TRƯỜNG CAO ĐẲNG NGHỀ CÔNG NGHIỆP HÀ NỘI
Chủ biên: Lê Văn Hùng
Đồng tác giả: Trần Thị Ngân

GIÁO TRÌNH
NGUYÊN LÝ HỆ ĐIỀU HÀNH

(Lưu hành nội bộ)

Hà Nội năm 2012

MỤC LỤC

LỜI GIỚI THIỆU

Trong hệ thống kiến thức chuyên ngành trang bị cho sinh viên nghề Quản trị mạng máy tính, môn học Nguyên lý hệ điều hành góp phần cung cấp những nội dung liên quan đến việc mô tả các phương pháp giải quyết các bài toán điều khiển hoạt động của hệ thống máy tính

Các nội dung chính được trình bày trong tài liệu này gồm các chương:

-Giới thiệu chung về hệ điều hành

- Điều khiển dữ liệu

- Điều khiển bộ nhớ

- Điều khiển CPU và Tiến trình

- Hệ điều hành đa xử lý

Mặc dầu có rất nhiều cố gắng, nhưng không tránh khỏi những khiếm khuyết, rất mong nhận được sự đóng góp ý kiến của độc giả để giáo trình được hoàn thiện hơn.

NGUYÊN LÝ HỆ ĐIỀU HÀNH

Mã môn học/mô đun:MH 10

Vị trí, ý nghĩa, vai trò môn học/mô đun:

- Vị trí: Môn học được bố trí sau khi sinh viên học xong các môn học chung, trước các môn học, mô đun đào tạo chuyên môn nghề.
- Tính chất: Là môn học cơ sở bắt buộc.

Mục tiêu của môn học/mô đun:

- 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, thực hiện của hệ điều hành;
- Hiểu cách giải quyết các vấn đề phát sinh trong hệ điều hành.
- Bố trí làm việc khoa học đảm bảo an toàn cho người và phương tiện học tập.

Nội dung chính của môn học /mô đun (danh sách các chương mục/bài học...):

Số TT	Tên chương, mục	Thời gian			
		Tổng số	Lý thuyết	Thực hành	Kiểm tra* (LT hoặc TH)
I	**Tổng quan về hệ điều**	**5**	**5**	**0**	**0**

	hành				
	Khái niệm về hệ điều hành	2	2	0	0
	Phân loại hệ điều hành	2	2	0	0
	Sơ lược lịch sử phát triển của HĐH	1	1	0	0
II	**Điều khiển dữ liệu**	**15**	**9**	**5**	**1**
	Các phương pháp tổ chức và truy nhập dữ liệu	5	3	2	0
	Bản ghi và khối	2	1	1	0
	Điều khiển buffer	2	1	1	0
	Quy trình chung điều khiển vào – ra	2	2	0	0
	Tổ chức lưu trữ dữ liệu trên bộ nhớ ngoài	4	2	1	1
III	**Điều khiển bộ nhớ**	**20**	**10**	**9**	**1**
	Quản lý và bảo vệ bộ nhớ	2	2	0	0
	Điều khiển bộ nhớ liên tục theo đa bài toán	8	3	5	0
	Điều khiển bộ nhớ gián đoạn	10	4	5	1
IV	**Điều khiển CPU, Điều khiển quá trình**	**25**	**12**	**12**	**1**
	Các khái niệm cơ bản	2	2	0	0
	Trạng thái của quá trình	5	2	3	0
	Điều phối quá trình	3	1	2	0
	Các thuật toán lập lịch	10	4	6	0
	Hệ thống ngắt	1	1	0	0
	Hiện tượng bế tắc	4	2	1	1
V	**Hệ điều hành đa xử lý**	**10**	**7**	**2**	**1**
	Hệ điều hành đa xử lý tập trung	5	3	2	0
	Hệ điều hành đa xử lý phân tán	5	3	1	1
Cộng		**75**	**43**	**28**	**4**

YÊU CẦU VỀ ĐÁNH GIÁ HOÀN THÀNH MÔN HỌC/MÔ ĐUN

-Về kiến thức: Được đánh giá qua bài kiểm tra viết, trắc nghiệm và bài tập lớn cuối môn đạt được các yêu cầu sau:

- ✓ Hiểu vai trò của hệ điều hành trong hệ thống máy tính.
- ✓ Biết các giai đoạn phát triển của hệ điều hành.
- ✓ Hiểu các chức năng và nguyên lý làm việc của hệ điều hành.
- ✓ Hiểu cách giải quyết các vấn đề phát sinh liên quan đến hệ điều hành.

-Về kỹ năng: Đánh giá kỹ năng thực hành của sinh viên trong các bài thực hành:

- ✓ Tính toán các giá trị tài nguyên theo các mẫu ví dụ tương ứng;
- ✓ Thuyết trình nhận thức về các thuật toán chia sẻ tài nguyên và điều phối các quá trình trên CPU, giải pháp phòng chống bế tắc và cách phòng tránh bế tắc
- ✓ Thao tác thực hành các kỹ năng, xử lý các tình huống với các hệ điều hành cụ thể được cài đặt. (WINDOWS, HĐH Mạng ...)

-Về thái độ: Thể hiện tính cẩn thận, tư duy logic, khoa học, tìm tòi, sáng tạo.

CHƯƠNG 1:GIỚI THIỆU CHUNG VỀ HỆ ĐIỀU HÀNH

Chương 1: G**iới thiệu chung về hệ điều hành Mã chương**:..M1

Mục tiêu:

- 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.

1. Khái niệm về hệ điều hành

Mục tiêu: Nắm được yêu cầu cần có hệ điều hành;
Nắm được khái niệm hệ điều hành

1.1. Tài nguyên hệ thống

Tài nguyên của một trung tâm máy tính được tổng hợp từ ba thành tố, đó là *tài nguyên về phần cứng, tài nguyên về phần mềm và tài nguyên về nguồn nhân lực* của trung tâm máy tính đó.

Trong các tài liệu giới thiệu về một trung tâm máy tính bất kỳ, các số liệu thống kê về phần cứng (số lượng và chủng loại máy tính, hệ thống thiết bị ngoại vi, khả năng liên kết với môi trường ngoài v.v…) luôn là những yếu tố được quan tâm sớm nhất và là thành tố dễ nhận biết nhất về sức mạnh của trung tâm máy tính đó.

Tài nguyên về phần mềm cũng được chú ý thông qua các thông tin về hệ điều hành được sử dụng, về các phần mềm ứng dụng đã có tại cơ sở tính toán đó. Hiện nay, tại những trung tâm tính toán mạnh, giá trị (tính theo tiền) thực sự của tài nguyên phần mềm lại cao hơn và vượt trội nhiều so với giá trị của tài nguyên phần cứng.

Tài nguyên về nguồn nhân lực cũng được chú ý, tuy rằng trong một số trường hợp, thành tố này lại khó nhận biết và khó đánh giá hơn so hai loại tài nguyên đã nói ở trên. Năng lực về nguồn nhân lực trong hệ thống nhằm đảm bảo việc thực hiện chức năng bảo trì, phục vụ và phát triển hệ thống (kỹ sư hệ thống, kỹ thuật viên, thao tác viên v.v…) thực sự lại đánh giá hơn rất nhiều so với phần cứng và phần mềm