trong thành ghi giới hạn, nếu nhỏ hơn thì từ chối không cho truy xuất, ngược lại thì cho phép truy xuất. Việc so sánh địa chỉ này cần phải có sự hỗ trợ của phần cứng và có thể làm giảm tốc độ truy xuất bộ nhớ của hệ thống nhưng bảo vệ được hệ điều hành tránh việc chương trình của người sử dụng làm hỏng hệ điều hành dẫn đến làm hỏng hệ thống.

Trong các hệ thống đa chương, phần user program lại được phân ra thành nhiều phân vùng (partition) với các biên vùng cố định có kích thước bằng nhau hay không bằng nhau. Trong trường hợp này một tiến trình có thể được nạp vào bất kỳ partition nào nếu kích thước của nó nhỏ hơn hoặc bằng kích thước của partition và partition này còn trống. Khi có một tiến trình cần được nạp vào bộ nhớ nhưng tất cả các partition đều đã chứa các tiến trình khác thì hệ điều hành có thể chuyển một tiến trình nào đó, mà hệ điều hành cho là hợp lệ (kích thước vừa đủ, không đang ở trạng thái ready hoặc running, không có quan hệ với các tiến trình running khác, ...), ra ngoài (swap out), để lấy partition trống đó nạp tiến trình vừa có yêu cầu. Đây là nhiệm vụ phức tạp của hệ điều hành, hệ điều hành phải chi phí cao cho công việc này.

Có hai trở ngại trong việc sử dụng các phân vùng cố định với kích thước bằng nhau:

Thứ nhất, khi kích thước của một chương trình là quá lớn so với kích thước của một partition thì người lập trình phải thiết kế chương trình theo cấu trúc overlay, theo đó chỉ những phần chia cần thiết của chương trình mới được nạp vào bộ nhớ chính khi khởi tạo chương trình, sau đó người lập trình phải nạp tiếp các modun cần thiết khác vào đúng partition của chương trình và sẽ ghi đè lên bất kỳ chương trình hoặc dữ liệu ở trong đó. Cấu trúc chương trình overlay tiết kiệm được bộ nhớ nhưng yêu cầu

cao ở người lập trình.

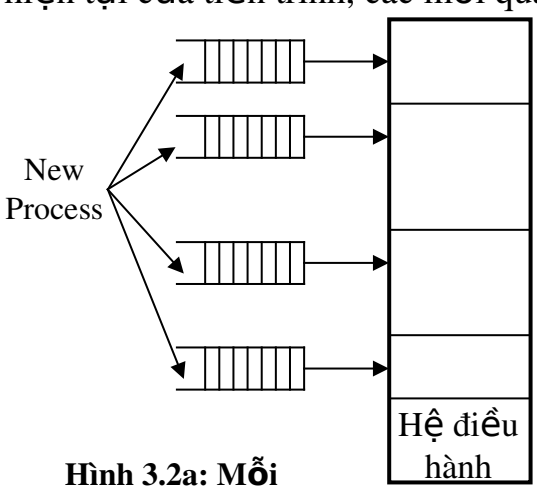
Thứ hai, khi kích thước của một chương trình nhỏ hơn kích thước của một partition hoặc quá lớn so với kích thước của một partition nhưng không phải là bội số của kích thước một partition thì dễ xảy ra hiện tượng phân mảnh bên trong (internal fragmentation) bộ nhớ, gây lãng phí bộ nhớ. Ví dụ, nếu có 3 không gian trống kích thước 30K nằm rãi rác trên bộ nhớ, thì cũng sẽ không nạp được một modun chương trình có kích thước 12K, hiện tượng này được gọi là hiện tượng phân mảnh bên trong.

Cả hai vấn đề trên có thể được khắc phục bằng cách sử dụng các phân vùng có kích thước không bằng nhau.

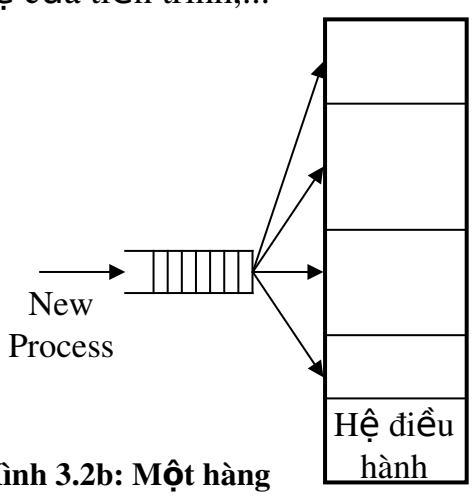
Việc đưa một tiến trình vào partition trong hệ thống đa chương với phân vùng cố định kích thước không bằng nhau sẽ phức tạp hơn nhiều so với trường hợp các phân vùng có kích thước bằng nhau. Với các partition có kích thước không bằng nhau thì có hai cách để lựa chọn khi đưa một tiến trình vào partition:

Mỗi phân vùng có một hàng đợi tương ứng, theo đó mỗi tiến trình khi cần được nạp vào bộ nhớ nó sẽ được đưa đến hành đợi của phân vùng có kích thước vừa đủ để chứa nó, để vào/để đợi được vào phân vùng. Cách tiếp cận này sẽ đơn giản trong việc đưa một tiến trình từ hàng đợi vào phân vùng vì không có sự lựa chọn nào khác ở đây, khi phân vùng mà tiến trình đợi trống nó sẽ được đưa vào phân vùng đó. Tuy nhiên các tiếp cận này kém linh động vì có thể có một phân vùng đang trống, trong khi đó có nhiều tiến trình đang phải phải đợi để được nạp vào các phân vùng khác, điều này gây lãng phí trong việc sử dụng bộ nhớ.

Hệ thống dùng một hàng đợi chung cho tất cả các phân vùng, theo đó tất cả các tiến trình muốn được nạp vào phân vùng nhưng chưa được vào sẽ được đưa vào hàng đợi chung này. Sau đó nếu có một phân vùng trống thì hệ thống sẽ xem xét để đưa một tiến trình có kích thước vừa đủ vào phân vùng trống đó. Cách tiếp cận này linh động hơn so với việc sử dụng nhiều hàng đợi như ở trên, nhưng việc chọn một tiến trình trong hàng đợi để đưa vào phân vùng là một việc làm khá phức tạp của hệ điều hành vì nó phải dựa vào nhiều yếu tố khác nhau như: độ ưu tiên của tiến trình, trạng thái hiện tại của tiến trình, các mối quan hệ của tiến trình,...



Hình 3.2a: Mỗi partition có một hàng đợi riêng



Hình 3.2b: Một hàng đợi chung cho tất cả partition

Mặc dầu sự phân vùng cố định với kích thước không bằng nhau cung cấp một sự mềm dẻo hơn so với phân vùng cố định với kích thước bằng nhau, nhưng cả hai loại này còn một số hạn chế sau đây:

Số lượng các tiến trình có thể hoạt động trong hệ thống tại một thời điểm phụ thuộc vào số lượng các phân vùng cố định trên bộ nhớ.

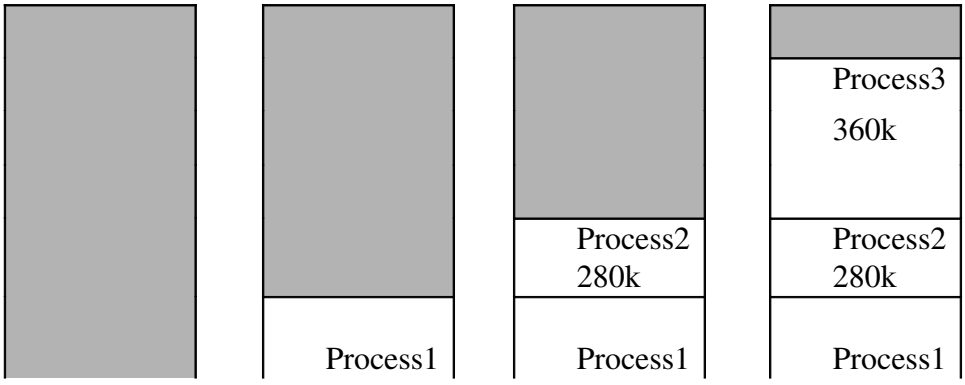
Tương tự như trên, nêu kích thước của tiến trình nhỏ hơn kích thước của một phân vùng thì có thể dẫn đến hiện tượng phân mảnh nội vi gây lãng phí trong việc sử dụng bộ nhớ.

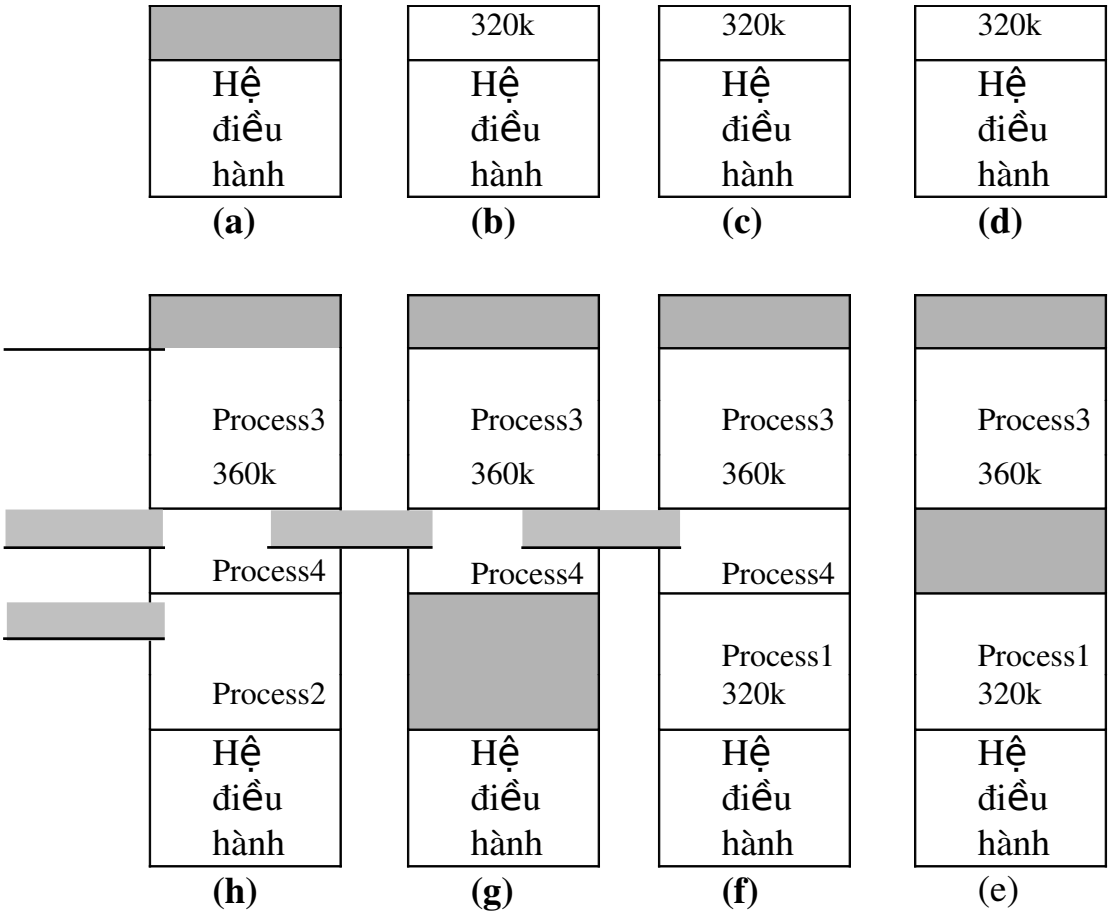
Sự phân vùng cố định ít được sử dụng trong các hệ điều hành hiện nay.

2.3.Cấp phát động

Để khắc phục một vài hạn chế của kỹ thuật phân vùng cố định, kỹ thuật phân vùng động ra đời. Kỹ thuật này thường được sử dụng trong các hệ điều hành gần đây như hệ điều hành mainframe của IBM, hệ điều hành OS/MVT,...

Trong kỹ thuật phân vùng động, số lượng các phân vùng trên bộ nhớ và kích thước của mỗi phân vùng là có thể thay đổi. Tức là phần user program trên bộ nhớ không được phân chia trước mà nó chỉ được ấn định sau khi đã có một tiến trình được nạp vào bộ nhớ chính. Khi có một tiến trình được nạp vào bộ nhớ nó được hệ điều hành cấp cho nó không gian vừa đủ để chứa tiến trình, phần còn lại để sẵn sàng cấp cho tiến trình khác sau này. Khi một tiến trình kết thúc nó được đưa ra ngoài và phần không gian bộ nhớ mà tiến trình này trả lại cho hệ điều hành sẽ được hệ điều hành cấp cho tiến trình khác, cả khi tiến trình này có kích thước nhỏ hơn kích thước của không gian nhớ trống đó.



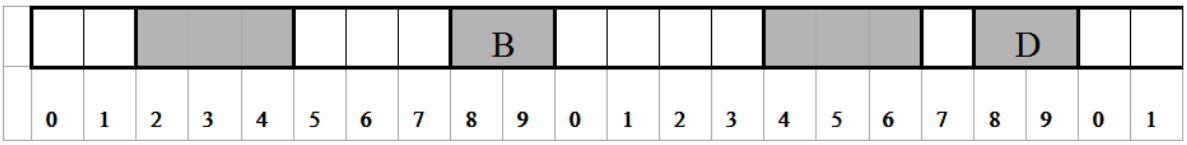


Hình 3.3: Kết quả của sự phân trang động với thứ tự nạp các tiến trình.

Hình vẽ 3.3 trên đây minh họa cho quá trình nạp/kết thúc các tiến trình theo thứ tự: nạp process1, nạp process2, nạp process3, kết thúc process2, nạp process4, kết thúc process1, nạp process2 vào lại, trong hệ thống phân vùng động. Như vậy dần dần trong bộ nhớ hình thành nhiều không gian nhớ có kích thước nhỏ không đủ chứa các tiến trình nằm rải rác trên bộ nhớ chính, hiện tượng này được gọi là hiện thượng phân mảnh bên ngoài (external fragmentation). Để chống lại sự lãng phí bộ nhớ do phân mảnh, thỉnh thoảng hệ điều hành phải thực hiện việc sắp xếp lại bộ nhớ, để các không gian nhớ nhỏ rời rác nằm liền kề lại với nhau tạo thành một khối nhớ có kích thước đủ lớn để chứa được một tiến trình nào đó. Việc làm này làm chậm tốc độ của hệ thống, hệ điều hành phải chi phí cao cho việc này, đặc biệt là việc tái định vị các tiến trình khi một tiến trình bị đưa ra khỏi bộ nhớ và được nạp vào lại bộ nhớ để tiếp tục hoạt động.

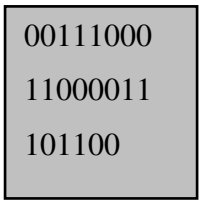
Trong kỹ thuật phân vùng động này hệ điều hành phải đưa ra các cơ chế thích hợp để quản lý các khối nhớ đã cấp phát hay còn trống trên bộ nhớ. Hệ điều hành sử dụng 2 cơ chế: Bản đồ bít và Danh sách liên kết. Trong cả 2 cơ chế này hệ điều hành đều chia không gian nhớ thành các đơn

vị cấp phát có kích thước bằng nhau, các đơn vị cấp phát liên tiếp nhau tạo thành một khối nhớ (block), hệ điều hành cấp phát các block này cho các tiến trình khi nạp tiến trình vào bộ nhớ.

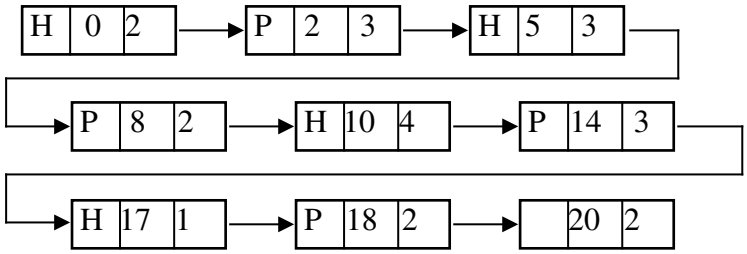


Hình 3.4a: Một đoạn nhớ bao gồm 22 đơn vị cấp phát, tạo thành 9 block, trong đó có 4 block đã cấp phát (tô đậm, kí hiệu là P) cho các tiến trình: A, B, C, D và 5 block chưa được cấp phát (để trắng, kí hiệu là H).

Trong cơ chế bản đồ bít: mỗi đơn vị cấp phát được đại diện bởi một bít trong bản đồ bít. Đơn vị cấp phát còn trống được đại diện bằng bít 0, ngược lại đơn vị cấp phát được đại diện bằng bít 1. Hình 3.4b là bản đồ bít của khối nhớ ở trên.



Hình 3.4b: quản lý các đơn vị cấp phát bằng bản đồ bít.



Hình 3.4c: quản lý các đơn vị cấp phát bằng danh sách liên kết.

Trong cơ chế danh sách liên kết: Mỗi block trên bộ nhớ được đại diện bởi một phần tử trong danh sách liên kết, mỗi phần tử này gồm có 3 trường chính: trường thứ nhất cho biết khối nhớ đã cấp phát (P: process) hay đang còn trống (H: Hole), trường thứ hai cho biết thứ tự của đơn vị cấp phát đầu tiên trong block, trường thứ ba cho biết block gồm bao nhiêu đơn vị cấp phát. Hình 3.4c là danh sách liên kết của khối nhớ ở trên.

Như vậy khi cần nạp một tiến trình vào bộ nhớ thì hệ điều hành phải dựa vào bản đồ bit hoặc danh sách liên kết để tìm ra một block có kích thước đủ để nạp tiến trình. Sau khi thực hiện một thao tác cấp phát hoặc sau khi đưa một tiến trình ra khỏi bộ nhớ thì hệ điều hành phải cập nhật lại bản đồ bít hoặc danh sách liên kết, điều này có thể làm giảm tốc độ thực hiện của hệ thống.

Chọn kích thước của một đơn vị cấp phát là một vấn đề quan trọng trong thiết kế, nếu kích thước đơn vị cấp phát nhỏ thì bản đồ bít sẽ lớn, hệ thống phải tốn bộ nhớ để chứa nó. Nếu kích thước của một đơn vị cấp phát lớn thì bản đồ bít sẽ nhỏ, nhưng sự lãng phí bộ nhớ ở đơn vị cấp

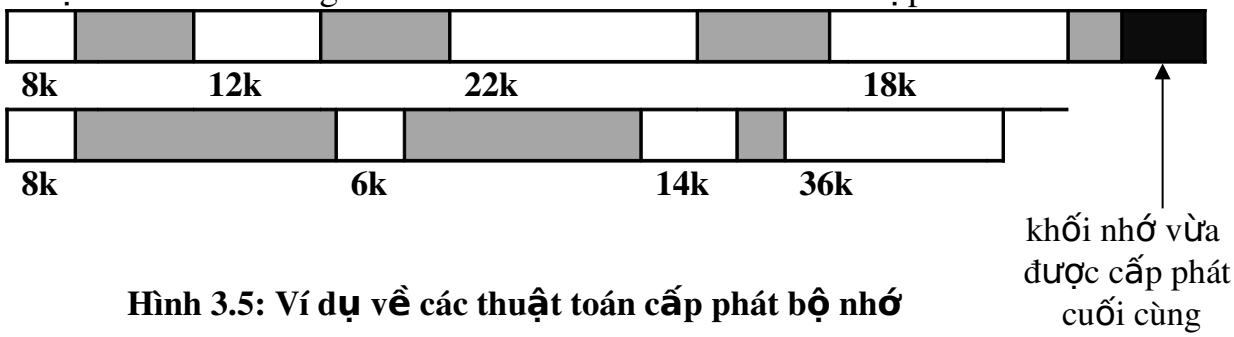
phát cuối cùng của một tiến trình sẽ lớn khi kích thước của tiến trình không phải là bội số của một đơn vị cấp phát. Điều vừa trình bày cũng đúng trong trường hợp danh sách liên kết.

Danh sách liên kết có thể được sắp xếp theo thứ tự tăng dần hoặc giảm dần của kích thước hoặc địa chỉ, điều này giúp cho việc tìm khối nhớ trống có kích thước vừa đủ để nạp các tiến trình theo các thuật toán dưới đây sẽ đạt tốc độ nhanh hơn và hiệu quả cao hơn. Một số hệ điều hành tổ chức 2 danh sách liên kết riêng để theo dõi các đơn vị cấp phát trên bộ nhớ, một danh sách để theo dõi các block đã cấp phát và một danh dách để theo dõi các block còn trống. Cách này giúp việc tìm các khối nhớ trống nhanh hơn, chỉ tìm trên danh sách các khối nhớ trống, nhưng tốn thời gian nhiều hơn cho việc cấp nhật danh sách sau mỗi thao tác cấp phát, vì phải thực hiện trên cả hai danh sách.

Khi có một tiến trình cần được nạp vào bộ nhớ mà trong bộ nhớ có nhiều hơn một khối nhớ trống (Free Block) có kích thước lớn hơn kích thước của tiến trình đó, thì hệ điều hành phải quyết định chọn một khối nhớ trống phù hợp nào để nạp tiến trình sao cho việc lựa chọn này dẫn đến việc sử dụng bộ nhớ chính là hiệu quả nhất. Có 3 thuật toán mà hệ điều hành sử dụng trong trường hợp này, đó là: Best-fit, First-fit, và Next-fit. Cả 3 thuật toán này đều phải chọn một khối nhớ trống có kích thước bằng hoặc lớn hơn kích thước của tiến trình cần nạp vào, nhưng nó có các điểm khác nhau cơ bản sau đây:

Best-fit: chọn khối nhớ có kích thước vừa đúng bằng kích thước của tiến trình cần được nạp vào bộ nhớ.

First-fit: trong trường hợp này hệ điều hành sẽ bắt đầu quét qua các khối nhớ trống bắt đầu từ khối nhớ trống đầu tiên trong bộ nhớ, và sẽ chọn khối nhớ trống đầu tiên có kích thước đủ lớn để nạp tiến trình.



Hình 3.5: Ví dụ về các thuật toán cấp phát bộ nhớ

Next-fit: tương tự như First-fit nhưng ở đây hệ điều hành bắt đầu quét từ khối nhớ trống kế sau khối nhớ vừa được cấp phát và chọn khối nhớ trống kế tiếp đủ lớn để nạp tiến trình.

Hình vẽ 3.5 cho thấy hiện tại trên bộ nhớ có các khối nhớ chưa được cấp phát theo thứ tự là: 8k, 12k, 22k, 18k, 8k, 6k, 14k, 36k. Trong trường

hợp này nếu có một tiến trình có kích thước 16k cần được nạp vào bộ nhớ, thì hệ điều hành sẽ nạp nó vào:

khối nhớ 22k nếu theo thuật toán First-fit
khối nhớ 18k nếu theo thuật toán Best-fit
khối nhớ 36k nếu theo thuật toán Next-fit

Như vậy nếu theo Best-fit thì sẽ xuất hiện một khối phân mảnh 2k, nếu theo First-fit thì sẽ xuất hiện một khối phân mảnh 6k, nếu theo Next-fit thì sẽ xuất hiện một khối phân mảnh 20k.

Các hệ điều hành không cài đặt cố định trước một thuật toán nào, tuỳ vào trường hợp cụ thể mà nó chọn cấp phát theo một thuật toán nào đó, sao cho chi phí về việc cấp phát là thấp nhất và hạn chế được sự phân mảnh bộ nhớ sau này. Việc chọn thuật toán này thường phụ thuộc vào thứ tự swap và kích thước của tiến trình. Thuật toán First-fit được đánh giá là đơn giản, dễ cài đặt nhưng mang lại hiệu quả cao nhất đặc biệt là về tốc độ cấp phát. Về hiệu quả thuật toán Next-fit không bằng First-fit, nhưng nó thường xuyên sử dụng được các khối nhớ trống ở cuối vùng nhớ, các khối nhớ ở vùng này thường có kích thước lớn nên có thể hạn chế được sự phân mảnh, theo ví dụ trên thì việc xuất hiện một khối nhớ trống 20k sau khi cấp một tiến trình 16k thì không thể gọi là phân mảnh được, nhưng nếu tiếp tục như thế thì dễ dẫn đến sự phân mảnh lớn ở cuối bộ nhớ. Thuật toán Best-fit, không như tên gọi của nó, đây là một thuật toán có hiệu suất thấp nhất, trong trường hợp này hệ điều hành phải duyệt qua tất các các khối nhớ trống để tìm ra một khối nhớ có kích thước vừa đủ để chứa tiến trình vừa yêu cầu, điều này làm giảm tốc độ cấp phát của hệ điều hành. Mặt khác với việc chọn kích thước vừa đủ có thể dẫn đến sự phân mảnh lớn trên bộ nhớ, tức là có quá nhiều khối nhớ có kích thước quá nhỏ trên bộ nhớ, nhưng nếu xét về mặt lãng phí bộ nhớ tại thời điểm cấp phát thì thuật toán này làm lãng phí ít nhất. Tóm lại, khó có thể đánh giá về hiệu quả sử dụng của các thuật toán này, vì hiệu quả của nó được xét trong "tương lai" và trên nhiều khía cạnh khác nhau chứ không phải chỉ xét tại thời điểm cấp phát. Và hơn nữa trong bản thân các thuật toán này đã có các mâu thuẩn với nhau về hiệu quả sử dụng của nó.

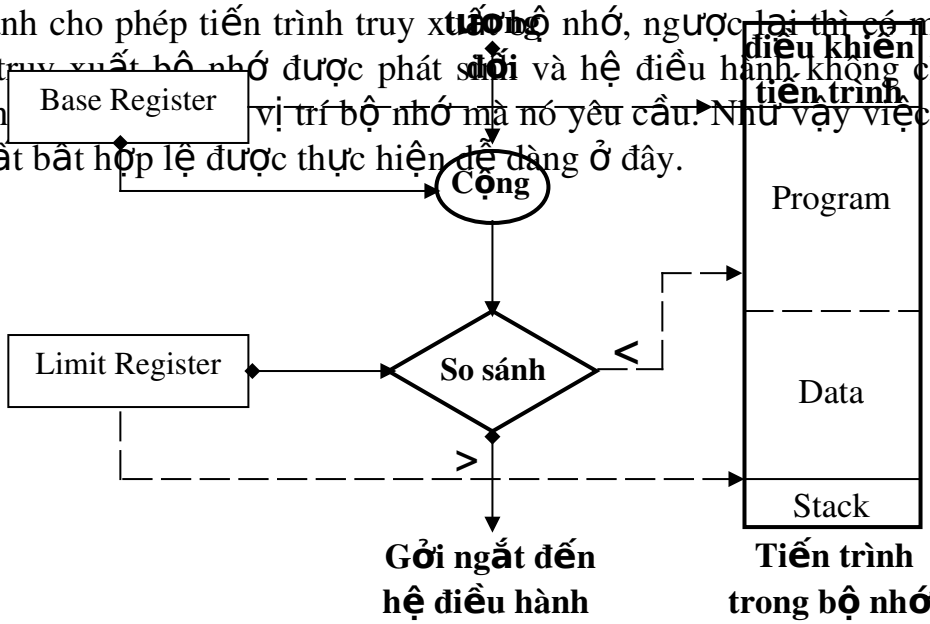
Do yêu cầu của công tác cấp phát bộ nhớ của hệ điều hành, một tiến trình đang ở trên bộ nhớ có thể bị đưa ra ngoài (swap-out) để dành chỗ nạp một tiến trình mới có yêu cầu, và tiến trình này sẽ được nạp vào lại (swap-in) bộ nhớ tại một thời điểm thích hợp sau này. Vấn đề đáng quan tâm ở đây là tiến trình có thể được nạp vào lại phân vùng khác với phân vùng mà nó được nạp vào lần đầu tiên. Có một lý do khác khiến các tiến trình phải thay đổi vị trí nạp so với ban đầu là khi có sự liên kết giữa các môdun tiến trình của một chương trình thì các tiến trình phải dịch chuyển ngay cả khi

chúng đã nằm trên bộ nhớ chính. Sự thay đổi vị trị/địa chỉ nạp này sẽ ảnh hưởng đến các thao tác truy xuất dữ liệu của chương trình vì nó sẽ khác với các địa chỉ tương đối mà người lập trình đã sử dụng trong code của chương trình. Ngoài ra khi một tiến trình được nạp vào bộ nhớ lần đầu tiên thì tất cả các địa chỉ tương đối được tham chiếu trong code chương trình được thay thế bằng địa chỉ tuyệt đối trong bộ nhớ chính, địa chỉ này được xác định bởi địa chỉ cơ sở, nơi tiến trình được nạp. Ví dụ trong chương trình có code truy xuất đến địa chỉ tương đối 100k, nếu chương trình này được nạp vào phân vùng 1 có địa chỉ bắt đầu là 100k thì địa chỉ truy xuất là 200k, nhưng nếu chương trình được nạp vào phân vùng 2 có địa chỉ bắt đầu là 200k, thì địa chỉ truy xuất sẽ là 300k. Để giải quyết vấn đề này hệ điều hành phải thực hiện các yêu cầu cần thiết của công tác tái định vị một tiến trình vào lại bộ nhớ. Ngoài ra ở đây hệ điều hành cũng phải tính đến việc bảo vệ các tiến trình trên bộ nhớ tránh tình trạng một tiến trình truy xuất đến vùng nhớ của tiến trình khác. Trong trường hợp này hệ điều hành sử dụng 2 thanh ghi đặc biệt:

Thanh ghi cơ sở (base register): dùng để ghi địa chỉ cơ sở của tiến trình tiến trình được nạp vào bộ nhớ.

Thanh ghi giới hạn (limit register): dùng để ghi địa chỉ cuối cùng của tiến trình trong bộ nhớ.

Khi một tiến trình được nạp vào bộ nhớ thì hệ điều hành sẽ ghi địa chỉ bắt đầu của phân vùng được cấp phát cho tiến trình vào thanh ghi cơ sở và địa chỉ cuối cùng của tiến trình vào thanh ghi giớ hạn. Việc thiết lập giá trị của các thanh ghi này được thực hiện cả khi tiến trình lần đầu tiên được nạp vào bộ nhớ và khi tiến trình được swap in vào lại bộ nhớ. Theo đó mỗi khi tiến trình thực hiện một thao tác truy xuất bộ nhớ thì hệ thống phải thực hiện 2 bước: Thứ nhất, cộng địa chỉ ô nhớ do tiến trình phát ra với giá trị địa chỉ trong thanh ghi cơ sở để có được địa chỉ tuyệt đối của ô nhớ cần truy xuất. Thứ hai, địa chỉ kết quả ở trên sẽ được so sánh với giá trị địa chỉ trong thành ghi giới hạn. Nếu địa chỉ nằm trong phạm vị giới hạn thì hệ điều hành cho phép tiến trình truy xuất bộ nhớ, ngược lại thì có một ngắt về lỗi truy xuất bộ nhớ được phát sinh và hệ điều hành không cho phép tiến trình truy xuất vị trí bộ nhớ mà nó yêu cầu. Như vậy việc bảo bệ truy xuất bất hợp lệ được thực hiện dễ dàng ở đây.



Hình 3.6 : Tái định vị với sự hỗ trợ của phần cứng

2.4.Quản lý bộ nhớ rỗi

Trong bộ nhớ trong, phân chia ra các vùng nhớ đã được phân phối cho các chương trình đang hoạt động và một số vùng nhớ khác loại chưa được sử dụng. Vùng bộ nhớ rỗi nói trên, khi cần tải một chương trình mới sẽ cần tới nó, và lúc đó nó đã được phân phối (bị bận). Sau khi một chương trình hoàn thiện công việc của mình, nó được kết thúc, vùng nhớ dành cho nó được giải phóng. Bài toán quản lý, phân phối bộ nhớ thường xuyên xảy ra, trong đó bài toán quản lý bộ nhớ rỗi là một bài toán quan trọng. Có một số phương pháp để quản lý bộ nhớ rỗi. Ví dụ như hệ điều hành đơn chương trình, sử dụng cận kề vùng bộ nhớ đã phân phối cho thông tin về vùng nhớ rỗi chưa được phân phối. Một trong những phương pháp quản lý bộ nhớ rỗi điển hình là phương pháp kề cận. Phương pháp kề cận có thể được chia ra: kề cận nhị phân và kề cận tổng quát.

2.4.1.phương pháp kề cận nhị phân

Phương pháp kề cận nhị phân tương ứng với việc phân chia nhị phân: Bộ nhớ được chia làm các khối nhớ có độ dài là 2^k đơn vị bộ nhớ (đơn vị phổ biến là trang), mỗi vùng bộ nhớ có độ dài 2^k phải được đặt ở địa chỉ chia hết cho 2^k.

Có mà một bảng quản lý các vùng bộ nhớ rỗi, phần từ k (k=0,1,2,…) trong bảng chứa danh sách móc nối mà các vùng bộ nhớ 2^k còn đang rỗi. Khi một chương trình đòi hỏi một vùng nhớ có đọ dài x thì thuật toán xử lý như sau: Tìm số k bé nhất để cung cấp đủ độ dài x đó (2^k>x). Nếu không có báo sai sót. Nếu có phân phối vào vùng đó: như vậy liên quan đến các thao tá: loại bỏ, bổ sung, hay vừa loại bỏ, vừa bổ sung các phần tử trong các danh sách trong bảng.

Trong các danh sách trên, chú ý có 3 danh sách không rỗng:

Danh sách với k=0: có 1 phần tử trỏ vào ô 25;

Danh sách với k=1: có 2 phần tử trỏ vào ô 2 và ô 4;

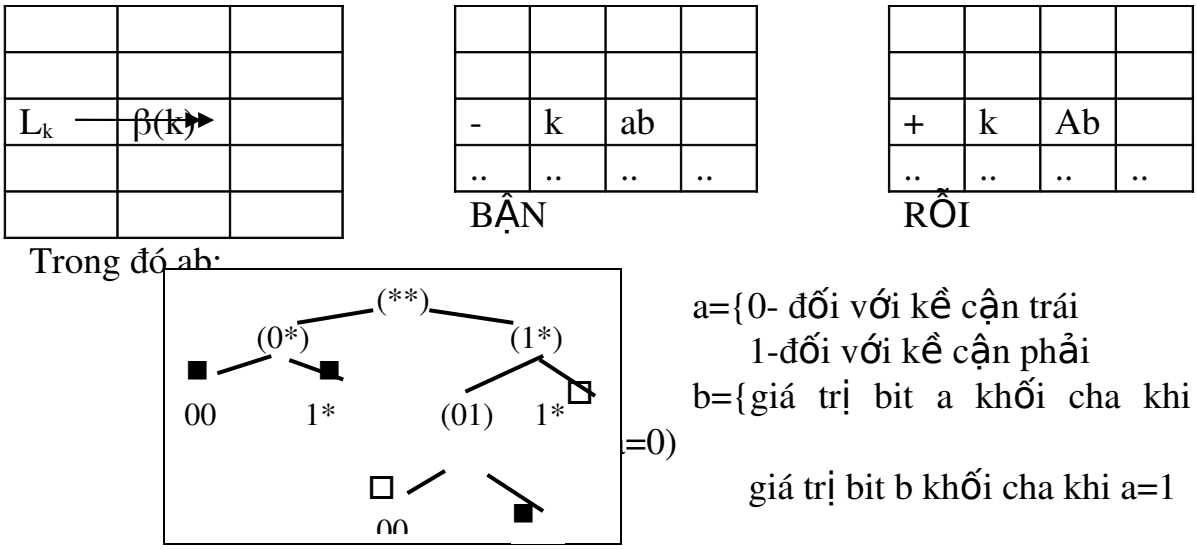
Danh sách với k=3: có 1 phần tử trỏ vào ô 16;

2.4.2.Phương pháp kề cận tổng quát

Kề cận tổng quát: nảy sinh vấn đề nếu độ dài x là 2^k +1, phải phân phối một miền bộ nhớ 2^{k+1} là khá lãng phí. Vì vậy, cải tiến các độ dài được chia không theo các vùng nhớ theo độ dài 2^k mà theo dãy nào đó, chẳng hạn theo dãy số Fibonaxy: 1,1,2,3,5,8,… ($L=L_{k-1} + L_{k-2}$, k>2). Chú ý rằng, dãy số Fibonaxy và dãy lỹ thừa 2 có cùng công thức chung:

$L_k=L_{k-1}+ L_{k-j}$,

Trong đó j hoặc 1 hoặc 2 (khi j=1 chúng ta nhận lại dãy lũy thừa 2 theo kề cận nhị phân, còn j=2 thì nhận được dãy Fibonaxy), và tổng quát hóa $L_k=L_{k-1}+L_{b(k)}$, trong đó b(k) là hàm theo k nào đó: b(k)<k.



Hình 3.7: Cây cấu trúc kề cận tổng quát

Bit a cho biết khối là kề cận tái hay phải, bit b nhận được từ dấu hiệu đã có của nút cha: bit a của cha nếu khối con đang xét là kề cận trái, và bit b của cha nếu khối con đang xét là kề cận phải. Khi tách khối, địa chỉ của kề cận trái (độ dài L_{k-1}) trùng với địa chỉ khối còn địa chỉ của kề cận con phải (độ dài $L_{\beta(k)}$) sẽ tăng L_{k-1} so với địa chỉ khối. Quá trình tách khối cho phép tìm kiếm được khối theo yêu cầu. Trình bày sự bận hoặc rỗi của từng khối được cho trong các danh sách móc nối và các bảng được thể hiện trong các hình vẽ bên trên (chú ý rằng trong các phần tử có chứa địa chỉ của mỗi khối). Tương tự như trường hợp kề cận nhị phân, sau mỗi lần giải phóng một khối thì quá trình ghép khối có thể xảy ra.

Tổng quát, cây được sử dụng có bậc lớn hơn 2 và lúc đó việc tách khối có thể thực hiện đối với cây lớn hơn.

3.Điều khiển bộ nhớ gián đoạn

Mục tiêu;

- Nắm được phương thức tối ưu hóa việc phân phối bội nhớ, tránh lãng phí và chia sẻ tài nguyên bộ nhớ.

3.1.Tổ chức gián đoạn

Như đã biết với hệ điều hành hoạt động theo chế độ đa người dùng, tại cùng một thời điểm có nhiều người cùng làm việc với máy: tồn tại nhiều chương trình đang có mặt tại bộ nhớ trong để làm việc và nói chung thì số chương trình này không giảm đi như đã xét. Vì bộ nhớ trong là rất hạn chế, có nhiều người dùng (do vậy có nhiều chương trình người dùng đang ở trong bộ nhớ trong) và chương trình người dùng có độ dài không thể giới hạn trước và vì vậy, không phải toàn bộ chương trình người dùng nào cũng phải trong bộ nhớ trong: một bộ phận nằm ở bộ nhớ trong và bộ phận còn lại nằm ở bộ nhớ ngoài. Để liên kết các bộ phận nói trên, không

thể sử dụng địa chỉ tương đối trong phân phối liên tục mà các bộ phận này phải thống nhất với nhau về hệ thống địa chỉ, các lệnh quy chiếu đến các địa chỉ thống nhất đó. Như vậy, với một chương trình người dùng, các địa chỉ thuộc vào không gian địa chỉ thực và không gian địa chỉ ảo.

Bộ nhớ trong được địa chỉ hóa (bằng số, bắt đầu là địa chỉ 0) và CPU trực tiếp thao tác lấy và ghi bộ nhớ đối với những địa chỉ thuộc bộ nhớ trong một tập hợp nào đó; tập hợp các địa chỉ nói trên được gọi là không gian địa chỉ thực. Lực lượng của không gian địa chỉ thực luôn được xác định trước và gắn với máy.

Trong chương trình(không phải viết trên ngôn ngữ máy), người lập trình hướng đến bộ nhớ qua tập hợp các tên logic là ký hiệu chứ không hoàn toàn là số - địa chỉ thực: một cách tổng quát, địa chỉ được biểu thị bằng tên, các tên nói trên tạo ra một không gian tên. Một chương trình được viết như một thể thống nhất có mối liên hệ giữa các tên nói trên. Tập hợp các tên sử dụng chưa được xác định trước. Tập hợp các tên – địa chỉ có lực lượng vượt quá địa chỉ có thực trong bộ nhớ. Với nhiều người dùng, một "tên" không phải gắn với một "định vị cố định" nào cả. Mặt khác, việc dùng tên của các người lập trình khác nhau là độc lập nhau, vì thế hệ thống cho phép không gian trên được phép dùng là "vô hạn".

Hệ thống chương trình cần phải định vị được "bộ nhớ " đối với mỗi tên trong chương trình: cần ánh xạ không gian tên vào địa chỉ vật lý và trong ánh xạ đó nảy sinh khái niệm không gian địa chỉ ảo. Ánh xạ từ không gian têm tới bộ nhớ vật lý được chia làm hai bước.

Bước 1: Do chương trình dịch đảm nhận. Việc xác định địa chỉ ảo không phải do chương trình người dùng hoặc hệ thống phần cứng mà do chương trình dịch trong hệ thống: địa chỉ ảo có thể là ký hiệu, số hoặc chỉ dẫn ra số. Tập hợp các địa chỉ ảo(do chương trình dịch trong hệ thống thiết lập) được gọi là không gian địa chỉ ảo (ngắn gọn là không gian địa chỉ).

Bước 2: Do hệ điều hành (cụ thể là do điều khiển bộ nhớ) ánh xạ địa chỉ ảo vào bộ nhớ vật lý. Tại giai đoạn này xảy ra quá trình tải bộ phận của chương trình vào bộ nhớ trong tại một vùng nhớ còn rỗi. Chương trình được tải trong bộ nhớ trong theo tập hợp các vùng nhớ rời rạc nhau đang dành cho nó.

Trong việc kiến thiết nảy sinhn các trường hợp:

- Đồng nhất không gian địa chỉ với bộ nhớ vật lý: ánh xạ chỉ cần chương trình hệ thống khi sinh mã máy chương trình, hệ điều hành chỉ đảm bảo phân phối liên tục cố định bộ nhớ. Assembler với tải và sử dụng trực tiếp là ví dụ cho trường hợp này.

- Đồng nhất không gian địa chỉ với không gian tên: Đảm bảo bằng hệ điều hành khi sử dụng bằng ký hiệu và hướng dẫn. Một ví dụ cho trường hợp này là trình thông dịch của APL trên IBM 370.

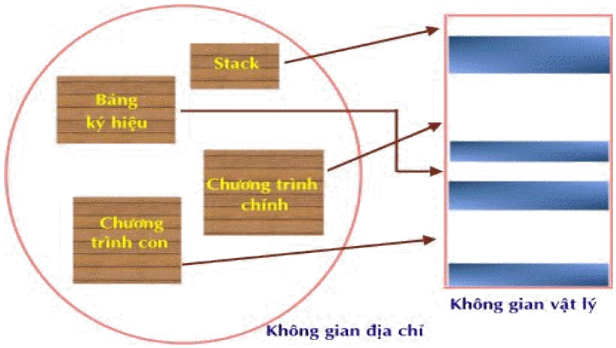
- Bộ dich sinh ra các địa chỉ tương đối, về bản chất được coi là địa chỉ ảo và sau đó hướng chương trình tới một đoạn nhớ liên tục. Sau khi tải, địa chỉ ảo bị xóa bỏ và truy nhập trực tiếp tới địa chỉ thực.

- Biện pháp giải quyết mềm rẻo nhất là bộ dịch xem xét địa chỉ ảo như là các địa chỉ tương đối và thông tin về địa chỉ đầu; còn hệ điều hành thực còn ánh xạ thứ hai không phải qua một bước mà là qua một số bước: thuật ngữ bộ nhớ ảo liên quan đến hệ thống bảo quản không gian địa chỉ ảo hiện tại của hệ thống. một địa chỉ ảo không phải luôn luôn hướng tới một địa chỉ bộ nhớ trong duy nhất. Biện pháp này thể hiện trong điều khiển theo segmen.

Nói chung, sử dụng bộ nhớ ảo đòi hỏi phải có cơ chế định vị lại địa chỉ (bước 2 nêu trên) mỗi khi tải lại chương trình và điều đó là hoàn toàn khác với phân phối liên tục. Ở chế độ phân phối bộ nhớ rời rạc, không diễn ra việc thay lại địa chỉ trong nội dung chương trình hay thay nội dung chương trình (không cho phép chương trình tự biến đổi mình).

3.2.Điều khiển bộ nhớ phân đoạn

Quan niệm không gian địa chỉ là một tập các *phân đoạn (segments)* – các phân đoạn là những phần bộ nhớ *kích thước khác nhau và có liên hệ logic với nhau*. Mỗi phân đoạn có một tên gọi (số hiệu phân đoạn) và một độ dài. Người dùng sẽ thiết lập mỗi địa chỉ với hai giá trị : *<số hiệu phân đoạn, offset>*.



Hình 3.8: Mô hình phân đoạn bộ nhớ

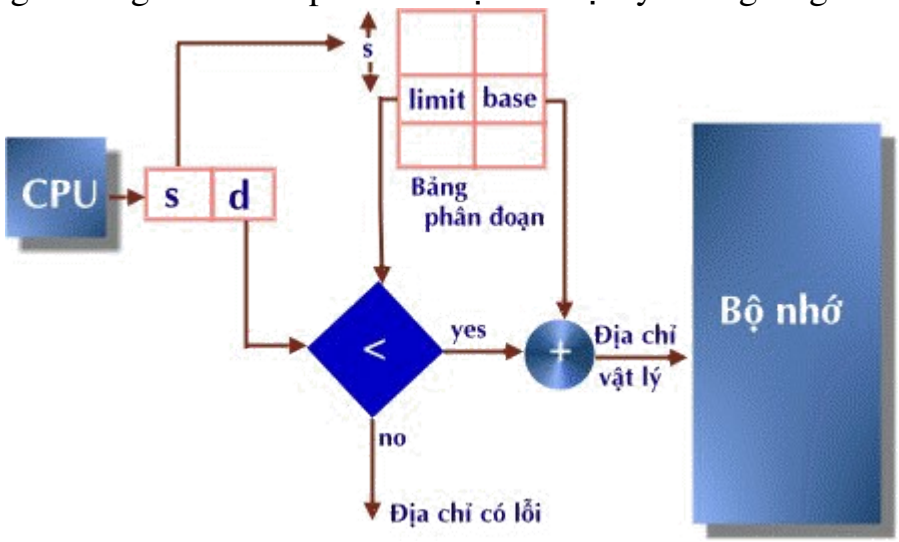
Cơ chế MMU trong kỹ thuật phân đoạn:

Cần phải xây dựng một ánh xạ để chuyển đổi các địa chỉ 2 chiều được người dùng định nghĩa thành địa chỉ vật lý một chiều. Sự chuyển đổi này được thực hiện qua một *bảng phân đoạn*. Mỗi thành phần trong bảng phân đoạn bao gồm một *thanh ghi nền* và một *thanh ghi giới hạn*. Thanh ghi nền lưu trữ địa chỉ vật lý nơi bắt đầu phân đoạn trong bộ nhớ, trong khi thanh ghi giới hạn đặc tả chiều dài của phân đoạn.

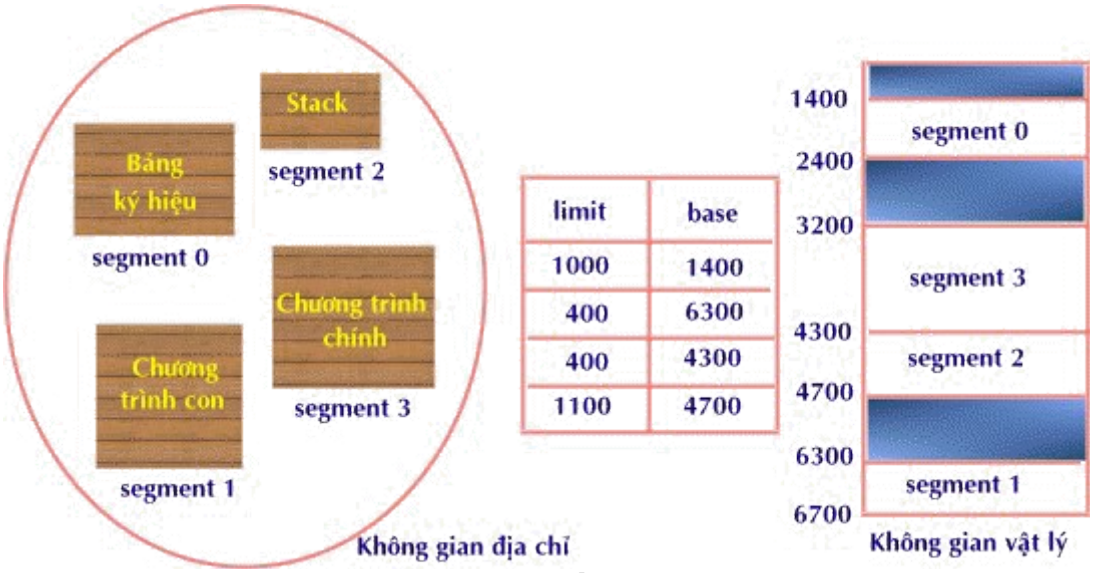
Chuyển đổi địa chỉ:

Mỗi địa chỉ ảo là một bộ *<s,d>* :

- *Số hiệu phân đoạn s* : được sử dụng như chỉ mục đến bảng phân đoạn
- *Địa chỉ tương đối d* : có giá trị trong khoảng từ 0 đến giới hạn chiều dài của phân đoạn. Nếu địa chỉ tương đối hợp lệ, nó sẽ được cộng với giá trị chứa trong thanh ghi nền để phát sinh địa chỉ vật lý tương ứng.



Hình3.9: Cơ chế phần cứng hổ trợ kĩ thuật phân đoạn

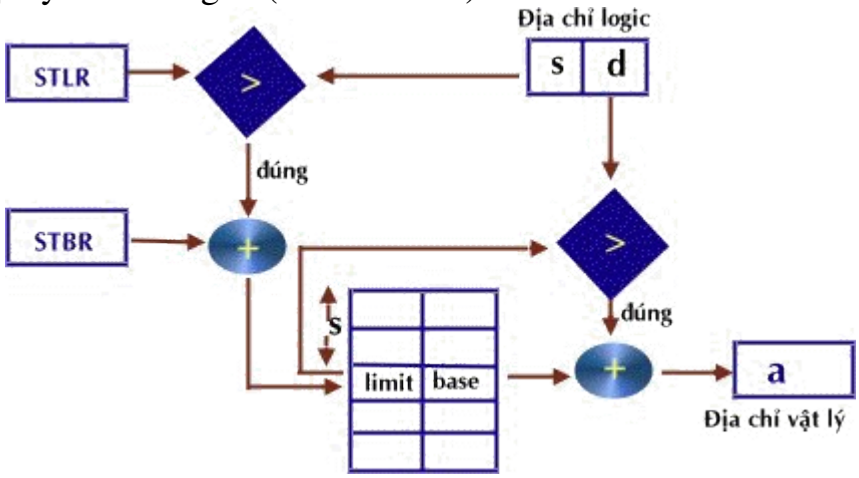


Hình 3.10: Hệ thống phân đoạn

Cài đặt bảng phân đoạn:

Có thể sử dụng các thanh ghi để lưu trữ bảng phân đoạn nếu số lượng phân đoạn nhỏ. Trong trường hợp chương trình bao gồm quá nhiều phân đoạn, bảng phân đoạn phải được lưu trong bộ nhớ chính. Một *thanh ghi nền bảng phân đoạn* (STBR) chỉ đến địa chỉ bắt đầu của bảng phân đoạn. Vì số lượng phân đoạn sử dụng trong một chương trình biến động, cần sử dụng thêm một *thanh ghi đặc tả kích thước bảng phân đoạn* (STLR).

Với một địa chỉ logic <*s,d*>, trước tiên số hiệu phân đoạn *s* được kiểm tra tính hợp lệ (s <STLR). Kế tiếp, cộng giá trị s với STBR để có được địa chỉ địa chỉ của phần tử thứ *s* trong bảng phân đoạn (STBR+*s*). Đia chỉ vật lý cuối cùng là (STBR+*s* + *d*)

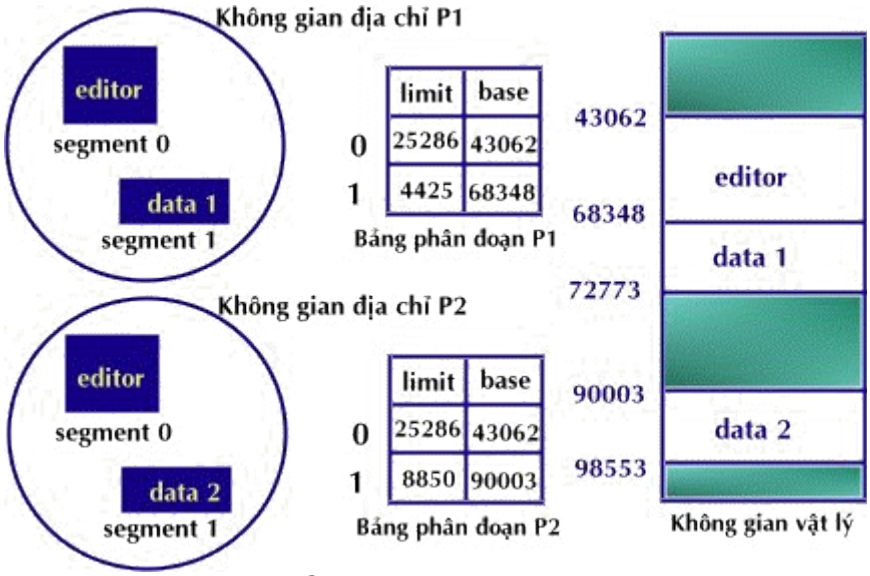


Hình 3.11. Sử dụng STBR, STLR và bảng phân đoạn

Bảo vệ: Một ưu điểm đặc biệt của cơ chế phân đoạn là khả năng đặc tả thuộc tính bảo vệ cho mỗi phân đoạn. Vì mỗi phân đoạn biểu diễn cho một phần của chương trình với ngữ nghĩa được người dùng xác định, người sử dụng có thể biết được một phân đoạn chứa đựng những gì bên trong, do vậy họ có thể đặc tả các thuộc tính bảo vệ thích hợp cho từng phân đoạn.

Cơ chế phần cứng phụ trách chuyển đổi địa chỉ bộ nhớ sẽ kiểm tra các bit bảo vệ được gán với mỗi phần tử trong bảng phân đoạn để ngăn chặn các thao tác truy xuất bất hợp lệ đến phân đoạn tương ứng.

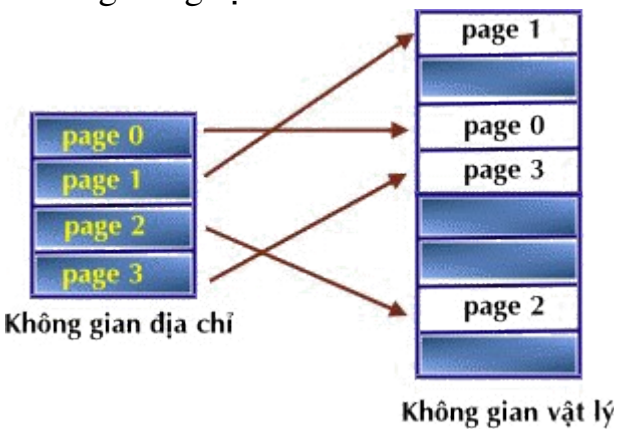
Chia sẻ phân đoạn: Một ưu điểm khác của kỹ thuật phân đoạn là khả năng chia sẻ ở mức độ phân đoạn. Nhờ khả năng này, các tiến trình có thể chia sẻ với nhau từng phần chương trình (ví dụ các thủ tục, hàm), không nhất thiết phải chia sẻ toàn bộ chương trình như trường hợp phân trang. Mỗi tiến trình có một bảng phân đoạn riêng, một phân đoạn được chia sẻ khi các phần tử trong bảng phân đoạn của hai tiến trình khác nhau cùng chỉ đến một vị trí vật lý duy nhất.



Hình 3.12 Chia sẻ code trong hệ phân đoạn

3.3.Điều khiển bộ nhớ phân trang

Phân bộ nhớ vật lý thành các khối (block) có kích thước cố định và bằng nhau, gọi là *khung trang (page frame)*. Không gian địa chỉ cũng được chia thành các khối có cùng kích thước với khung trang, và được gọi là *trang* (*page*). Khi cần nạp một tiến trình để xử lý, các trang của tiến trình sẽ được nạp vào những khung trang còn trống. Một tiến trình kích thước N trang sẽ yêu cầu N khung trang tự do.



Hình 3.13 Mô hình bộ nhớ phân trang

Cơ chế MMU trong kỹ thuật phân trang:

Cơ chế phần cứng hỗ trợ thực hiện chuyển đổi địa chỉ trong cơ chế phân trang là bảng trang (*pages table*). Mỗi phần tử trong bảng trang cho biết các địa chỉ bắt đầu của vị trí lưu trữ trang tương ứng trong bộ nhớ vật lý (số hiệu khung trang trong bộ nhớ vật lý đang chứa trang).

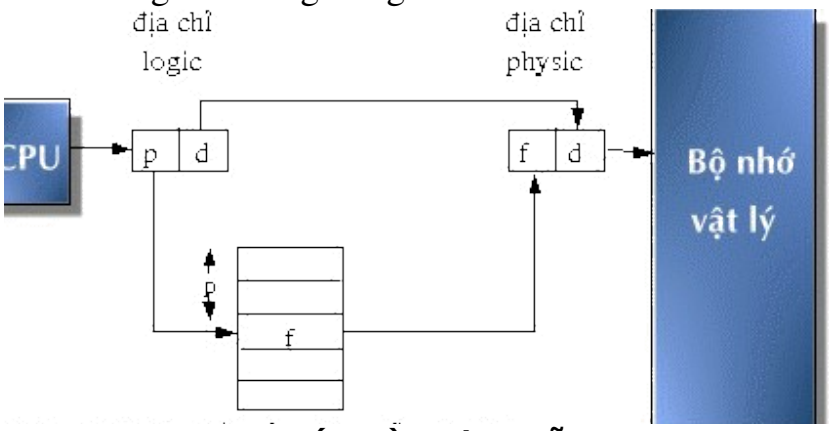
Chuyển đổi địa chỉ:

Mỗi địa chỉ phát sinh bởi CPU được chia thành hai phần:

- *số hiệu trang (p)*: sử dụng như chỉ mục đến phần tử tương ứng trong bảng trang.

- *địa chỉ tương đối trong trang (d)*: kết hợp với địa chỉ bắt đầu của trang để tạo ra địa chỉ vật lý mà trình quản lý bộ nhớ sử dụng.

Kích thước của trang do phần cứng qui định. Để dễ phân tích địa chỉ ảo thành số hiệu trang và địa chỉ tương đối, kích thước của một trang thông thường là một lũy thừa của 2 (biến đổi trong phạm vi 512 bytes và 8192 bytes). Nếu kích thước của không gian địa chỉ là 2^m và kích thước trang là 2^n, thì *m-n* bits cao của địa chỉ ảo sẽ biểu diễn số hiệu trang, và *n* bits thấp cho biết địa chỉ tương đối trong trang.

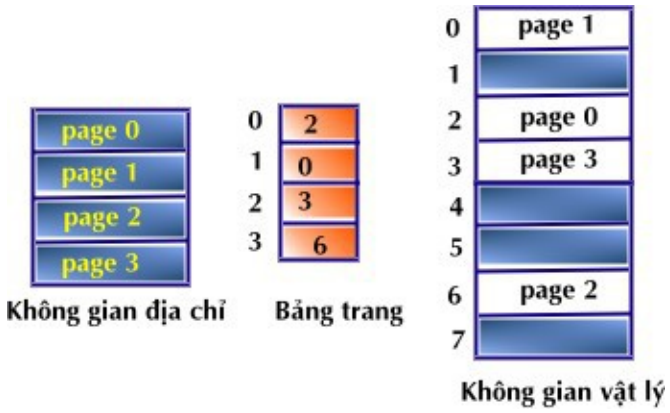


Hình 3. 14: Cơ chế phần cứng hỗ trợ phân trang

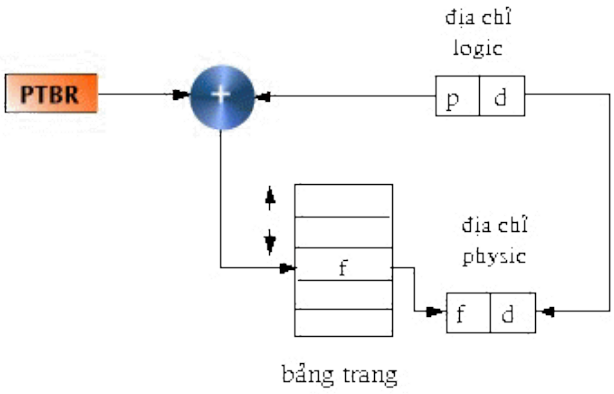
Cài đặt bảng trang:

Trong trường hợp đơn giản nhất, bảng trang một tập các thanh ghi được sử dụng để cài đặt bảng trang. Tuy nhiên việc sử dụng thanh ghi chỉ phù hợp với các bảng trang có kích thước nhỏ, nếu bảng trang có kích thước lớn, nó phải được lưu trữ trong bộ nhớ chính, và sử dụng một thanh ghi để lưu địa chỉ bắt đầu lưu trữ bảng trang (PTBR).

Theo cách tổ chức này, mỗi truy xuất đến dữ liệu hay chỉ thị đều đòi hỏi hai lần truy xuất bộ nhớ : một cho truy xuất đến bảng trang và một cho bản thân dữ liệu!



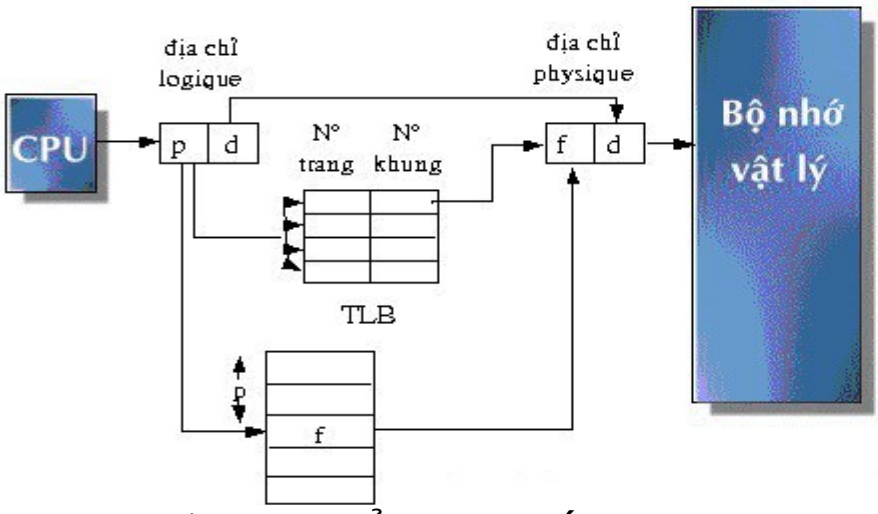
Hình 3.15: Mô hình bộ nhớ phân trang



Hình 3.16 : Sử dụng thanh ghi nền trỏ đến bảng trang

Có thể né tránh bớt việc truy xuất bộ nhớ hai lần bằng cách sử dụng thêm một vùng nhớ đặc biệt , với tốc độ truy xuất nhanh và cho phép tìm kiếm song song, vùng nhớ cache nhỏ này thường được gọi là bộ nhớ kết hợp (TLBs). Mỗi thanh ghi trong bộ nhớ kết hợp gồm một từ khóa và một giá trị, khi đưa đến bộ nhớ kết hợp một đối tượng cần tìm, đối tượng này sẽ được so sánh cùng lúc với các từ khóa trong bộ nhớ kết hợp để tìm ra phần tử tương ứng. Nhờ đặc tính này mà việc tìm kiếm trên bộ nhớ kết hợp được thực hiện rất nhanh, nhưng chi phí phần cứng lại cao.

Trong kỹ thuật phân trang, TLBs được sử dụng để lưu trữ các trang bộ nhớ được truy cập gần hiện tại nhất. Khi CPU phát sinh một địa chỉ, số hiệu trang của địa chỉ sẽ được so sánh với các phần tử trong TLBs, nếu có trang tương ứng trong TLBs, thì sẽ xác định được ngay số hiệu khung trang tương ứng, nếu không mới cần thực hiện thao tác tìm kiếm trong bảng trang.



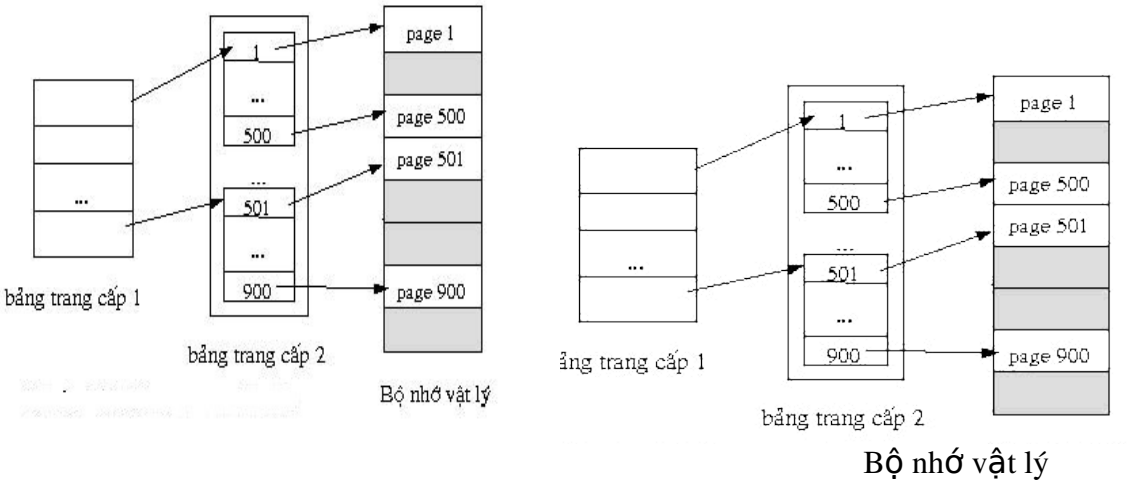
Hình 3.17: Bảng trang với TLBs

Tổ chức bảng trang:

Mỗi hệ điều hành có một phương pháp riêng để tổ chức lưu trữ bảng trang. Đa số các hệ điều hành cấp cho mỗi tiến trình một bảng trang. Tuy nhiên phương pháp này không thể chấp nhận được nếu hệ điều hành cho

phép quản lý một không gian địa chỉ có dung lượng quá (2^{32}, 2^{64}): trong các hệ thống như thế, bản thân bảng trang đòi hỏi một vùng nhớ qúa lớn! Có hai giải pháp cho vấn đề này:

- *Phân trang đa cấp:* phân chia bảng trang thành các phần nhỏ, bản thân bảng trang cũng sẽ được phân trang



Hình 3.18: Bảng trang nhị cấp

Bảng trang nghịch đảo: sử dụng duy nhất một *bảng trang nghịch đảo* cho tất cả các tiến trình . Mỗi phần tử trong *bảng trang nghịch đảo* phản ánh một khung trang trong bộ nhớ bao gồm địa chỉ logic của một trang đang được lưu trữ trong bộ nhớ vật lý tại khung trang này, cùng với thông tin về tiến trình đang được sỡ hữu trang. Mỗi địa chỉ ảo khi đó là một bộ ba <idp, p, d >

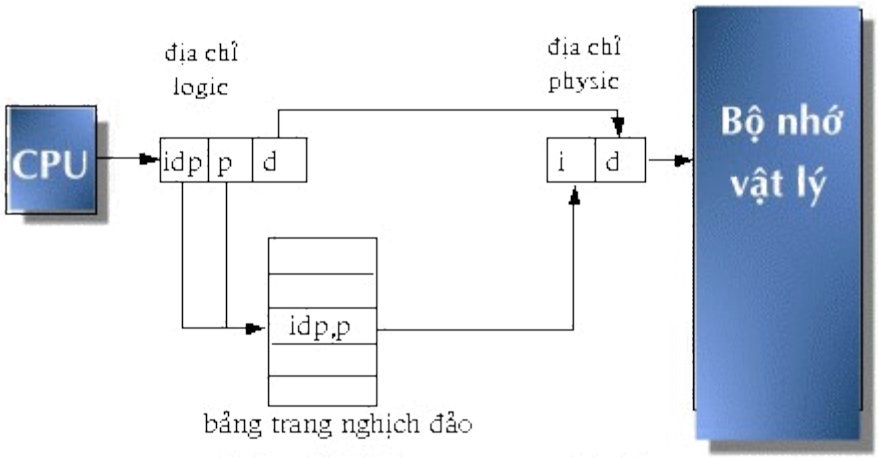
Trong đó :

idp là định danh của tiến trình

p là số hiệu trang

d là địa chỉ tương đối trong trang

Mỗi phần tử trong bảng trang nghịch đảo là một cặp <idp, p >. Khi một tham khảo đến bộ nhớ được phát sinh, một phần địa chỉ ảo là <idp, p > được đưa đến cho trình quản lý bộ nhớ để tìm phần tử tương ứng trong bảng trang nghịch đảo, nếu tìm thấy, địa chỉ vật lý <i,d> sẽ được phát sinh. Trong các trường hợp khác, xem như tham khảo bộ nhớ đã truy xuất một địa chỉ bất hợp lệ.



Hình 3.19: bảng trang nghịch đảo

Bảo vệ:

Cơ chế bảo vệ trong hệ thống phân trang được thực hiện với các bit bảo vệ được gắn với mỗi khung trang. Thông thường , các bit này được lưu trong bảng trang , vì mỗi truy xuất đến bộ nhớ đều phải tham khảo đến bảng trang để phát sinh địa chỉ vật lý, khi đó, hệ thống có thể kiểm tra các thao tác truy xuất trên khung trang tương ứng có hợp lệ với thuộc tính bảo vệ của nó không.

Ngoài ra, một bit phụ trội được thêm vào trong cấu trúc một phần tử của bảng trang : bit hợp lệ-bất hợp lệ (valid-invalid).

Hợp lệ : trang tương ứng thuộc về không gian địa chỉ của tiến trình.

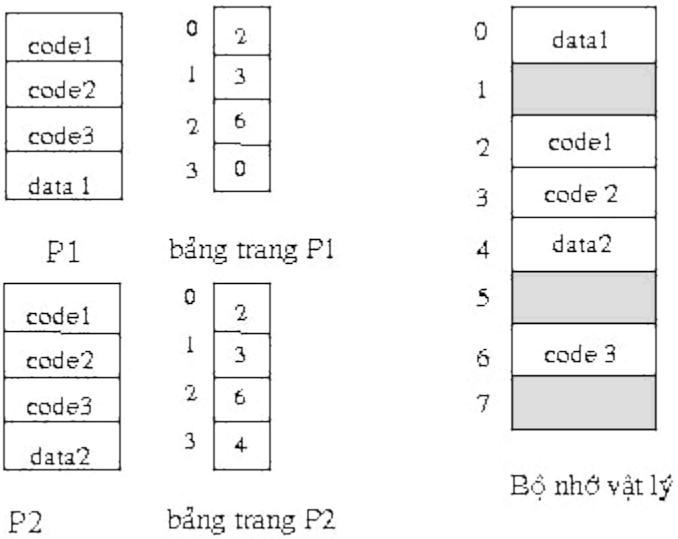
Bất hợp lệ : trang tương ứng không nằm trong không gian địa chỉ của tiến trình, điều này có nghĩa tiến trình đã truy xuất đến một địa chỉ không được phép.

số hiệu khung trang	bit valid-invalid

Hình 3.20 Cấu trúc một phần tử trong bảng trang.

Chia sẻ bộ nhớ trong cơ chế phân trang:

Một ưu điểm của cơ chế phân trang là cho phép chia sẻ các trang giữa các tiến trình.Trong trường hợp này, sự chia sẻ được thực hiện bằng cách ánh xạ nhiều địa chỉ logic vào một địa chỉ vật lý duy nhất. Có thể áp dụng kỹ thuật này để cho phép có tiến trình chia sẻ một vùng code chung*:* nếu có nhiều tiến trình của cùng một chương trình, chỉ cần lưu trữ một đoạn code của chương trình này trong bộ nhớ, các tiến trình sẽ có thể cùng truy xuất đến các trang chứa code chung này. Lưu ý để có thể chia sẻ một đoạn code, đoạn code này phải có thuộc tính *reenterable* (cho phép một bản sao của chương trình được sử dụng đồng thời bởi nhiều tác vụ).



Hình 3.21 chia sẻ các trang trong hệ phân trang

CÂU HỎI CỦNG CỐ BÀI HỌC

1.Trình bày được nguyên lý điều khiển bộ nhớ của HĐH, phương thức tối ưu hóa việc phân phối bội nhớ.
2.Trình bày điều khiển bộ nhớ liên tục
3. Trình bày điều khiển bộ nhớ gián đoạn
4. Xét một hệ thống trong đó một chương trình khi được nạp vào bộ nhớ sẽ phân biệt hoàn toàn phân đoạn code và phân đoạn data. Giả sử CPU sẽ xác định được khi nào cần truy xuất lệnh hay dữ liệu , và phải truy xuất ở đâu. Khi đó mỗi chương trình sẽ được cung cấp 2 bộ thanh ghi base-limit : một cho phân đoạn code, và một cho phân đoạn data. Bộ thanh ghi base-limit của phân đoạn code tự động được đặt thuộc tính readonly. Thảo luận các ưu và khuyết điểm của hệ thống này.
5.Tại sao kích thước trang luôn là lũy thừa của 2 ?
6.Xét một không gian địa chỉ có 8 trang, mỗi trang có kích thước 1K. ánh xạ vào bộ nhớ vật lý có 32 khung trang.
a) Địa chỉ logic gồm bao nhiêu bit ?
b) Địa chỉ physic gồm bao nhiêu bit ?
7.Tại sao trong hệ thống sử dụng kỹ thuật phân trang, một tiến trình không thể truy xuất đến vùng nhớ không được cấp cho nó ? Làm cách nào hệ điều hành có thể cho phép sự truy xuất này xảy ra ? Hệ điều hành có nên cho phép điều đó không ? Tại sao ?

CHƯƠNG 3:
ĐIỀU KHIỂN CPU, ĐIỀU KHIỂN QUÁ TRÌNH

Mã chương: MH15-04

Giới thiệu:

Những hệ thống máy tính ban đầu cho phép chỉ một chương trình được thực thi tại một thời điểm. Chương trình này có toàn quyền điều khiển hệ thống và có truy xuất tới tất cả tài nguyên của hệ thống. Những hệ thống máy tính hiện nay cho phép nhiều chương trình được nạp vào bộ nhớ và được thực thi đồng hành. Sự phát triển này yêu cầu sự điều khiển mạnh mẽ hơn và phân chia nhiều hơn giữa các quá trình. Yêu cầu này dẫn đến khái niệm quá trình, một chương trình đang thực thi. Quá trình là một đơn vị công việc trong một hệ điều hành chia thời hiện đại.

Một hệ điều hành phức tạp hơn được mong đợi nhiều hơn trong việc thực hiện các hành vi của người dùng. Mặc dù quan tâm chủ yếu của hệ điều hành là thực thi chương trình người dùng, nhưng nó cũng quan tâm đến các tác vụ khác nhau bên ngoài nhân. Do đó, một hệ thống chứa tập hợp các quá trình: quá trình hệ điều hành thực thi mã hệ thống, quá trình người dùng thực thi mã người dùng. Tất cả quá trình này có tiềm năng thực thi đồng hành, với một CPU (hay nhiều CPU) được đa hợp giữa chúng. Bằng cách chuyển đổi CPU giữa các quá trình, hệ điều hành có thể làm cho máy tính hoạt động với năng suất cao hơn**.**

Mục Tiêu:

- Nắm nguyên lý điều phối các quá trình được thực hiện trên CPU, tối ưu hóa sử dụng tài nguyên CPU, các giải pháp lập lịch mà hệ điều hành thực hiện nhằm điều phối các quá trình được thực hiện trên CPU.
- Hiểu được các nguyên nhân gây bế tắc của hệ thống và cách phòng ngừa, xử lý bế tắc**.**
- Rèn luyện khả năng tư duy, lập luận có tính khoa học
- Tinh thần hỗ trợ nhau trong học tập.

Nội dung chính:

1.Trạng thái của quá trình

Mục tiêu:

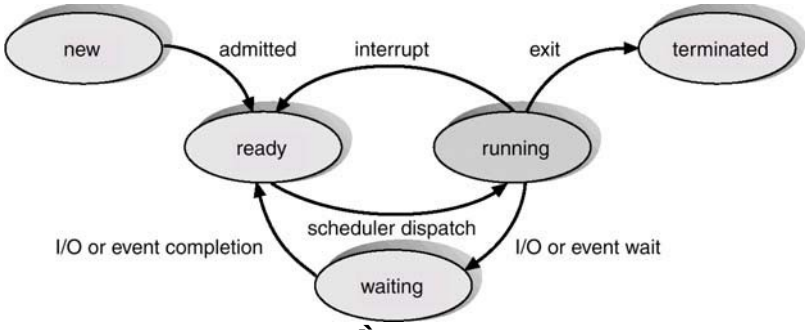
- Nắm được trạng thái của quá trình.

Khi một quá trình thực thi, nó thay đổi trạng thái.Trạng thái của quá trình được định nghĩa bởi các hoạt động hiện hành của quá trình đó. Mỗi quá trình có thể ở một trong những trạng thái sau:

- Mới (new): quá trình đang được tạo ra
- Đang chạy (running): các chỉ thị đang được thực thi

- Chờ (waiting): quá trình đang chờ sự kiện xảy ra (như hoàn thành việc nhập/xuất hay nhận tín hiệu)
- Sẵn sàng (ready): quá trình đang chờ được gán tới một bộ xử lý.
- Kết thúc (terminated): quá trình hoàn thành việc thực thi

Các tên trạng thái này là bất kỳ, và chúng khác nhau ở các hệ điều hành khác nhau. Tuy nhiên, các trạng thái mà chúng hiện diện được tìm thấy trên tất cả hệ thống. Các hệ điều hành xác định mô tả trạng thái quá trình. Chỉ một quá trình có thể đang chạy tức thì trên bất kỳ bộ xử lý nào mặc dù nhiều quá trình có thể ở trạng thái sẵn sàng và chờ.



Hình 4.1 Lưu đồ trạng thái quá trình

1.1.Chế độ xử lý của tiến trình

Để đảm bảo hệ thống hoạt động đúng đắn, hệ điều hành cần phải được bảo vệ khỏi sự xâm phạm của các tiến trình. Bản thân các tiến trình và dữ liệu cũng cần được bảo vệ để tránh các ảnh hưởng sai lạc lẫn nhau. Một cách tiếp cận để giải quyết vấn đề là phân biệt hai chế độ xử lý cho các tiến trình : *chế độ không đặc quyền* và *chế độ đặc quyền* nhờ vào sự trợ giúp của cơ chế phần cứng. Tập lệnh của CPU được phân chia thành các lệnh đặc quyền và lệnh không đặc quyền. Cơ chế phần cứng chỉ cho phép các lệnh đặc quyền được thực hiện trong chế độ đặc quyền. Thông thường chỉ có hệ điều hành hoạt động trong chế độ đặc quyền, các tiến trình của người dùng hoạt động trong chế độ không đặc quyền, không thực hiện được các lệnh đặc quyền có nguy cơ ảnh hưởng đến hệ thống. Như vậy hệ điều hành được bảo vệ. Khi một tiến trình người dùng gọi đến một lời gọi hệ thống, tiến trình của hệ điều hành xử lý lời gọi này sẽ hoạt động trong chế độ đặc quyền, sau khi hoàn tất thì trả quyền điều khiển về cho tiến trình người dùng trong chế độ không đặc quyền.

Hình 4.2 Hai chế độ xử lý

1.2.Cấu trúc dữ liệu khối quản lý tiến trình

Hệ điều hành quản lý các tiến trình trong hệ thống thông qua khối quản lý tiến trình (process control block -PCB). PCB là một vùng nhớ lưu trữ các thông tin mô tả cho tiến trình, với các thành phần chủ yếu bao gồm :

- Định danh của tiến trình (1) : giúp phân biệt các tiến trình
- Trạng thái tiến trình (2): xác định hoạt động hiện hành của tiến trình.
- Ngữ cảnh của tiến trình (3): mô tả các tài nguyên tiến trình đang trong quá trình, hoặc để phục vụ cho hoạt động hiện tại, hoặc để làm cơ sở phục hồi hoạt động cho tiến trình, bao gồm các thông tin về:

- *Trạng thái CPU*: bao gồm nội dung các thanh ghi, quan trọng nhất là con trỏ lệnh IP lưu trữ địa chỉ câu lệnh kế tiếp tiến trình sẽ xử lý. Các thông tin này cần được lưu trữ khi xảy ra một ngắt, nhằm có thể cho phép phục hồi hoạt động của tiến trình đúng như trước khi bị ngắt.

-*Bộ xử lý*: dùng cho máy có cấu hình nhiều CPU, xác định số hiệu CPU mà tiến trình đang sử dụng.

-*Bộ nhớ chính*: danh sách các khối nhớ được cấp cho tiến trình.

-*Tài nguyên sử dụng*: danh sách các tài mguyên hệ thống mà tiến trình đang sử dụng.

-*Tài nguyên tạo lập*: danh sách các tài nguyên được tiến trình tạo lập.

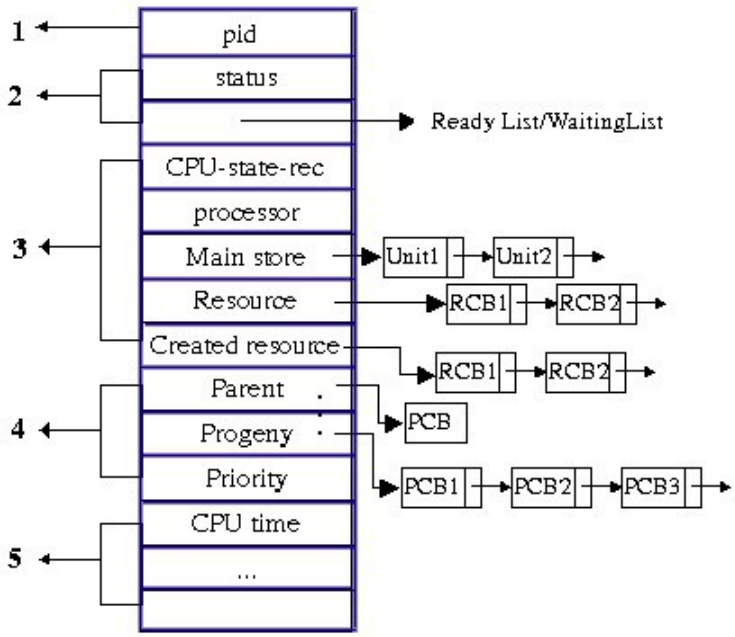
- Thông tin giao tiếp (4): phản ánh các thông tin về quan hệ của tiến trình với các tiến trình khác trong hệ thống :

-*Tiến trình cha*: tiến trình tạo lập tiến trình này .

-*Tiến trình con*: các tiến trình do tiến trình này tạo lập .

-*Độ ưu tiên* : giúp bộ điều phối có thông tin để lựa chọn tiến trình được cấp CPU.

- **Thông tin thống kê (5)**: đây là những thông tin thống kê về hoạt động của tiến trình, như thời gian đã sử dụng CPU,thời gian chờ. Các thông tin này có thể có ích cho công việc đánh giá tình hình hệ thống và dự đoán các tình huống tương lai.



Hình 4.3 Khối mô tả tiến trình

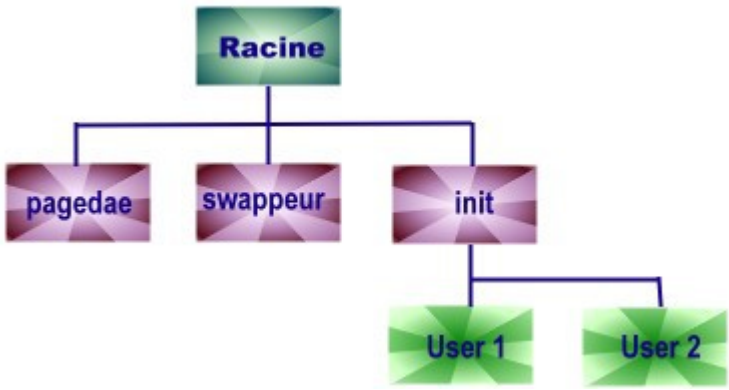
1.3.Thao tác trên tiến trình

Hệ điều hành cung cấp các thao tác chủ yếu sau đây trên một tiến trình :

- Tạo lập tiến trình (create)
- Kết thúc tiến trình (destroy)
- Tạm dừng tiến trình (suspend)
- Tái kích hoạt tiến trình (resume)
- Thay đổi độ ưu tiên tiến trình

1.3.1.Tạo lập tiến trình

Trong quá trình xử lý, một tiến trình có thể tạo lập nhiều tiến trình mới bằng cách sử dụng một lời gọi hệ thống tương ứng. Tiến trình gọi lời gọi hệ thống để tạo tiến trình mới sẽ được gọi là tiến trình *cha*, tiến trình được tạo gọi là tiến trình *con*. Mỗi tiến trình con đến lượt nó lại có thể tạo các tiến trình mới…quá trình này tiếp tục sẽ tạo ra một *cây tiến trình*.



Hình 4.4.Một cây tiến trình trong hệ thống UNIX

Các công việc hệ điều hành cần thực hiện khi tạo lập tiến trình bao gồm :

- Định danh cho tiến trình mới phát sinh
- Đưa tiến trình vào danh sách quản lý của hệ thống
- Xác định độ ưu tiên cho tiến trình
- Tạo PCB cho tiến trình
- Cấp phát các tài nguyên ban đầu cho tiến trình

Khi một tiến trình tạo lập một tiến trình con, tiến trình con có thể sẽ được hệ điều hành trực tiếp cấp phát tài nguyên hoặc được tiến trình cha cho thừa hưởng một số tài nguyên ban đầu.

Khi một tiến trình tạo tiến trình mới, tiến trình ban đầu có thể xử lý theo một trong hai khả năng sau :

- Tiến trình cha tiếp tục xử lý đồng hành với tiến trình con.
- Tiến trình cha chờ đến khi một tiến trình con nào đó, hoặc tất cả các tiến trình con kết thúc xử lý.

Các hệ điều hành khác nhau có thể chọn lựa các cài đặt khác nhau để thực hiện thao tác tạo lập một tiến trình.

1.3.2.Kết thúc tiến trình

Một tiến trình kết thúc xử lý khi nó hoàn tất chỉ thị cuối cùng và sử dụng một lời gọi hệ thống để yêu cầu hệ điều hành hủy bỏ nó. Đôi khi một tiến trình có thể yêu cầu hệ điều hành kết thúc xử lý của một tiến trình khác. Khi một tiến trình kết thúc, hệ điều hành thực hiện các công việc :

- thu hồi các tài nguyên hệ thống đã cấp phát cho tiến trình
- hủy tiến trình khỏi tất cả các danh sách quản lý của hệ thống
- hủy bỏ PCB của tiến trình

Hầu hết các hệ điều hành không cho phép các tiến trình con tiếp tục tồn tại nếu tiến trình cha đã kết thúc. Trong những hệ thống như thế, hệ điều hành sẽ tự động phát sinh một loạt các thao tác kết thúc tiến trình con.

1.3.3.Cấp phát tài nguyên cho tiến trình

Khi có nhiều người sử dụng đồng thời làm việc trong hệ thống, hệ điều hành cần phải cấp phát các tài nguyên theo yêu cầu cho mỗi người sử dụng. Do tài nguyên hệ thống thường rất giới hạn và có khi không thể chia sẻ, nên hiếm khi tất cả các yêu cầu tài nguyên đồng thời đều được thỏa mãn. Vì thế cần phải nghiên cứu một phương pháp để chia sẻ một số tài nguyên hữu hạn giữa nhiều tiến trình người dùng đồng thời. Hệ điều hành quản lý nhiều loại tài nguyên khác nhau (CPU, bộ nhớ chính, các thiết bị ngoại vi …), với mỗi loại cần có một cơ chế cấp phát và các chiến lược cấp phát hiệu qủa. Mỗi tài nguyên được biểu diễn thông qua một cấu trúc dữ liệu, khác nhau về chi tiết cho từng loại tài nguyên, nhưng cơ bản chứa đựng các thông tin sau :

- Định danh tài nguyên

- Trạng thái tài nguyên : đây là các thông tin mô tả chi tiết trạng thái tài nguyên : phần nào của tài nguyên đã cấp phát cho tiến trình, phần nào còn có thể sử dụng ?

- Hàng đợi trên một tài nguyên : danh sách các tiến trình đang chờ được cấp phát tài nguyên tương ứng.

- Bộ cấp phát : là đoạn code đảm nhiệm việc cấp phát một tài nguyên đặc thù. Một số tài nguyên đòi hỏi các giải thuật đặc biệt (như CPU, bộ nhớ chính, hệ thống tập tin), trong khi những tài nguyên khác (như các thiết bị nhập/xuất) có thể cần các giải thuật cấp phát và giải phóng tổng quát hơn.

Hình 4.5 Khối quản lý tài nguyên

Các mục tiêu của kỹ thuật cấp phát :

- Bảo đảm một số lượng hợp lệ các tiến trình truy xuất đồng thời đến các tài nguyên không chia sẻ được.
- Cấp phát tài nguyên cho tiến trình có yêu cầu trong một khoảng thời gian trì hoãn có thể chấp nhận được.
- Tối ưu hóa sự sử dụng tài nguyên.

Để có thể thõa mãn các mục tiêu kể trên, cần phải giải quyết các vấn đề nảy sinh khi có nhiều tiến trình đồng thời yêu cầu một tài nguyên không thể chia sẻ.

2. Điều phối quá trình

Mục tiêu:

- *Nắm nguyên lý điều phối các quá trình được thực hiện trên CPU, tối ưu hóa sử dụng tài nguyên CPU.*

Trong môi trường hệ điều hành đa nhiệm, bộ phận điều phối tiến trình có nhiệm vụ xem xét và quyết định khi nào thì dừng tiến trình hiện tại để thu hồi processor và chuyển processor cho tiến trình khác, và khi đã có được processor thì chọn tiến trình nào trong số các tiến trình ở trạng thái ready để cấp processor cho nó. Ở đây chúng ta cần phân biệt sự khác nhau giữa điều độ tiến trình và điều phối tiến trình.

2.1. Giới thiệu

- **Các cơ chế điều phối tiến trình:** Trong công tác điều phối tiến trình bộ điều phối sử dụng hai cơ chế điều phối: Điều phối độc quyền và điều phối không độc quyền.

Điều phối độc quyền: Khi có được processor tiến trình toàn quyền sử dụng processor cho đến khi tiến trình kết thúc xử lý hoặc tiến trình tự động trả lại processor cho hệ thống. Các quyết định điều phối xảy ra khi: Tiến trình chuyển trạng thái từ Running sang Blocked hoặc khi tiến trình kết thúc.

Điều phối không độc quyền: Bộ phận điều phối tiến trình có thể tạm dừng tiến trình đang xử lý để thu hồi processor của nó, để cấp cho tiến trình khác, sao cho phù hợp với công tác điều phối hiện tại. Các quyết định điều phối xảy ra khi: Tiến trình chuyển trạng thái hoặc khi tiến trình kết thúc.

- **Các đặc điểm của tiến trình:** Khi tổ chức điều phối tiến trình, bộ phần điều phối tiến trình của hệ điều hành thường dựa vào các đặc điểm của tiến trình. Sau đây là một số đặc điểm của tiến trình:

Tiến trình thiên hướng Vào/Ra: Là các tiến trình cần nhiều thời gian hơn cho việc thực hiện các thao tác xuất/nhập dữ liệu, so với thời gian mà tiến trình cần để thực hiện các chỉ thị trong nó, được gọi là các tiến trình thiên hướng Vào/Ra.

Tiến trình thiên hướng xử lý: Ngược lại với trên, đây là các tiến trình cần nhiều thời gian hơn cho việc thực hiện các chỉ thị trong nó, so với thời gian mà tiến trình để thực hiện các thao tác Vào/Ra.

Tiến trình tương tác hay xử lý theo lô: Tiến trình cần phải trả lại kết quả tức thời (như trong hệ điều hành tương tác) hay kết thúc xử lý mới trả về kết quả (như trong hệ điều hành xử lý theo lô).

Độ ưu tiên của tiến trình: Mỗi tiến trình được gán một độ ưu tiên nhất định, độ ưu tiên của tiến trình có thể được phát sinh tự động bởi hệ thống hoặc được gán tường minh trong chương trình của người sử dụng. Độ ưu tiên của tiến trình có hai loại: Thứ nhất, độ ưu tiên tĩnh: là độ ưu tiên gán trước cho tiến trình và không thay đổi trong suốt thời gian sống của tiến trình. Thứ hai, độ ưu tiên động: là độ ưu tiên được gán cho tiến trình trong quá trình hoạt động của nó, hệ điều hành sẽ gán lại độ ưu tiên cho tiến trình khi môi trường xử lý của tiến trình bị thay đổi. Khi môi trường xử lý của tiến trình bị thay đổi hệ điều hành phải thay đổi độ ưu tiên của tiến trình cho phù hợp với tình trạng hiện tại của hệ thống và công tác điều phối tiến trình của hệ điều hành.

Thời gian sử dụng processor của tiến trình: Tiến trình cần bao nhiêu khoảng thời gian của processor để hoàn thành xử lý.

Thời gian còn lại tiến trình cần processor: Tiến trình còn cần bao nhiêu khoảng thời gian của processor nữa để hoàn thành xử lý.

Bộ phận điều phối tiến trình thường dựa vào đặc điểm của tiến trình để thực hiện điều phối ở mức tác tụ, hay điều phối tác vụ. Điều phối tác vụ được phải thực hiện trước điều phối tiến trình. Ở mức này hệ điều hành thực hiện việc chọn tác vụ để đưa vào hệ thống. Khi có một tiến trình được tạo lập hoặc khi có một tiến trình kết thúc xử lý thì bộ phận điều phối tác vụ được kích hoạt. Điều phối tác vụ quyết định sự đa chương của hệ thống và hiệu quả cũng như mục tiêu của điều phối của bộ phận điều phối tiến trình. Ví dụ, để khi thác tối đa thời gian xử lý của processor thì bộ phận điều phối tác vụ phải đưa vào hệ thống số lượng các tiến trình tính hướng Vào/Ra cân đối với số lượng các tiến trình tính hướng xử lý, các tiến trình này thuộc những tác vụ nào. Nếu trong hệ thống có quá nhiều tiến trình tính hướng Vào/Ra thì sẽ lãng phí thời gian xử lý của processor. Nếu trong hệ thống có quá nhiều tiến trình tính hướng xử lý thì processor không thể đáp ứng và có thể các tiến trình phải đợi lâu trong hệ thống, dẫn đến hiệu quả tương tác sẽ thấp.

- **Mục tiêu điều phối:** bộ phận điều phối tiến trình của hệ điều hành phải đạt được các mục tiêu sau đây trong công tác điều phối của nó.

Sự công bằng (Fairness): Các tiến trình đều công bằng với nhau trong việc chia sẻ thời gian xử lý của processor, không có tiến trình nào phải chờ đợi vô hạn để được cấp processor.

Tính hiệu quả (Efficiency): Tận dụng được 100% thời gian xử lý của processor. Trong công tác điều phối, khi processor rỗi bộ phận điều phối sẽ chuyển ngay nó cho tiến trình khác, nếu trong hệ thống có tiến trình đang ở trạng thái chờ processor, nên mục tiêu này dễ đạt được. Tuy nhiên, nếu hệ điều hành đưa vào hệ thống quá nhiều tiến trình thiên hướng vào/ra, thì nguy cơ processor bị rỗi là có thể. Do đó, để đạt được mục tiêu này hệ điều hành phải tính toán và quyết định nên đưa vào hệ thống bao nhiêu tiến trình thiên hướng vào/ra, bao nhiêu tiến trình thiên hướng xử lý, là thích hợp.

Thời gian đáp ứng hợp lý (Response time): Đối với các tiến trình tương tác, đây là khoảng thời gian từ khi tiến trình đưa ra yêu cầu cho đến khi nhận được sự hồi đáp. Một tiến trình đáp ứng yêu cầu của người sử dụng, phải nhận được thông tin hồi đáp từ yêu cầu của nó thì nó mới có thể trả lời người sử dụng. Do đó, theo người sử dụng thì bộ phận điều phối phải cực tiểu hoá thời gian hồi đáp của các tiến trình, có như vậy thì tính tương tác của tiến trình mới tăng lên.

Thời gian lưu lại trong hệ thống (Turnaround time): Đây là khoảng thời gian từ khi tiến trình được đưa ra đến khi được hoàn thành.

Bao gồm thời gian thực hiện thực tế cộng với thời gian đợi tài nguyên (bao gồm cả đợi processor). Đại lượng này dùng trong các hệ điều hành xử lý theo lô. Do đó, bộ phận điều phối phải cực tiểu thời gian hoàn thành (lưu lại trong hệ thống) của các tác vụ xử lý theo lô.

Thông lượng tối đa (Throughtput): Chính sách điều phối phải cố gắng để cực đại được số lượng tiến trình hoàn thành trên một đơn vị thời gian. Mục tiêu này ít phụ thuộc vào chính sách điều phối mà phụ thuộc nhiều vào thời gian thực hiện trung bình của các tiến trình.

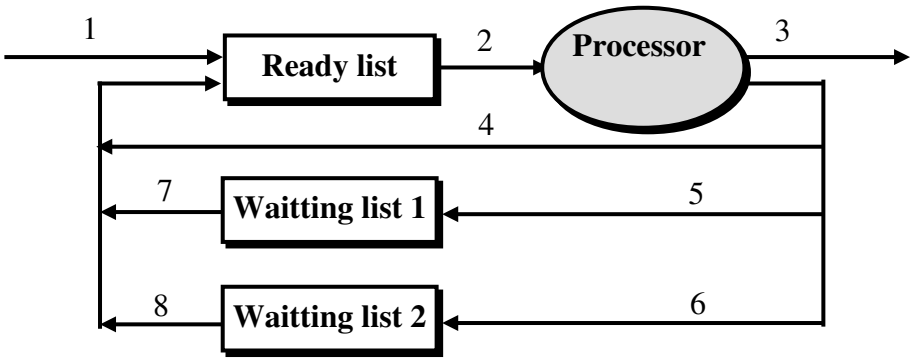
Công tác điều phối của hệ điều hành khó có thể thỏa mãn đồng thời tất cả các mục tiêu trên vì bản thân các mục tiêu này đã có sự mâu thuẫn với nhau. Các hệ điều hành chỉ có thể dung hòa các mục tiêu này ở một mức độ nào đó.

Ví dụ: Giả sử trong hệ thống có bốn tiến trình P_1, P_2, P_3, P_4, thời gian (t) mà các tiến trình này cần processor để xử lý lần lượt là 1, 12, 2, 1. Nếu ban đầu có 2 tiến trình P_1 và P_2 ở trạng thái ready thì chắc chắn bộ phận điều phối sẽ cấp processor cho P_1. Sau khi P_1 kết thúc thì processor sẽ được cấp cho P_2 để P_2 hoạt động (running), khi P_2 thực hiện được 2t thì P_3 được đưa vào trạng thái ready. Nếu để P_2 tiếp tục thì P_3 phải chờ lâu (chờ 8t), như vậy sẽ vi phạm mục tiêu thời gian hồi đáp và thông lượng tối đa (đối với P_3). Nếu cho P_2 dừng để cấp processor cho P_3 hoạt động đến khi kết thúc, khi đó thì P_4 vào trạng thái ready, bộ điều phối sẽ cấp processor cho P_4, và cứ như thế, thì P_2 phải chờ lâu, như vậy sẽ đạt được mục tiêu: thời gian hồi đáp và thông lượng tối đa nhưng vi phạm mục tiêu: công bằng và thời gian lưu lại trong hệ thống (đối với P_2).

2.2. Tổ chức điều phối

Để tổ chức điều phối tiến trình hệ điều hành sử dụng hai danh sách: Danh sách sẵn sàng (Ready list) dùng để chứa các tiến trình ở trạng thái sẵn sàng. Danh sách đợi (Waiting list) dùng để chứa các tiến trình đang đợi để được bổ sung vào danh sách sẵn sàng.

Chỉ có những tiến trình trong ready list mới được chọn để cấp processor. Các tiến trình bị chuyển về trạng thái blocked sẽ được bổ sung vào waiting list. Hệ thống chỉ có duy nhất một ready list, nhưng có thể tồn tại nhiều waiting list. Thông thường hệ điều hành thiết kế nhiều waitting list, mỗi waitting list dùng để chứa các tiến trình đang đợi được cấp phát một tài nguyên hay một sự kiện riêng biệt nào đó. Hình sau đây minh hoạ cho việc chuyển tiến trình giữa các danh sách:



Hình 4.6: Sơ đồ chuyển tiến trình vào các danh sách

Trong đó:

1. Tiến trình trong hệ thống được cấp đầy đủ tài nguyên chỉ thiếu processor.
2. Tiến trình được bộ điều phối chọn ra để cấp processor để bắt đầu xử lý.
3. Tiến trình kết thúc xử lý và trả lại processor cho hệ điều hành.
4. Tiến trình hết thời gian được quyền sử dụng processor (time-out), bị bộ điều phối tiến trình thu hồi lại processor.
5. Tiến trình bị khóa (blocked) do yêu cầu tài nguyên nhưng chưa được hệ điều hành cấp phát. Khi đó tiến trình được đưa vào danh sách các tiến trình đợi tài nguyên (waiting list 1).
6. Tiến trình bị khóa (blocked) do đang đợi một sự kiện nào đó xảy ra. Khi đó tiến trình được bộ điều phối đưa vào danh sách các tiến trình đợi tài nguyên (waiting list 2).
7. Tài nguyên mà tiến trình yêu cầu đã được hệ điều hành cấp phát. Khi đó tiến trình được bộ điều phối chuyển sang danh sách các tiến trình ở trạng thái sẵn sang (ready list) để chờ được cấp processor để được hoạt động.
8. Sự kiện mà tiến trình chờ đã xảy ra. Khi đó tiến trình được bộ điều phối chuyển sang danh sách các tiến trình ở trạng thái sẵn sang (ready list) để chờ được cấp processor.

2.3. Các chiến lược điều phối

Chiến lược FIFO (First In First Out): trong chiến lược này, khi processor rỗi thì hệ điều hành sẽ cấp nó cho tiến trình đầu tiên trong ready list, đây là tiến trình được chuyển sang trạng thái ready sớm nhất, có thể là tiến trình được đưa vào hệ thống sớm nhất. FIFO được sử dụng trong điều phối độc quyền nên khi tiến trình được cấp processor nó sẽ sở hữu processor cho đến khi kết thúc xử lý hay phải đợi một thao tác vào/ra hoàn thành, khi đó tiến trình chủ động trả lại processor cho hệ thống.

Ví dụ: Nếu hệ điều hành cần cấp processor cho 3 tiến trình P1, P2, P3,

với thời điểm vào ready list và khoảng thời gian mỗi tiến trình cần processor được mô tả trong bảng sau:

Tiến trình	thời điểm vào	t/g xử lý
P_1	0	24
P_2	1	3
P_3	2	3

Thì thứ tự cấp processor cho các tiến trình diễn ra như sau:

Tiến trình:	P_1	P_2	P_3
Thời điểm:	0	24	27

Vậy thời gian chờ của tiến trình P_1 là 0, của P_2 là 23 (24 - 0), của P_3 là 25 (24 + 3 - 2). Và thời gian chờ đợi trung bình của các tiến trình là:

$$(0 + 23 + 25)/3 = 16.$$

Như vậy FIFO tồn tại một số hạn chế: Thứ nhất, có thời gian chờ đợi trung bình lớn nên không phù hợp với các hệ thống chia sẻ thời gian. Thứ hai, khả năng tương tác kém khi nó được áp dụng trên các hệ thống uniprocessor. Thứ ba, nếu các tiến trình ở đầu ready list cần nhiều thời gian của processor thì các tiến trình ở cuối ready list sẽ phải chờ lâu mới được cấp processor.

Chiến lược phân phối xoay vòng (RR: Round Robin): trong chiến lược này, ready list được thiết kết theo dạng danh sách nối vòng. Tiến trình được bộ điều phối chọn để cấp processor cũng là tiến trình ở đầu ready list, nhưng sau một khoảng thời gian nhất định nào đó thì bộ điều phối lại thu hồi lại processor của tiến trình vừa được cấp processor và chuyển processor cho tiến trình kế tiếp (bây giờ đã trở thành tiến trình đầu tiên) trong ready list, tiến trình vừa bị thu hồi processor được đưa vào lại cuối ready list. Rõ ràng đây là chiến lược điều phối không độc quyền.

Khoảng khoản thời gian mà mỗi tiến trình được sở hữu processor để hoạt động là bằng nhau, và thường được gọi là Quantum.

Ví dụ: Nếu hệ điều hành cần cấp processor cho 3 tiến trình P_1, P_2, P_3 với thời điểm vào ready list và khoảng thời gian mỗi tiến trình cần processor được mô tả trong bảng sau:

Tiến trình	thời điểm vào	t/g xử lý
P_1	0	24
P_2	1	3
P_3	2	3

Quantum = 4

Thì thứ tự cấp processor cho các tiến trình lần lượt là:

Tiến trình	P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1
Thời điểm	0	4	7	10	14	18	22	26

Vậy thời gian chờ đợi trung bình sẽ là: (0 + 6 + 3 + 5)/3 = 4.46

Như vậy RR có thời gian chờ đợi trung bình nhỏ hơn so với FIFO

Trong chiến lược này, vấn đề đặt ra đối với công tác thiết kế là: nên chon quantum bằng bao nhiêu là thích hợp, nếu quantum nhỏ thì hệ thống phải tốn nhiều thời gian cho việc cập nhật ready list và chuyển trạng thái tiến trình, dẫn đến vi phạm mục tiêu: khai thác tối đa thời gian xử lý của processor. Nếu quantum lớn thì thời gian chờ đợi trung bình và thời gian hồi đáp sẽ tăng lên, dẫn đến tính tương tác của hệ thống bị giảm xuống.

Chiến lược theo độ ưu tiên: trong chiến lược này, bộ phận điều phối tiến trình dựa vào độ ưu tiên của các tiến trình để tổ chức cấp processor cho tiến trình. Tiến trình được chọn để cấp processor là tiến trình có độ ưu tiên cao nhất, tại thời điểm hiện tại.

Ở đây hệ điều hành thường tổ chức gán độ ưu tiên cho tiến trình theo nguyên tắc kết hợp giữ gán tĩnh và gán động. Khi khởi tạo tiến trình được gán độ ưu tiên tĩnh, sau đó phụ thuộc vào môi trường hoạt động của tiến trình và công tác điều phối tiến trình của bộ phận điều phối mà hệ điều hành có thể thay đổi độ ưu tiên của tiến trình.

Khi hệ thống phát sinh một tiến trình ready mới, thì bộ phận điều phối sẽ so sánh độ ưu tiên của tiến trình mới phát sinh với độ ưu tiên của tiến trình đang sở hữu processor (tạm gọi là tiến trình hiện tại). Nếu tiến trình mới có độ ưu tiên thấp hơn tiến trình hiện tại thì bộ phận điều phối sẽ chèn nó vào ready list tại vị trí thích hợp. Nếu tiến trình mới có độ ưu tiên cao hơn tiến trình hiện tại thì bộ điều phối sẽ thu hồi processor từ tiến trình hiện tại để cấp cho tiến trình mới yêu cầu, nếu là điều phối không độc quyền, hoặc chèn tiến trình mới vào ready list tại vị trí thích hợp, nếu là điều phối độc quyền.

Chiến lược này cũng phải sử dụng ready list, và ready list luôn được xếp theo thứ tự giảm dần của độ ưu tiên kể từ đầu danh sách. Điều này có nghĩa là tiến trình được chọn để cấp processor là tiến trình ở đầu ready list.

Ví dụ: Nếu hệ điều hành cần cấp processor cho 3 tiến trình P_1, P_2, P_3 với độ ưu tiên và khoảng thời gian mỗi tiến trình cần processor được mô tả trong bảng sau:

Tiến trình	độ ưu tiên	thời gian xử lý
P1	3	24
P2	1	3
P3	2	3

Thì thứ tự cấp processor (theo nguyên tắc độc quyền) cho các tiến trình lần lượt là:

Tiến trình	P2	P3	P1

Thời điểm 0 4 7

Chiến lược này có thể dẫn đến hậu quả: các tiến trình có độ ưu tiên thấp sẽ rơi vào tình trạng chờ đợi vô hạn. Để khắc phục điều này hệ điều hành thường hạ độ ưu tiên của các tiến trình có độ ưu tiên cao sau mỗi lần nó được cấp processor.

Chiến lược SJF (Shortest Job Fist: công việc ngắn nhất): Đây là trường hợp đặc biệt của chiến lược theo độ ưu tiên. Trong chiến lược này độ ưu tiên P của mỗi tiến trình là 1/t, với t là khoảng thời gian mà tiến trình cần processor. Bộ điều phối sẽ chọn tiến trình có P lớn để cấp processor, tức là ưu tiên cho những tiến trình có thời gian xử lý (thời gian cần processor) nhỏ.

Chiến lược này có thể có thời gian chờ đợi trung bình đạt cực tiểu. Nhưng hệ điều hành khó có thể đoán được thời gian xử lý mà tiến trình yêu cầu.

Chiến lược nhiều cấp độ ưu tiên: Hệ điều hành phân lớp các tiến trình theo độ ưu tiên của chúng để có cách thức điều phối thích hợp cho từng lớp tiến trình. Mỗi cấp độ ưu tiên có một realy list riêng. Bộ điều phối dùng chiến lược điều phối thích hợp cho từng realy list. Hệ điều hành cũng phải thiết kế một cơ chế thích hợp để điều phối tiến trình giữa các lớp.

Trong chiến lược này hệ điều hành sử dụng độ ưu tiên tĩnh, và điều phối không độc quyền, do đó một tiến trình thuộc ready list ở cấp ưu tiên i sẽ chỉ được cấp phát processor khi trong ready list ở cấp ưu tiên j (j > i) không còn một tiến trình nào.

Các tiến trình ở ready list có độ ưu tiên thấp sẽ phải chờ đợi processor trong một khoảng thời gian dài, có thể là vô hạn. Để khắc phục điều này hệ điều hành xây dựng chiến lược điều phối: Nhiều mức độ ưu tiên xoay vòng. Trong chiến lược này hệ điều hành chuyển dần một tiến trình ở ready list có độ ưu tiên cao xuống ready list có độ ưu tiên thấp hơn sau mỗi lần sử dụng procesor, và ngược lại một tiến trình ở lâu trong ready list có độ ưu tiên thấp thì sẽ được chuyển dần lên ready list có độ ưu tiên cao hơn.

Khi xây dựng chiến lược nhiều mức độ ưu tiên xoay vòng hệ điều hành cần xác định các thông tin sau: Số lượng các lớp ưu tiên. Chiến lược điều phối riêng cho từng read list trong mỗi lớp ưu tiên. Một tiến trình ready mới sẽ được đưa vào ready list nào. Khi nào thì thực hiện việc di chuyển một tiến trình từ ready list này sang ready list khác.

3. Bài toán đồng bộ hóa

Mục tiêu:

- *Năm được các giải pháp lập lịch mà hệ điều hành thực hiện nhằm điều phối các quá trình được thực hiện trên CPU.*

3.1 Giải pháp Busy-Waiting

3.1.1.Các giải pháp phần mềm

Sử dụng các biến cờ hiệu:

Tiếp cận : các tiến trình chia sẻ một biến chung đóng vai trò « chốt cửa » (lock) , biến này được khởi động là 0. Một tiến trình muốn vào miền găng trước tiên phải kiểm tra giá trị của biến lock. Nếu lock = 0, tiến trình đặt lại giá trị cho lock = 1 và đi vào miền găng. Nếu lock đang nhận giá trị 1, tiến trình phải chờ bên ngoài miền găng cho đến khi lock có giá trị 0. Như vậy giá trị 0 của lock mang ý nghĩa là không có tiến trình nào đang ở trong miền găng, và lock=1 khi có một tiến trình đang ở trong miền găng.

Cấu trúc một chương trình sử dụng biến khóa để đồng bộ

```
while (TRUE) {
while(lock==1);//wait
lock=1;
critical-section();
lock=0;
Noncritical-section ();
}
```

Thảo luận : Giải pháp này có thể vi phạm điều kiện thứ nhất: hai tiến trình có thể cùng ở trong miền găng tại một thời điểm. Giả sử một tiến trình nhận thấy lock = 0 và chuẩn bị vào miền găng, nhưng trước khi nó có thể đặt lại giá trị cho lock là 1, nó bị tạm dừng để một tiến trình khác hoạt động. Tiến trình thứ hai này thấy lock vẫn là 0 thì vào miền găng và đặt lại lock = 1. Sau đó tiến trình thứ nhất được tái kích hoạt, nó gán lock = 1 lần nữa rồi vaò miền găng. Như vậy tại thời điểm đó cả hai tiến trình đều ở trong miền găng.

Sử dụng việc kiểm tra luân phiên :

Tiếp cận : Đây là một giải pháp đề nghị cho hai tiến trình. Hai tiến trình này sử dụng chung biến *turn* (phản ánh phiên tiến trình nào được vào miền găng), được khởi động với giá trị 0. Nếu *turn = 0*, tiến trình A được vào miền găng. Nếu *turn = 1,* tiến trình A đi vào một vòng lặp chờ đến khi *turn* nhận giá trị *0*. Khi tiến trình A rời khỏi miền găng, nó đặt giá trị *turn* về 1 để cho phép tiến trình B đi vào miền găng.

Cấu trúc các tiến trình trong giải pháp kiểm tra luân phiên

```
while (TRUE) {
while(turn !=0);//wait
critical-section();
```

```
turn=1;
Noncritical-section ();
}
```
(a) Cấu trúc tiến trình A
```
while (TRUE) {
while(turn !=1);//wait
critical-section();
turn=0;
Noncritical-section ();
}
```
(b) Cấu trúc tiến trình B

Thảo luận: Giải pháp này dựa trên việc thực hiện sự kiểm tra nghiêm nhặt đến lượt tiến trình nào được vào miền găng. Do đó nó có thể ngăn chặn được tình trạng hai tiến trình cùng vào miền găng, nhưng lại có thể vi phạm điều kiện thứ ba: một tiến trình có thể bị ngăn chặn vào miền găng bởi một tiến trình khác không ở trong miền găng. Giả sử tiến trình B ra khỏi miền găng rất nhanh chóng. Cả hai tiến trình đều ở ngoài miền găng, và turn = 0. Tiến trình A vào miền găng và ra khỏi nhanh chóng, đặt lại giá trị của turn là1, rồi lại xử lý đoạn lệnh ngoài miền găng lần nữa. Sau đó, tiến trình A lại kết thúc nhanh chóng đoạn lệnh ngoài miền găng của nó và muốn vào miền găng một lần nữa. Tuy nhiên lúc này B vẫn còn mãi xử lý đoạn lệnh ngoài miền găng của mình, và turn lại mang giá trị 1 ! Như vậy, giải pháp này không có giá trị khi có sự khác biệt lớn về tốc độ thực hiện của hai tiến trình, nó vi phạm cả điều kiện thứ hai.

Giải pháp của Peterson

Tiếp cận : Petson đưa ra một giải pháp kết hợp ý tưởng của cả hai giải pháp kể trên. Các tiến trình chia sẻ hai biến chung :
```
int turn; // đến phiên ai
int interesse[2]; // khởi động là FALSE
```
Nếu *interesse[i]* = TRUE có nghĩa là tiến trình Pi muốn vào miền găng. Khởi đầu, *interesse[0]=interesse[1]=FALSE* và giá trị của est được khởi động là 0 hay 1. Để có thể vào được miền găng, trước tiên tiến trình Pi đặt giá trị *interesse[i]*=TRUE (xác định rằng tiến trình muốn vào miền găng), sau đó đặt turn=*j* (đề nghị thử tiến trình khác vào miền găng). Nếu tiến trình Pj không quan tâm đến việc vào miền găng (*interesse[j]=FALSE*), thì Pi có thể vào miền găng, nếu không, Pi phải chờ đến khi *interesse[j]=FALSE*. Khi tiến trình Pi rời khỏi miền găng, nó đặt lại giá trị cho *interesse[i]*= FALSE.

Cấu trúc tiến trình Pi trong giải pháp Peterson
```
while (TRUE) {
int j=1-i;//jlà tiến trình còn lại
```

```
interesse[i]= TRUE;
turn = j;
while (turn == j && interesse[j]==TRUE);
critical-section ();
interesse[i] = FALSE;
Noncritical-section ();
}
```

Thảo luận: giải pháp này ngăn chặn được tình trạng mâu thuẫn truy xuất : mỗi tiến trình Pi chỉ có thể vào miền găng khi *interesse[j]=FALSE* hoặc *turn = i*. Nếu cả hai tiến trình đều muốn vào miền găng thì *interesse[i] = interesse[j] =TRUE* nhưng giá trị của *turn* chỉ có thể hoặc là 0 hoặc là 1, do vậy chỉ có một tiến trình được vào miền găng.

3.1.2.Các giải pháp phần cứng

Cấm ngắt:

Tiếp cận: cho phép tiến trình cấm tất cả các ngắt trước khi vào miền găng, và phục hồi ngắt khi ra khỏi miền găng. Khi đó, ngắt đồng hồ cũng không xảy ra, do vậy hệ thống không thể tạm dừng hoạt động của tiến trình đang xử lý để cấp phát CPU cho tiến trình khác, nhờ đó tiến trình hiện hành yên tâm thao tác trên miền găng mà không sợ bị tiến trình nào khác tranh chấp.

Thảo luận: giải pháp này không được ưa chuộng vì rất thiếu thận trọng khi cho phép tiến trình người dùng được phép thực hiện lệnh cấm ngắt. Hơn nữa, nếu hệ thống có nhiều bộ xử lý, lệnh cấm ngắt chỉ có tác dụng trên bộ xử lý đang xử lý tiến trình, còn các tiến trình hoạt động trên các bộ xử lý khác vẫn có thể truy xuất đến miền găng !

Chỉ thị TSL (Test-and-Set):

Tiếp cận: đây là một giải pháp đòi hỏi sự trợ giúp của cơ chế phần cứng. Nhiều máy tính cung cấp một chỉ thị đặc biệt cho phép kiểm tra và cập nhật nội dung một vùng nhớ trong một thao tác không thể phân chia, gọi là chỉ thị *Test-and-Set Lock* (TSL) và được định nghĩa như sau:

```
Test-and-Setlock(boolean target)
{
Test-and-Setlock = target;
target = TRUE;
}
```

Nếu có hai chỉ thị TSL xử lý đồng thời (trên hai bộ xử lý khác nhau), chúng sẽ được xử lý tuần tự . Có thể cài đặt giải pháp truy xuất độc quyền với TSL bằng cách sử dụng thêm một biến lock, được khởi gán là FALSE. Tiến trình phải kiểm tra giá trị của biến lock trước khi vào miền găng, nếu lock = FALSE, tiến trình có thể vào miền găng.

Cấu trúc một chương trình trong giải pháp TSL

```
while (TRUE) {
while (Test-and-Setlock(lock));
critical-section ();
lock = FALSE;
Noncritical-section ();
}
```

Thảo luận : cũng giống như các giải pháp phần cứng khác, chỉ thị TSL giảm nhẹ công việc lập trình để giải quyết vấn để, nhưng lại không dễ dàng để cài đặt chỉ thị TSL sao cho được xử lý một cách không thể phân chia, nhất là trên máy với cấu hình nhiều bộ xử lý.

Tất cả các giải pháp trên đây đều phải thực hiện một vòng lặp để kiểm tra liệu nó có được phép vào miền găng, nếu điều kiện chưa cho phép, tiến trình phải chờ tiếp tục trong vòng lặp kiểm tra này. Các giải pháp buộc tiến trình phải liên tục kiểm tra điều kiện để phát hiện thời điểm thích hợp được vào miền găng như thế được gọi các giải pháp « *busy waiting* ». Lưu ý rằng việc kiểm tra như thế tiêu thụ rất nhiều thời gian sử dụng CPU, do vậy tiến trình đang chờ vẫn chiếm dụng CPU. Xu hướng giải quyết vấn đề đồng bộ hoá là nên tránh các giải pháp « *busy waiting*

3.2.Giải pháp Sleep and Wakeup

Để loại bỏ các bất tiện của giải pháp « busy waiting », chúng ta có thể tiếp cận theo hướng cho một tiến trình chưa đủ điều kiện vào miền găng chuyển sang trạng thái blocked, từ bỏ quyền sử dụng CPU. Để thực hiện điều này, cần phải sử dụng các thủ tục do hệ điều hành cung cấp để thay đổi trạng thái tiến trình. Hai thủ tục cơ bản *SLEEP* và *WAKEUP* thường được sử dụng để phục vụ mục đích này.

SLEEP là một lời gọi hệ thống có tác dụng tạm dừng hoạt động của tiến trình (blocked) gọi nó và chờ đến khi được một tiến trình khác « đánh thức ». Lời gọi hệ thống *WAKEUP* nhận một tham số duy nhất : tiến trình sẽ được tái kích hoạt (đặt về trạng thái ready).

Ý tưởng sử dụng SLEEP và WAKEUP như sau : khi một tiến trình chưa đủ điều kiện vào miền găng, nó gọi *SLEEP* để tự khóa đến khi có một tiến trình khác gọi *WAKEUP* để giải phóng cho nó. Một tiến trình gọi *WAKEUP* khi ra khỏi miền găng để đánh thức một tiến trình đang chờ, tạo cơ hội cho tiến trình này vào miền găng :

Cấu trúc chương trình trong giải pháp SLEEP and WAKEUP

```
int busy;// 1 nếu miền găng đang bị chiếm, nếu không
là 0
int blocked;// đếm số lượng tiến trình đang bị khóa
while (TRUE) {
if (busy){
blocked = blocked + 1;
```

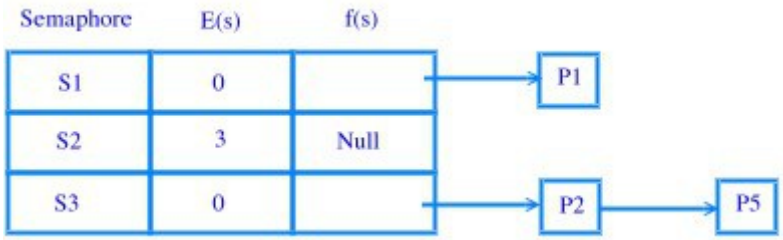
```
sleep();
}
else busy = 1;
critical-section ();
busy = 0;
if(blocked){
wakeup(process);
blocked = blocked - 1;
}
Noncritical-section ();
}
```

Khi sử dụng SLEEP và WAKEUP cần hết sức cẩn thận, nếu không muốn xảy ra tình trạng mâu thuẫn truy xuất trong một vài tình huống đặc biệt như sau : giả sử tiến trình A vào miền găng, và trước khi nó rời khỏi miền găng thì tiến trình B được kích hoạt. Tiến trình B thử vào miền găng nhưng nó nhận thấy A đang ở trong đó, do vậy B tăng giá trị biến *blocked* và chuẩn bị gọi *SLEEP* để tự khoá. Tuy nhiên trước khi B có thể thực hiện *SLEEP*, tiến trình A lại được tái kích hoạt và ra khỏi miền găng. Khi ra khỏi miền găng A nhận thấy có một tiến trình đang chờ (*blocked*=1) nên gọi *WAKEUP* và giảm giá trị của *blocked*. Khi đó tín hiệu *WAKEUP* sẽ lạc mất do tiến trình B chưa thật sự « ngủ » để nhận tín hiệu đánh thức !Khi tiến trình B được tiếp tục xử lý, nó mới goi *SLEEP* và tự khó vĩnh viễn !
Vấn đề ghi nhận được là tình trạng lỗi này xảy ra do việc kiểm tra tư cách vào miền găng và việc gọi SLEEP hay WAKEUP là những hành động tách biệ, có thể bị ngắt nửa chừng trong quá trình xử lý, do đó có khi tín hiệu WAKEUP gởi đến một tiến trình chưa bị khóa sẽ lạc mất.
Để tránh những tình huống tương tự, hệ điều hành cung cấp những cơ chế đồng bộ hóa dựa trên ý tưởng của chiến lược « SLEEP and WAKEUP » nhưng được xây dựng bao hàm cả phương tiện kiểm tra điều kiện vào miền găng giúp sử dụng an toàn.

3.2.1.Semaphore

Tiếp cận: Được **Dijkstra** đề xuất vào 1965, một semaphore ***s*** là một *biến* có các thuộc tính sau:
Một giá trị nguyên dương *e(s)*
Một hàng đợi *f(s)* lưu danh sách các tiến trình đang bị khóa (chờ) trên semaphore *s*
Chỉ có hai thao tác được định nghĩa trên semaphore
Down(s): giảm giá trị của semaphore *s* đi 1 đơn vị nếu semaphore có trị e(s) > 0, và tiếp tục xử lý. Ngược lại, nếu e(s) 0, tiến trình phải chờ đến khi e(s) >0.

Up(s): tăng giá trị của semaphore *s* lên 1 đơn vị. Nếu có một hoặc nhiều tiến trình đang chờ trên semaphore *s*, bị khóa bởi thao tác **Down**, thì hệ thống sẽ chọn một trong các tiến trình này để kết thúc thao tác **Down** và cho tiếp tục xử lý.



Hình 4.7 Semaphore

Cài đặt: Gọi *p* là tiến trình thực hiện thao tác *Down(s)* hay *Up(s).*

```
Down(s):
e(s) = e(s) - 1;
if e(s) < 0 {
status(P)= blocked;
enter(P,f(s));
}
Up(s):
e(s) = e(s) + 1;
if s   0 {
exit(Q,f(s)); //Q là tiến trình đang chờ trên s
status (Q) = ready;
enter(Q,ready-list);
}
```

Lưu ý cài đặt này có thể đưa đến một giá trị âm cho semaphore, khi đó trị tuyệt đối của semaphore cho biết số tiến trình đang chờ trên semaphore.

Điều quan trọng là các thao tác này cần thực hiện một cách không bị phân chia, không bị ngắt nữa chừng, có nghĩa là không một tiến trình nào được phép truy xuất đến semaphore nếu tiến trình đang thao tác trên semaphore này chưa kết thúc xử lý hay chuyển sang trạng thái blocked.

Sử dụng: có thể dùng semaphore để giải quyết vấn đề truy xuất độc quyền hay tổ chức phối hợp giữa các tiến trình.

Tổ chức truy xuất độc quyền với Semaphores: khái niệm *semaphore* cho phép bảo đảm nhiều tiến trình cùng truy xuất đến miền găng mà không có sự mâu thuẫn truy xuất. *n* tiến trình cùng sử dụng một semaphore s, e(s) được khởi gán là 1. Để thực hiện đồng bộ hóa, tất cả các tiến trình cần phải áp dụng cùng cấu trúc chương trình sau đây:

Cấu trúc một chương trình trong giải pháp semaphore

```
while (TRUE) {
Down(s)
```

```
critical-section ();
Up(s)
Noncritical-section ();
}
```

Tổ chức đồng bộ hóa với Semaphores: với semaphore có thể đồng bộ hóa hoạt động của hai tiến trình trong tình huống một tiến trình phải đợi một tiến trình khác hoàn tất thao tác nào đó mới có thể bắt đầu hay tiếp tục xử lý. Hai tiến trình chia sẻ một semaphore s, khởi gán e(s) là 0. Cả hai tiến trình có cấu trúc như sau:

Cấu trúc chương trình trong giải pháp semaphore

```
P1:
while (TRUE) {
job1();
Up(s); //đánh thức P2
}
P2:
while (TRUE) {
Down(s); // chờ P1
job2();
}
```

Thảo luận : Nhờ có thực hiện một các không thể phân chia, semaphore đã giải quyết được vấn đề tín hiệu "đánh thức" bị thất lạc. Tuy nhiên, nếu lập trình viên vô tình đặt các primitive Down và Up sai vị trí, thứ tự trong chương trình, thì tiến trình có thể bị khóa vĩnh viễn.

Ví dụ :

```
while (TRUE) {
Down(s)
critical-section ();
Noncritical-section ();
}
```

tiến trình trên đây quên gọi Up(s), và kết quả là khi ra khỏi miền găng nó sẽ không cho tiến trình khác vào miền găng !

Vì thế việc sử dụng đúng cách semaphore để đồng bộ hóa phụ thuộc hoàn toàn vào lập trình viên và đòi hỏi lập trình viên phải hết sức thận trọng.

3.2.2Monitors

Tiếp cận: Để có thể dễ viết đúng các chương trình đồng bộ hóa hơn, Hoare(1974) và Brinch & Hansen (1975) đã đề nghị một cơ chế cao hơn được cung cấp bởi ngôn ngữ lập trình , là *monitor*. Monitor là một cấu trúc đặc biệt bao gồm các thủ tục, các biến và cấu trúc dữ liệu có các thuộc tính sau :

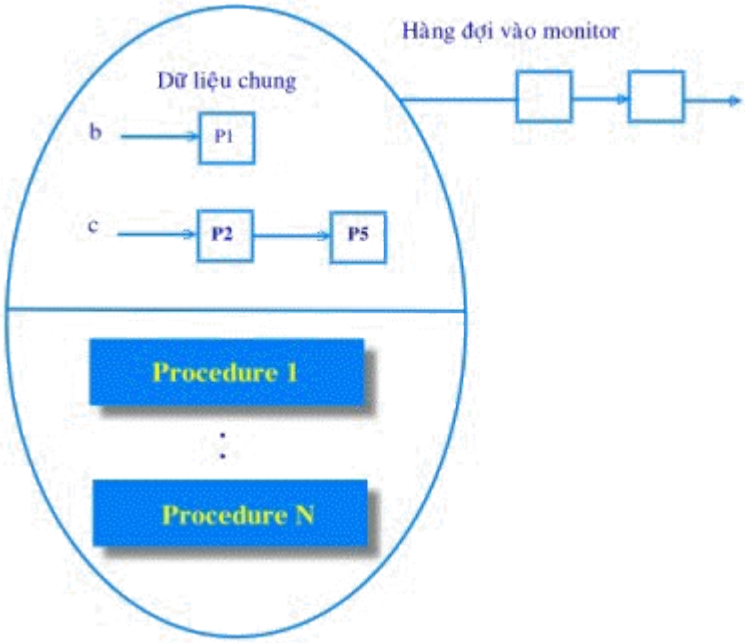
Các biến và cấu trúc dữ liệu bên trong monitor chỉ có thể được thao tác bởi các thủ tục định nghĩa bên trong monitor đó. (*encapsulation*).

Tại một thời điểm, chỉ có một tiến trình duy nhất được hoạt động bên trong một monitor (*mutual exclusive*).

Trong một monitor, có thể định nghĩa các *biến điều kiện* và hai thao tác kèm theo là **Wait** và **Signal** như sau : gọi ***c*** là biến điều kiện được định nghĩa trong monitor:

Wait(c): chuyển trạng thái tiến trình gọi sang blocked , và đặt tiến trình này vào hàng đợi trên biến điều kiện ***c***.

Signal(c): nếu có một tiến trình đang bị khóa trong hàng đợi của ***c***, tái kích hoạt tiến trình đó, và tiến trình gọi sẽ rời khỏi monitor.



Hình 4.8 Monitor và các biến điều kiện

Cài đặt : trình biên dịch chịu trách nhiệm thực hiện việc truy xuất độc quyền đến dữ liệu trong monitor. Để thực hiện điều này, một semaphore nhị phân thường được sử dụng. Mỗi monitor có một hàng đợi toàn cục lưu các tiến trình đang chờ được vào monitor, ngoài ra, mỗi biến điều kiện ***c*** cũng gắn với một hàng đợi ***f(c)*** và hai thao tác trên đó được định nghĩa như sau:

```
Wait(c) :
status(P)= blocked;
enter(P,f(c));
Signal(c) :
if (f(c) != NULL){
exit(Q,f(c)); //Q là tiến trình chờ trên c
statusQ) = ready;
enter(Q,ready-list);
}
```

Sử dụng: Với mỗi nhóm tài nguyên cần chia sẻ, có thể định nghĩa một monitor trong đó đặc tả tất cả các thao tác trên tài nguyên này với một số điều kiện nào đó.:

Cấu trúc một monitor

```
monitor <tên monitor >
condition <danh sách các biến điều kiện>;
<déclaration de variables>;
    procedure Action1();
  {

  }
          ....

  procedure Actionn();
  {
  }
 end monitor;
```

Các tiến trình muốn sử dụng tài nguyên chung này chỉ có thể thao tác thông qua các thủ tục bên trong monitor được gắn kết với tài nguyên:

Cấu trúc tiến trình Pi trong giải pháp monitor

```
while (TRUE) {
Noncritical-section ();
<monitor>.Actioni; //critical-section();
Noncritical-section ();
}
```

Thảo luận: Với monitor, việc truy xuất độc quyền được bảo đảm bởi trình biên dịch mà không do lập trình viên, do vậy nguy cơ thực hiện đồng bộ hóa sai giảm rất nhiều. Tuy nhiên giải pháp monitor đòi hỏi phải có một ngôn ngữ lập trình định nghĩa khái niệm monitor, và các ngôn ngữ như thế chưa có nhiều.

3.2.3.Trao đổi thông điệp

Tiếp cận: giải pháp này dựa trên cơ sở trao đổi thông điệp với hai primitive Send và Receive để thực hiện sự đồng bộ hóa:

Send(destination, message): gởi một thông điệp đến một tiến trình hay gởi vào hộp thư.

Receive(source,message): nhận một thông điệp thừ một tiến trình hay từ bất kỳ một tiến trình nào, tiến trình gọi sẽ chờ nếu không có thông điệp nào để nhận.

Sử dụng: Có nhiều cách thức để thực hiện việc truy xuất độc quyền bằng cơ chế trao đổi thông điệp. Đây là một mô hình đơn giản: một tiến trình kiểm soát việc sử dụng tài nguyên và nhiều tiến trình khác yêu cầu tài

nguyên này. Tiến trình có yêu cầu tài nguyên sẽ gởi một thông điệp đến tiến trình kiểm soát và sau đó chuyển sang trạng thái blocked cho đến khi nhận được một thông điệp chấp nhận cho truy xuất từ tiến trình kiểm soát tài nguyên.Khi sử dụng xong tài nguyên , tiến trình gởi một thông điệp khác đến tiến trình kiểm soát để báo kết thúc truy xuất. Về phần tiến trình kiểm soát , khi nhận được thông điệp yêu cầu tài nguyên, nó sẽ chờ đến khi tài nguyên sẵn sàng để cấp phát thì gởi một thông điệp đến tiến trình đang bị khóa trên tài nguyên đó để đánh thức tiến trình này.

Cấu trúc tiến trình yêu cầu tài nguyên trong giải pháp message

```
while (TRUE) {
Send(process controler, request message);
Receive(process controler, accept message);
critical-section ();
Send(process controler, end message);
Noncritical-section ();
}
```

Thảo luận: Các primitive semaphore và monitor có thể giải quyết được vấn đề truy xuất độc quyền trên các máy tính có một hoặc nhiều bộ xử lý chia sẻ một vùng nhớ chung. Nhưng các primitive không hữu dụng trong các hệ thống phân tán, khi mà mỗi bộ xử lý sỡ hữu một bộ nhớ riêng biệt và liên lạc thông qua mạng. Trong những hệ thống phân tán như thế, cơ chế trao đổi thông điệp tỏ ra hữu hiệu và được dùng để giải quyết bài toán đồng bộ hóa.

4.Bế tắc-Giải pháp phòng ngừa và xử lý

Mục tiêu

- Hiểu được các nguyên nhân gây bế tắc của hệ thống và cách phòng ngừa, xử lý bế tắc.

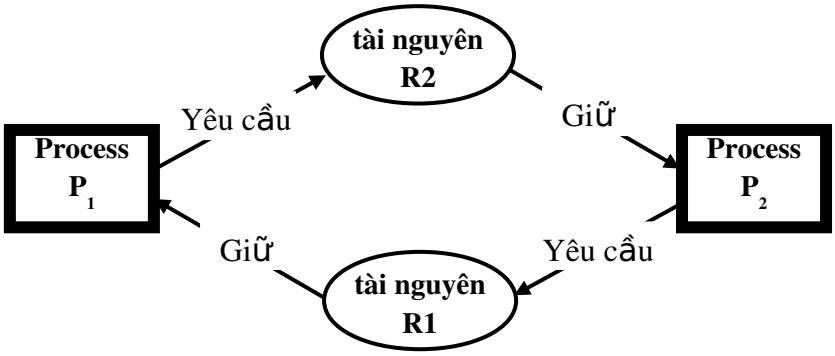
4.1.Bế tắc

Tất cả hiện tượng bế tắc đều bắt nguồn từ sự xung đột về tài nguyên của hai hoặc nhiều tiến trình đang hoạt động đồng thời trên hệ thống. Tài nguyên ở đây có thể là một ổ đĩa, một record trong cơ sở dữ liệu, hay một không gian địa chỉ trên bộ nhớ chính. Sau đây là một số ví dụ để minh hoạ cho điều trên.

Ví dụ 1: Giả sử có hai tiến trình P_1 và P_2 hoạt động đồng thời trong hệ thống. Tiến trình P_1 đang giữ tài nguyên R1 và xin được cấp R2 để tiếp tục hoạt động, trong khi đó tiến trình P_2 đang giữ tài nguyên R2 và xin được cấp R1 để tiếp tục hoạt động. Trong trường hợp này cả P_1 và P_2 sẽ không tiếp tục hoạt động được. Như vậy P_1 và P_2 rơi vào trạng thái tắc nghẽn. Ví dụ này có thể được minh hoạ bởi sơ đồ ở hình 2.

Bế tắc thường xảy ra do xung đột về tài nguyên thuộc loại không

phân chia được, một số ít trường hợp xảy ra với tài nguyên phân chia được. Ví dụ sau đây là trường hợp bế tắc do xung đột về tài nguyên bộ nhớ, là tài nguyên thuộc loại phân chia được.



Ví dụ 2: Giả sử không gian bộ nhớ còn trống là 200Kb, và trong hệ thống có hai tiến trình P_1 và P_2 hoạt động đồng thời. P_1 và P_2 yêu cầu được sử dụng bộ nhớ như sau:

P1	P2
….	….
Request1 80Kb	Request1 70Kb
…..	…..
Request2 30Kb	Request2 40Kb
…..	…..

Bế tắc xảy ra khi cả hai tiến trình cùng yêu cầu thêm bộ nhớ lần thứ hai. Tại thời điểm này không gian bộ nhớ còn trống là 50Kb, lớn hơn lượng bộ nhớ mà mỗi tiến trình yêu cầu (30Kb và 40Kb), nhưng vì cả hai tiến trình đồng thời yêu cầu thêm bộ nhớ, nên hệ thống không để đáp ứng được, và bế tắc xảy ra.

Ví dụ 3: Trong các ứng dụng cơ sở dữ liệu, một chương trình có thể khoá một vài record mà nó sử dụng, để dành quyền điều khiển về cho nó. Nếu tiến trình P_1 khoá record R1, tiến trình P_2 khoá record R2, và rồi sau đó mỗi tiến trình lại cố gắng khoá record của một tiến trình khác. bế tắc sẽ xảy ra.

Như vậy bế tắc là hiện tượng: Trong hệ thống xuất hiện một tập các tiến trình, mà mỗi tiến trình trong tập này đều chờ được cấp tài nguyên, mà tài nguyên đó đang được một tiến trình trong tập này chiếm giữ. Và sự đợi này có thể kéo dài vô hạn nếu không có sự tác động từ bên ngoài.

Trong trường hợp của ví dụ 1 ở trên: hai tiến trình P_1 và P_2 sẽ rơi vào trạng thái bế tắc, nếu không có sự can thiệp của hệ điều hành. Để phá bỏ bế tắc này hệ điều hành có thể cho tạm dừng tiến trình P_1 để thu hồi lại tài

nguyên R1, lấy R1 cấp cho tiến trình P_2 để P_2 hoạt động và kết thúc, sau đó thu hồi cả R1 và R2 từ tiến trình P_2 để cấp cho P_1 và tái kích hoạt P_1 để P_1 hoạt động trở lại. Như vậy sau một khoảng thời gian cả P_1 và P_2 đều ra khỏi tình trạng bế tắc.

Trong trường hợp của ví dụ 2 ở trên: nếu hai tiến trình này không đồng

thời yêu cầu thêm bộ nhớ thì bế tắc không thể xảy ra, hoặc khi cả hai tiến trình đồng thời yêu cầu thêm bộ nhớ thì hệ điều hành phải kiểm tra lượng bộ nhớ còn trống của hệ thống, nếu không đáp ứng cho cả hai tiến trình thì hệ điều hành phải có cơ chế ngăn chặn (từ chối) một tiến trình và chỉ cho một tiến trình được quyền sử dụng bộ nhớ (đáp ứng) thì bế tắc cũng không thể xảy ra. Tuy nhiên để giải quyết vấn đề bế tắc do thiếu bộ nhớ, các hệ điều hành thường sử dụng cơ chế bộ nhớ ảo. Bộ nhớ ảo là một phần quan trong của hệ điều hành mà chúng ta sẽ khảo sát ở chương *Quản lý bộ nhớ* của tài liệu này.

Khi hệ thống xảy ra bế tắc nếu hệ điều hành không kịp thời phá bế tắc thì hệ thống có thể rơi vào tình trạng treo toàn bộ hệ thống. Như trong trường hợp bế tắc ở ví dụ 1, nếu sau đó có tiến trình P_3, đang giữ tài nguyên R3, cần R2 để tiếp tục thì P_3 cũng sẽ rơi vào tập tiến trình bị bế tắc, rồi sau đó nếu có tiến trình P_4 cần tài nguyên R1 và R3 để tiếp tục thì P_4 cũng rơi vào tập các tiến trình bị bế tắc như P_3, … cứ thế dần dần có thể dẫn đến một thời điểm tất cả các tiến trình trong hệ thống đều rơi vào tập tiến trình bế tắc. Và như vậy hệ thống sẽ bị treo hoàn toàn.

4.2.Điều kiện hình thành bế tắc

Năm 1971, Coffman đã đưa ra và chứng tỏ được rằng, nếu hệ thống tồn tại đồng thời bốn điều kiện sau đây thì hệ thống sẽ xảy ra bế tắc:

1.Loại trừ lẫn nhau (mutual excution) hay độc quyền sử dụng: Đối với các tài nguyên không phân chia được thì tại mỗi thời điểm chỉ có một tiến trình sử dụng được tài nguyên.

2.Giữ và đợi (hold and wait): Một tiến trình hiện tại đang chiếm giữ tài nguyên, lại xin cấp phát thêm tài nguyên mới.

3.Không ưu tiên (No preemption): Không có tài nguyên nào có thể được giải phóng từ một tiến trình đang chiếm giữ nó.

Trong nhiều trường hợp các điều kiện trên là rất cần thiết đối với hệ thống. Sự thực hiện độc quyền là cần thiết để bảo đảm tính đúng đắn của kết quả và tính toàn vẹn của dữ liệu (chúng ta đã thấy điều này ở phần tài nguyên găng trên đây). Tương tự, sự ưu tiên không thể thực hiện một cách tuỳ tiện, đặc biệt đối với các tài nguyên có liên quan với nhau, việc giải phóng từ một tiến trình này có thể ảnh hưởng đên kết quả xử lý của các tiến trình khác.

Sự bế tắc có thể tồn tại với ba điều kiện trên, nhưng cũng có thể không xảy ra chỉ với 3 điều kiện đó. Để chắc chắn bế tắc xảy ra cần phải có điều kiện thứ tư

3.Đợi vòng tròn (Circular wait): Đây là trường hợp của ví dụ 1 mà chúng ta đã nêu ở trên. Tức là, mỗi tiến trình đang chiếm giữ tài nguyên mà tiến trình khác đang cần.

Ba điều kiện đầu là điều kiện cần chứ không phải là điều kiện đủ để xảy ra bế tắc. Điều kiện thứ tư là kết quả tất yếu từ ba điều kiện đầu.

Ngăn chặn bế tắc

Ngăn chặn bế tắc là thiết kế một hệ thống sao cho hiện tượng bế tắc bị loại trừ. Các phương thức ngăn chặn bế tắc đều tập trung giải quyết bốn điều kiện gây ra bế tắc, sao cho hệ thống không thể xảy ra đồng thời bốn điều kiện bế tắc:

Đối với điều kiện độc quyền: Điều kiện này gần như không tránh khỏi, vì sự độc quyền là cần thiết đối với tài nguyên thuộc loại phân chia được như các biến chung, các tập tin chia sẻ, hệ điều hành cần phải hỗ trợ sự độc quyền trên các tài nguyên này. Tuy nhiên, với những tài nguyên thuộc loại không phân chia được hệ điều hành có thể sử dụng kỹ thuật SPOOL (Smulataneous Peripheral Operation Online) để tạo ra nhiều tài nguyên ảo cung cấp cho các tiến trình đồng thời.

Đối với điều kiện giữ và đợi: Điều kiện này có thể ngăn chặn bằng cách yêu cầu tiến trình yêu cầu tất cả tài nguyên mà nó cần tại một thời điểm và tiến trình sẽ bị khoá (blocked) cho đến khi yêu cầu tài nguyên của nó được hệ điều hành đáp ứng. Phương pháp này không hiệu quả. Thứ nhất, tiến trình phải đợi trong một khoảng thời gian dài để có đủ tài nguyên mới có thẻ chuyển sang hoạt động được, trong khi tiến trình chỉ cần một số ít tài nguyên trong số đó là có thể hoạt động được, sau đó yêu cầu tiếp. Thứ hai, lãng phí tài nguyên, vì có thể tiến trình giữa nhiều tài nguyên mà chỉ đến khi sắp kết thúc tiến trình mới sử dụng, và có thể đây là những tài nguyên mà các tiến trình khác đang rất cần. Ở đây hệ điều hành có thể tổ chức phân lớp tài nguyên hệ thống. Theo đó tiến trình phải trả tài nguyên ở mức thấp mới được cấp phát tài nguyên ở cấp cao hơn.

Đối với điều kiện No preemption: Điều kiện này có thể ngăn chặn bằng cách, khi tiến trình bị rơi vào trạng thái khoá, hệ điều hành có thể thu hồi tài nguyên của tiến trình bị khoá để cấp phát cho tiến trình khác và cấp lại đầy đủ tài nguyên cho tiến trình khi tiến trình được đưa ra khỏi trạng thái khoá.

Đối với điều kiện chờ đợi vòng tròn: Điều kiện này có thể ngăn chặn bằng cách phân lớp tài nguyên của hệ thống. Theo đó, nếu một tiến trình được cấp phát tài nguyên ở lớp L, thì sau đó nó chỉ có thể yêu cầu các

tài nguyên ở lớp thấp hơn lớp L.

4.3.Xử lý bế tắc

Các phương thức ngăn chặn bế tắc ở trên đều tập trung vào việc hạn chế quyền truy xuất đến tài nguyên và áp đặt các ràng buộc lên các tiến trình. Điều này có thể ảnh hưởng đến mục tiêu khai thác hiệu quả tài nguyên của hệ điều hành, ngăn chặn độc quyền trên tài nguyên là một ví dụ, hệ điều hành phải cài đặt các cơ chế độc quyền để bảo vệc các tài nguyên chia sẻ. Và như đã phân tích ở trên việc cấp phát tài nguyên một lần cho các tiến trình để ngăn chặn hiện tượng hold and wait cũng tồn tại một vài hạn chế.

Các hệ điều hành có thể giải quyết vấn đề bế tắc theo hướng phát hiện bế tắc để tìm cách thoát khỏi bế tắc. Phát hiện bế tắc không giới hạn truy xuất tài nguyên và không áp đặt các ràng buộc lên tiến trình. Với phương thức phát hiện bế tắc, các yêu cầu cấp phát tài nguyên được đáp ứng ngay nếu có thể. Để phát hiện bế tắc hệ điều hành thường cài đặt một thuật toán để phát hiện hệ thống có tồn tại hiện tượng chờ đợi vòng tròn hay không.

Việc kiểm tra, để xem thử hệ thống có khả năng xảy ra bế tắc hay không có thể được thực hiện liên tục mỗi khi có một yêu cầu tài nguyên, hoặc chỉ thực hiện thỉnh thoảng theo chu kỳ, phụ thuộc vào sự bế tắc xảy ra như thế nào. Việc kiểm tra bế tắc mỗi khi có yêu cầu tài nguyên sẽ nhận biết được khả năng xảy ra bế tắc nhanh hơn, thuật toán được áp dụng đơn giản hơn vì chỉ dự vào sự thay đổi trạng thái của hệ thống. Tuy nhiên, hệ thống phải tốn nhiều thời gian cho mỗi lần kiểm tra bế tắc.

Mỗi khi bế tắc được phát hiện, hệ điều hành thực hiện một vài giải pháp để thoát khỏi bế tắc. Sau đây là một vài giải pháp có thể:

1.Thoát tất cả các tiến trình bị bế tắc. Đây là một giải pháp đơn giản nhất, thường được các hệ điều hành sử dụng nhất.

2.Sao lưu lại mỗi tiến trình bị bế tắc tại một vài điểm kiển tra được định nghĩa trước, sau đó khởi động lại tất cả các tiến trình. Giải pháp này yêu cầu hệ điều hành phải lưu lại các thông tin cần thiết tại điểm dừng của tiến trình, đặc biệt là con trỏ lệnh và các tài nguyên tiến trình đang sử dụng, để có thể khởi động lại tiến trình được. Giải pháp này có nguy cơ xuất hiện bế tắc trở lại là rất cao, vì khi tất cả các tiến trình đều được reset trở lại thì việc tranh chấp tài nguyên là khó tránh khỏi. Ngoài ra hệ điều hành thường phải chi phí rất cao cho việc tạm dừng và tái kích hoạt tiến trình.

3.Chỉ kết thúc một tiến trình trong tập tiến trình bị bế tắc, thu hồi tài nguyên của tiến trình này, để cấp phát cho một tiến trình nào đó trong tập tiến trình bế tắc để giúp tiến trình này ra khỏi bế tắc, rồi gọi lại thuật

toán kiểm tra bế tắc để xem hệ thống đã ra khỏi bế tắc hay chưa, nếu rồi thì dừng, nếu chưa thì tiếp tục giải phóng thêm tiến trình khác. Và lần lượt như thế cho đến khi tất cả các tiến trình trong tập tiến trình bế tắc đều ra khỏi tình trạng bế tắc. Trong giả pháp này vấn đề đặt ra đối với hệ điều hành là nên chọn tiến trình nào để giải phóng đầu tiên và dựa vào tiêu chuẩn nào để chọn lựa sao cho chi phí để giải phóng bế tắc là thấp nhất.
4.Tập trung toàn bộ quyền ưu tiên sử dụng tài nguyên cho một tiến trình, để tiến trình này ra khỏi bế tắc, và rồi kiểm tra xem hệ thống đã ra khỏi bế tắc hay chưa, nếu rồi thì dừng lại, nếu chưa thì tiếp tục. Lần lượt như thế cho đến khi hệ thống ra khỏi bế tắc. Trong giải pháp này hệ điều hành phải tính đến chuyện tái kích hoạt lại tiến trình sau khi hệ thống ra khỏi bế tắc.

Đối với các giải pháp 3 và 4, hệ điều hành dựa vào các tiêu chuẩn sau đây để chọn lựa tiến trình giải phóng hay ưu tiên tài nguyên: Thời gian xử lý ít nhất; Thời gian cần processor còn lại ít nhất; Tài nguyên cần cấp phát là ít nhất; Quyền ưu tiên là thấp nhất.

CÂU HỎI CỦNG CỐ BÀI HỌC

1. Tổ chức điều phối tiến trình ?
2. Phân tích ưu, khuyết điểm của các chiến lược điều phối
3. Bài toán đồng bộ hóa là gì?
4. Bế tắc và giải pháp phòng ngừa?
5.Giả sử có các quá trình sau trong hệ thống :

Quá trình	Thời điểm vào RL	Thời gian CPU
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

Sử dụng nguyên tắc điều phối độc quyền và các thông tin có được tại thời điểm ra quyết định để trả lời các câu hỏi sau đây :

a)Cho biết thời gian lưu lại trung bình trong hệ thống (turnaround time) của các quá trình trong thuật toán điều phối FIFO.

b)Cho biết thời gian lưu lại trung bình trong hệ thống (turnaround time) của các quá trình trong thuật toán điều phối SJF.

c)Thuật toán SJF dự định cải tiến sự thực hiện của hệ thống , nhưng lưu ý chúng ta phải chọn điều phối P_1 tại thời điểm 0 vì không biết rằng sẽ có hai quá trình ngắn hơn vào hệ thống sau đó . Thử tính thời gian lưu lại trung bình trong ệ thống nếu để CPU nhàn rỗi trong 1 đơn vị thời gian đầu tiên và sau đó sử dụng SJF để điều phối. Lưu ý P_1 và P_2 sẽ phải chờ trong suốt thời gian nhàn rỗi này, do vậy thời gian chờ của chúng tăng lên. Thuật toán điều phối này được biết đến như điều phối dựa trên thông tin về tương lai.

6.Phân biệt sự khác nhau trong cách tiếp cận để ưu tiên cho quá trình ngắn trong các thuật toán điều phối sau :

a) FIFO.

b)RR

c)Điều phối với độ ưu tiên đa cấp

CHƯƠNG 4:
HỆ ĐIỀU HÀNH ĐA XỬ LÝ

Mã chương: MH15-05

Giới thiệu:

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 chữ 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 trung gian.
- Mặt 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 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 tùy ý.

Trong chương này, chúng ta tập trung xét chủ yếu về hai hệ thống trên.

Mục Tiêu:

- Hiểu khái quát được xu thế sử dụng hệ thống đa xử lý hiện nay.

- Hiểu được những nét cơ bản về hệ điều hành đa xử lý nhằm trang bị khả năng tự nghiên cứu trong tương lai.
- Rèn luyện khả năng tư duy, lập luận có tính khoa học
- Tinh thần hỗ trợ nhau trong học tập.

Nội Dung:

1.Hệ điều hành đa xử lý tập trung

Mục tiêu:

- Hiểu khái quát được xu thế sử dụng hệ thống đa xử lý hiện nay.

- Hiểu được những nét cơ bản về hệ điều hành đa xử lý nhằm trang bị khả năng tự nghiên cứu trong tương lai.

1.1.Hệ thống đa xử lý

Hệ thống nhiều CPU

Hiện nay, từ tốc độ phát triển nhanh của công nghệ, máy tính ngày càng được phổ dụng trong xã hội. Mức độ thâm nhập của máy tính vào cuộc sống càng cao thì yêu cầu nâng cao năng lực của máy tính lại ngày càng trở nên cấp thiết. Bộ nhớ chính ngày càng rộng lớn; đĩa từ có dung lượng càng rộng, tốc độ truy nhập ngày càng cao; hệ thống thiết bị ngoại vi càng phong phú, hình thức giao tiếp người-máy ngày càng đa dạng. Như đã nói, CPU là một tài nguyên thể hiện chủ yếu nhất năng lực của hệ thống máy tính, vì vậy một trong những vấn đề trọng tâm nhất để tăng cường năng lực của hệ thống là tăng cường năng lực của CPU. Đối với vấn đề này, nảy sinh các giải pháp theo hai hướng:

Giải pháp tăng cường năng lực của một CPU riêng cho từng máy tính: công nghệ vi mạch ngày càng phát triển vì vậy năng lực của từng CPU cũng ngày nâng cao, các dự án vi mạch VLSI với hàng triệu, hàng chục tiệu transitor được triển khai. Tuy nhiên giải pháp này cũng nảy sinh những hạn chế về kỹ thuật: tốc độ truyền thông tin không vượt qua tốc độ ánh sáng; khoảng cách gần nhất giữa hai thành phần không thể giảm thiểu quá nhỏ v.v.

Song song với giải pháp tăng cường năng lực từng CPU là giải pháp liên kết nhiều CPU để tạo ra một hệ thống chung có năng lực đáng kể: việc xử lý song song tạo ra nhiều lợi điểm. Thứ nhất, chia các phần nhỏ công việc cho mỗi CPU đảm nhận, năng suất tăng không chỉ theo tỷ lệ thuận với một hệ số nhân mà còn cao hơn do không mất thời gian phải thực hiện những công việc trung gian. Thứ hai, giải pháp này còn có lợi điểm tích hợp các hệ thống máy đã có để tạo ra một hệ thống mới với sức mạnh tăng gấp bội.

Chúng ta khảo sát một số nội dung chọn giải pháp đa xử lý theo nghĩa một hệ thống tính toán được tổ hợp không chỉ một CPU mà nhiều CPU trong một máy tính (hệ đa xử lý tập trung) hoặc nhiều máy tính trong một

hệ thống thống nhất. Gọi chung các hệ có nhiều CPU như vậy là hệ đa xử lý.

Phân loại các hệ đa xử lý

Có một số cách phân loại các hệ đa xử lý:

• Phân loại theo vị trí đặt các CPU: tập trung hoặc phân tán.

Các siêu máy tính (supercomputer) là các ví dụ về hệ đa xử lý tập trung. Đặc trưng của hệ thống này là các CPU được liên kết với nhau trong một máy tính duy nhất đảm bảo độ kết dính phần cứng chặt. Ví dụ về hệ đa xử lý phân tán là các hệ thống tính toán phân tán dựa trên mạng máy tính với độ kết dính phần cứng lỏng.

• Phân loại theo đặc tính của các CPU thành phần: hệ đa xử lý thuần nhất hoặc hệ đa xử lý không thuần nhất v.v. Một ví dụ quen thuộc về hệ không thuần nhất là thiết bị xử lý trong máy vi tính gồm CPU xử lý chung và CPU xử lý dấu phảy động. Siêu máy tính ILLIAC-IV gồm nhiều CPU có đặc trưng giống nhau là một ví dụ về hệ thuần nhất.

• Cách phân loại điển hình là dựa theo kiểu các CPU thành phần tiếp nhận và xử lý dữ liệu trong một nhịp làm việc. Cách phân loại này bao gồm cả máy tính đơn xử lý thông thường:

- Đơn chỉ thị, đơn dữ liệu (SISD: Single Data Single Instruction) được thể hiện trong máy tính thông thường; Mỗi lần làm việc, CPU chỉ xử lý "một dữ liệu" và chỉ có một chỉ thị (instruction, câu lệnh) được thực hiện. Đây là máy tính đơn xử lý.

- Đơn chỉ thị, đa dữ liệu (SIMD: Single Instruction Multiple Data):

Các bộ xử lý trong cùng một nhịp làm việc thực hiện chỉ cùng một chỉ thị. Ví dụ như phép cộng hai vector cho trước: Các CPU thành phần đều thực hiện các phép cộng theo đối số tương ứng tại mỗi CPU; sau đó, chọn tiếp chỉ thị mới để tiếp tục công việc.

Thông thường, hệ thống có bộ phận điều khiển riêng cho việc chọn chỉ thị và mọi CPU thành phần cùng thực hiện chỉ thị đó (bộ xử lý ma trận).

- Đa chỉ thị, đơn dữ liệu (MISD: Multiple Instruction Single Data):

Trong các máy tính thuộc loại này, hệ thống gồm nhiều CPU, các CPU liên kết nhau tuần tự: output của CPU này là input của CPU tiếp theo (Bộ xử lý vector). Các CPU kết nối theo kiểu này được gọi là kết nối "dây chuyền".

- Đa chỉ thị, đa câu lệnh (MIMD):

Mỗi CPU có bộ phân tích chương trình riêng; chỉ thị và dữ liệu gắn với mỗi CPU: nhịp hoạt động của các CPU này hoàn toàn "độc lập nhau".

1.2.Hệ điều hành đa xử lý tập trung

Hệ đa xử lý tập trung hoạt động trên các máy tính có nhiều CPU mà điển hình là các siêu máy tính: CRAY-1; ILLIAC-IV. –IV, hitachi và các máy tính nhiều xử lý hiện nay(máy tính của khoa CNTT, trường ĐHKHTN-ĐHQGHN có hai bộ xử lý) v.v. Các tài nguyên khác CPU có thể được phân chia cho các CPU trong các hệ điều hành đa xử lý, hai bài toán lớn nhất có thể kể đến là phân phối bộ nhớ và phân phối CPU.

1.2.1.Phân phối bộ nhớ

Các quá trình xuất hiện trong bộ nhớ chung. Việc phân phối bộ nhớ được tiến hành cho các quá trình theo các chế độ điều khiển bộ nhớ đã cài đặt: Phân phối theo chế độ mẻ hay phân phối gián đoạn.

Để tăng tốc độ làm vieecj với bộ nhớ(bài toán xử lý con trỏ ngoài v.v.) có thể gắn với mỗi CPU một catche nhớ (máy ILLIAC-IV mõi CPU có catche la 2KB). Phân ra hai loại thâm nhập catche: tĩnh và động. Thâm nhập tĩnh; Mỗi CPU chỉ thâm nhập catche tương ứng, không thâm nhập dữ liệu tại vùng catche của các CPU khác. Thâm nhập động cho phép CPU của máy này có thể thâm nhập các catche của CPU khác (như máy ILLIAC-IV có thể cho phép lấy thông tin trên catche của các máy kề cận).

1.2.2.Bài toán điều khiển CPU

Có nhiều CPU, việc điều khiển CPU được phân ra một số cách như sau:

Toàn bộ các CPU dành cho một quá trình: Một quá trình được phân phối CPU, song tự quá trình nói trên nảy sinh các quá trình con; mỗi quá trình con được giải quyết trên mỗi CPU. Các "quá trình con" có thể coi như một tính toán hết sức đơn giản nào đó: Máy tính đa xử lý vector chia các công đoạn của quá trình và mỗi CPU thực hiện một quá trình con (một công đoạn) trong quá trình đó. Máy tính đa xử lý ma trận cho phép mọi CPU cùng thực hiện một thao tác: Ví dụ cộng hai ma trận 20x20 nếu coa 400 CPU lắp 20x20 thì chỉ một nhịp thời gian sẽ song toàn bộ cộng hai ma trận cỡ đó.

Mỗi CPU xử lý một "quá trình con" riêng. Ngay trong máy vi tính, mỗi CPU 80x86 xử lý chung, 80x87 xử lý các phép toán dấu phẩy động. Trong thời gian 87 hoạt động, 80x86 có thể dùng cho quá trình khác. Việc phân ra như thế làm cho mức độ chuyên nghiệp hóa cao hơn, giá trị tổng thể tăng lên đáng kể.

Về vòng xếp hàng có thể xem xét theo hai mô hình đưới đây:

- Mô hình tĩnh: Hoặc mỗi CPU có một dòng xếp hàng riêng; mỗi bài toán được gắn với từng dòng xếp hàng, việc điều khiển mỗi dòng xếp hàng như đã được chỉ ra độc lập với các dòng xếp hàng khác, mỗi quá trình được phát sinh gắn với một dòng xếp hàng nào đó;

- Mô hình động: Toàn bộ hệ thống gồm một hay một vài dòng xếp hàng, các quá trình được xếp lên các CPU khi rỗi (có thể sử dụng kiểu dữ liệu semaphone nhiều giá trị để phân phối CPU cho các quá trình này).

2.Thuật toán song song và ngôn ngữ lập trình song song

Mục tiêu:

- Hiểu được thuật toán song song

2.1.Thuật toán song song

Ví dụ 1: Tính (a1a2+a3a4)(a5a6+a7a8)

1.Với máy một câu lệnh, với>=4 CPU

a1a2 a3a4 a5a6 a7a8 (cùng thực hiện 4 lệnh nhân)

a1a2 + a3a4 a5a6 + a7a8 (cùng thực hiện 2 lệnh cộng)

(a1a2 + a3a4)(a5a6 + a7a8) (thực hiện chỉ 1 lệnh nhân)

2.Với máy 2 CPU, trong một nhịp cho phép các CPU thành phần thực hiện các lệnh khác nhau:

a1a2 a3a4 (cùng thực hiện 2 lệnh nhân)

a1a2 + a3a4 a5a6 (một lệnh cộng, một lệnh nhân)

a7a8 (một lệnh nhân)

a5a6 + a7a8 (một lệnh cộng)

(a1a2 + a3a4)(a5a6 + a7a8) (một lệnh nhân)

Tiến trình chia các quá trình con độc lập, có thể thực hiện song song. Trong khi thực hiện song song có hoặc không có xảy ra tranh chấp tài nguyên.

Mỗi quá trình con lại thực hiện tuần tự trong hệ thống.

Việc xem xét các thuật toán song song không chỉ ở mức độ quá trình, mà còn ở giữa các "bước tính toán" (bao gồm mức thấp hơn như "phép toán" hay "lệnh") trong một thuật toán giải quyết bài toán chung (xem hai thí dụ trên). Như vậy cần nghiên cứu tính song song ngay trong một thuật toán.

Giả sử tập hợp các "bước tính toán" trong một thuật toán được phân hoạch thành một số nhóm nào đó theo tính chất:

1.Các nhóm được đánh chỉ số với ý nghĩa các "bước" trong nhóm với chỉ số nhỏ cần được thực hiện trước các "bước" trong mỗi nhóm có chỉ số cao: nhóm chỉ số nhỏ là "đi trước".

2. "bước" trong mỗi nhóm chỉ phụ thuộc dữ liệu theo hoặc dữ liệu vào, hoặc kết quả của các "bước" thuộc các nhóm đi trước; các "bước" trong cùng một nhóm được thực hiện một cách song song.

Dạng trình bày thuật toán theo các nhóm như nêu trên được gọi là "dạng song song" của thuật toán.

Mỗi nhóm trong dạng song song được gọi là một "lớp" (tầng); số lớp được gọi là "chiều cao" trong dạng song song; số cực đại các bước trong

một lớp được gọi là "bề rộng" của dạng song song. Như vậy, khi đã có dạng song song, cần thể hiện trên các hệ thống đa xử lý(tính toán song song). Có thể nhận thấy một sự tương ứng:

- Chiều cao tương ứng với thời gian thực hiện thuật toán;
- Bề rộng tương ứng với số CPU đủ để thực hiện thuật toán với chiều cao nói trên.

Một bài toán được đặt ra: Để giả quyết một bài toán khi cho trước máy tính đa xử lý với N CPU, cần tìm dạng song song sao cho có bề rộng không vượt quá N và chiều cao nhỏ nhất.

Một bài toán ít gặp hơn song cũng cần được giải quyết: Coi rằng số CPU là tùy ý có thể có. Tìm dạng song song có chiều cao nhỏ nhất.

Có thể biểu diễn thuật toán goomg một số bước: Mỗi bước như một đỉnh của đồ thị, bước đi sau bước khác (trực tiếp) bằng một cung đi từ bước trước tới bước sau: lúc đó nhận được đồ thị định hướng hữu hạn, các bước được xây dựng sao cho không có chu trình trong đồ thị nói trên.

2.2.Ngôn ngữ lập trình song song

Lợi điểm của việc nghiên cứu tính toán song song chỉ khả thi khi thực sự trên máy tính có cơ chế thực hiện các yêu cầu tính toán song song. Điểr đó dẫn đến việc hình thành các ngôn ngữ lập trình song song với một số tính chất sau đây:

Thể hiện tính toán song song trong các ngôn ngữ theo hai hướng: Tính toán song song trên các kiểu dữ liệu như vector, ma trận, cấu trúc…, và thực hiện song song các quá trình con;

Thực hiện song song theo kiểu dữ liệu trong đó mỗi thao tác thực hiện trong một nhịp thời gian, trong một nhịp các thành phần của cùng một ma trận được tính toán (cộng, trừ, nhân…). Điều này thể hiện rất rõ trong các máy tính vector và ma trận.

Thực hiện song song các quá trình con: như đã biết dạng song song của một thuật toán được trình bày thông qua việc "tuần tự" thực hiện các lớp, trong đó, các bước trong cùng một "lớp" được thực hiện song song nhau.

Ví dụ, nếu $V=\sum V_j$ thì có thể viết thuật toán dạng song song là:

$V_1 => V_2 => V_3 \ldots => V_k$

Tring đó, các "nhóm" V_j là tuần tự nhau từ nhóm này sang nhóm khác, hoặc trình bày trong chương trình:

V_1

V_2

...

V_k

Trong đó, mỗi V_j liệt kê nhóm các quá trình con chạy song song với nhau được. Như vậy, tương ứng với mỗi lớp V_j có cách thức cho biết các quá trình con thuộc lớp đó được thực hiện song song. Trong một số trường hợp phức tạp hơn, cho phép đối với mỗi bước trong một lớp còn kèm theo thời gian thực hiện bước đó, độ ưu tiên của từng bước trong lớp để ưu tiên trong việc phân phối CPU.

Một số hướng nghiên cứu hiện nay có liên quan đến thuật toán song song được quan tâm:

- Biến đổi cây tính toán tương đương và tìm dạng song song chuẩn cho các thuật toán tuần tự;

- Nghiên cứu các khía cạnh song song và các thuật toán song song trong phương pháp số. Một trong những áp dụng kha điển hình của các supercomputer là thực hiện tính toán song song trong những bài toán tính toán Fourier nhanh.

3.Hệ điều hành đa xử lý phân tán

Mục tiêu:

- Hiểu được những nét cơ bản về hệ điều hành đa xử lý nhằm trang bị khả năng tự nghiên cứu trong tương lai.

3.1.Giới thiệu hệ phân tán

Như chúng ta đã biết HĐH hiện đại thường tập trung vào chức năng máy tính ảo, nhấn mạnh mức dịch vụ hệ thống và vì vậy thuận tiện hơn quan niệm HĐH phân tán như một bộ tích hợp các dịch vụ hệ thống cho phép trình diễn cái nhìn trong suốt tới hệ thống máy tính với tài nguyên và điều khiển phân tán (đặt tại nhiều vị trí địa lý khác nhau). Có thể nói HĐH phân tán là HĐH kết nối chặt về phần mềm trên nền tảng kết nối lỏng về phần cứng. Theo một cách nói khác, HĐH phân tán cung cấp cho người sử dụng cách thức làm việc như với một HĐH tập trung trong điều kiện phân tán cả phần cứng lẫn phần mềm.

Một vấn đề đặt ra cho chính khái niệm HĐH phân tán. Tồn tại nhiều cách hiểu về HĐH phân tán, song có rất hiếm tài liệu cho một định nghĩa chính thức về HĐH phân tán. Trong nhiều ngữ cảnh, ng-ời ta còn sử dụng khái niệm "hệ phân tán" thay thế cho khái niệm "HĐH phân tán". Chúng ta chấp nhận định nghĩa được đưa ra.

Hệ phân tán là tổ hợp bao gồm các máy tính độc lập với trình diễn hệ thống như một máy tính đơn trước người dùng.

HĐH phân tán được phát triển trên cơ sở một số tiền đề sau đây:

• Thứ nhất, do nhu cầu tăng không ngừng việc chia xẻ tài nguyên và thông tin mà các HĐH đã có từ trước không đáp ứng được.

Trong quá trình triển khai ứng dụng Tin học vào đời sống, các mạng máy tính được phát triển không ngừng, các tài nguyên của các máy tính trong

mạng (phần cứng, phần mềm) ngày càng đ-ợc mở rộng và nâng cấp, giá trị các tài nguyên này càng tăng nhanh dẫn đến sự tăng tr-ởng v-ợt bậc nhu cầu chia xẻ tài nguyên và thông tin trong một hệ thống thống nhất. HĐH tập trung và HĐH mạng thuần túy không đáp ứng được nhu cầu đối với sự tăng trưởng đó.

• Tiền đề thứ hai liên quan đến việc giá các trạm làm việc giảm nhanh chóng.

Việc giảm giá các trạm làm việc làm cho chúng được sử dụng phổ dụng hơn, số lượng và chất lượng các trạm làm việc cũng tăng không ngừng mà từ đó làm tăng yêu cầu xử lý phân tán. Điều này tạo ra nhiều vị trí có khả năng xử lý và lưu trữ thông tin hơn mà từ đó cần thiết phải phối hợp để chia xẻ tốt hơn tiềm năng lưu trữ và xử lý của các vị trí đó.

• Việc sử dụng rộng rãi các mạng

Trên cơ sở việc kết nối mạng để triển khai HĐH mạng tạo nên một cơ sở kỹ thuật hạ tầng (phần cứng, kết nối mạng, phần mềm) làm nền tảng phát triển HĐH phân tán.

• Tính thuần thục về kỹ nghệ phần mềm của các chuyên gia phát triển HĐH. Kinh nghiệm xây dựng HĐH trước đây (HĐH tập trung, HĐH mạng) cho phép nâng cao trình độ để đủ năng lực xây dựng HĐH phân tán.

3.2.Đặc điểm hệ phân tán

Hệ phân tán có các đặc điểm cơ bản là Tính chia xẻ tài nguyên, Tính mở, Khả năng song song, Tính mở rộng, Khả năng thứ lỗi, Tính trong suốt.

Tính chia xẻ tài nguyên

Thuật ngữ tài nguyên được dùng để chỉ tất cả mọi thứ có thể được chia xẻ trong hệ phân tán, bao gồm từ các thiết bị phần cứng (Đĩa, máy in ...) tới các đối tượng (file, các cửa sổ, CSDL và các đối tượng dữ liệu khác).

Trong hệ phân tán, chia xẻ tài nguyên được hiểu là tài nguyên của hệ thống được các QT chia xẻ (sử dụng chung) mà không bị hạn chế bởi tình trạng phân tán tài nguyên theo vị trí địa lý.

Việc chia xẻ tài nguyên trên hệ phân tán - trong đó tài nguyên bị lệ thuộc về mặt vật lý với một máy tính nào đó - được thực hiện thông qua truyền thông. Để chia xẻ tài nguyên một cách hiệu quả thì mỗi tài nguyên cần phải được quản lý bởi một chương trình có giao diện truyền thông, các tài nguyên có thể truy nhập, cập nhật được một cách tin cậy và nhất quán. Quản lý tài nguyên ở đây bao gồm lập kế hoạch và dự phòng, đặt tên các lớp tài nguyên, cho phép tài nguyên được truy cập từ nơi khác, ánh xạ tên tài nguyên vào địa chỉ truyền thông ...

Tính mở

Tính mở của một hệ thống máy tính là tính dễ dàng mở rộng phần cứng (thiết bị ngoại vi, bộ nhớ, các giao diện truyền thông ...) và phần mềm (các

mô hình HĐH, các giao thức truyền thông, các dịch vụ chia xẻ tài nguyên ...) của nó. Nói một cách khác, tính mở của hệ thống phân tán mang ý nghĩa bao hàm tính dễ dàng cấu hình cả phần cứng lẫn phần mềm của nó.

Tính mở của hệ phân tán được thể hiện là hệ thống có thể được tạo nên từ nhiều loại phần cứng và phần mềm của nhiều nhà cung cấp khác nhau với điều kiện các thành phần này phải theo một tiêu chuẩn chung (liên quan đến HĐH là tính đa dạng tài nguyên; liên quan đến nhà cung cấp tài nguyên là tính chuẩn).

Tính mở của Hệ phân tán được xem xét theo mức độ bổ sung thêm các dịch vụ chia xẻ tài nguyên mà không phá hỏng hay nhân đôi các dịch vụ đang tồn tại. Tính mở được hoàn thiện bằng cách xác định hay phân định rõ các giao diện chính của hệ phân tán và làm cho nó tương thích với các nhà phát triển phần mềm (tức là các giao diện chính của HĐH phân tán cần phổ dụng).

Tính mở của HĐH phân tán được thi hành dựa trên việc cung cấp cơ chế truyền thông giữa các QT và công khai các giao diện được dùng để truy cập tài nguyên chung.

Khả năng song song

Hệ phân tán hoạt động trên một mạng truyền thông có nhiều máy tính, mỗi máy tính có thể có một hoặc nhiều CPU. Trong cùng một thời điểm nếu có từ hai QT trở lên cùng tồn tại, ta nói rằng chúng được thực hiện đồng thời. Việc thực hiện các QT đồng thời theo cơ chế phân chia thời gian (một CPU) hay song song (nhiều CPU).

Khả năng làm việc song song trong hệ phân tán được thi hành do hai tình huống:

- Nhiều người sử dụng đồng thời đưa ra các lệnh hay tương tác với chương trình ứng dụng (đồng thời xuất hiện nhiều QT khách).
- Nhiều QT phục vụ chạy đồng thời, mỗi QT đáp ứng yêu cầu của một trong số các QT Khách.

Từ điều kiện đa xử lý, khả năng song song của hệ thống phân tán trở thành một thuộc tính của nó.

Khả năng mở rộng

Hệ phân tán có khả năng hoạt động tốt và hiệu quả ở nhiều mức khác nhau. Một hệ phân tán nhỏ nhất có thể hoạt động chỉ cần hai trạm làm việc và một phục vụ file.

Các hệ lớn có thể bao gồm hàng nghìn máy tính, nhiều phục vụ File và phục vụ máy in ...

Khả năng mở rộng của một hệ phân tán được đặc trưng bởi tính không thay đổi phần mềm hệ thống và phần mềm ứng dụng khi hệ thống được mở rộng.

Điều này chỉ đạt ở mức độ nào đó đối với hệ phân tán hiện tại (không thể hoàn toàn như định nghĩa trên). Yêu cầu mở rộng không chỉ là mở rộng về phần cứng hay về mạng trên đó hệ thống bao trùm mà còn cần phải được phân tích, đánh giá trên tất cả các khía cạnh khi thiết kế hệ phân tán. Một ví dụ đơn giản là tình huống tần suất sử dụng một file quá cao xuất hiện như kết quả của việc tăng số người sử dụng trên mạng. Để tránh tình trạng tắc nghẽn xảy ra nếu như chỉ có một phục vụ đáp ứng các yêu cầu truy cập file đó, cần nhân bản file đó trên một vài phục vụ và hệ thống được thiết kế sao cho dễ dàng bổ sung phục vụ. Có thể tính đến các giải pháp khác là sử dụng Cache và bản sao dữ liệu.

Khả năng thứ lỗi

Khả năng thứ lỗi thể hiện việc hệ thống không bị sụp đổ bởi các sự cố do các lỗi hành phần (cả phần cứng lẫn phần mềm) trong một bộ phận nào đó.

Việc thiết kế khả năng chịu lỗi của các hệ thống máy tính dựa trên hai giải pháp sau đây:

- Dùng khả năng thay thế để đảm bảo việc hoạt động liên tục và hiệu quả.
- Dùng các chương trình đảm bảo cơ chế phục hồi dữ liệu khi xảy ra sự cố.

Để xây dựng một hệ thống có thể khắc phục sự cố theo cách thứ nhất thì có thể chọn giải pháp nối hai máy tính với nhau để thực hiện cùng một chương trình mà một trong hai máy đó chạy ở chế độ Standby (không tải hay chờ). Giải pháp này khá tốn kém vì phải nhân đôi phần cứng của hệ thống.

Giải pháp khác nhằm giảm bớt phí tổn là dùng nhiều phục vụ khác nhau cung cấp các ứng dụng quan trọng để các phục vụ này có thể thay thế nhau khi sự cố xuất hiện. Khi không có sự cố thì các phục vụ chạy bình th-ờng (nghĩa là vẫn phục vụ ác yêu cầu của khách). Khi xuất hiện sự cố trên một phục vụ nào đó, các ứng dụng khách tự chuyển hưng sang các phục vụ còn lại. Với cách thứ hai thì phần mềm phục hồi được cho trạng thái dữ liệu hiện thời (trạng thái trưc khi xảy ra sự cố) có thể được lôĩ được phát hiện. Chú ý rằng với cách thức này, một mặt thì cùng một dịch vụ có thể được sẵn sàng trên nhiều máy và mặt khác, trên một máy lại có sẵn một số dịch vụ khác nhau.

Hệ phân tán cung cấp khả năng sẵn sàng cao để đối phó với các sai hỏng phần cứng. Khả năng sẵn sàng của hệ thống được đo bằng tỷ lệ thời gian mà hệ thống sẵn sàng làm việc so với thời gian có sự cố. Khi một máy trên mạng sai hỏng thì chỉ có công việc liên quan đến các thành phần sai hỏng bị ảnh hưởng. Người sử dụng có thể chuyển đến một trạm khác nếu máy họ

đang sử dụng bị hỏng, một QT phục vụ có thể được khởi động lại trên một máy khác.

Tính trong suốt

Như đã được trình bày ở trên, tính trong suốt là tính chất căn bản của hệ phân tán. Tính trong suốt của hệ phân tán được hiểu như là sự che khuất đi các thành phần riêng biệt của hệ thống máy tính (phần cứng và phần mềm) đối với người sử dụng và những người lập trình ứng dụng. Người sử dụng có quyền truy cập đến dữ liệu đặt tại một điểm dữ liệu ở xa một cách tự động nhờ hệ thống mà không cần biết đến sự phân tán của tất cả dữ liệu trên mạng. Hệ thống tạo cho người dùng cảm giác là dữ liệu được coi như đặt tại máy tính cục bộ của mình. Các thể hiện điển hình về tính trong suốt của HĐH phân tán được trình bày trong phần sau.

CÂU HỎI CỦNG CỐ BÀI HỌC

1.Thế nào là hệ điều hành đa xử lý tập trung.

2. Phân biệt hệ điều hành đa xử lý tập trung với hệ điều hành đa xử lý phân tán.

3.trình bày thuật toán song song và ngôn ngữ lập trình song song.

TÀI LIỆU THAM KHẢO

1. Nguyễn Thanh Tùng. *Giáo trình hệ điều hành.1995*.Khoa Công nghệ thông tin – Đại Học Bách Khoa Hà Nội.
2. Hà Quang Thụy. *Giáo trình nguyên lý các hệ điều hành. 1998*. Đại học Khoa học tự nhiên – Đại học Quốc gia Hà Nội.
3. Đặng Vũ Tùng. *Giáo trình nguyên lý hệ điều hành.* Nhà xuất bản hà Nội 2005- 3. "*Tập slide bài giảng*". Bộ môn Các hệ thống thông tin.
4. A. Silberschatz, P. B. Galvin, G. Gagne, Wiley & Sons. "*Operating system Concepts*". 2002.
5. William Stallings, "*Operating Systems – Internals and Design Principles"*, Pearson Education International, 2005.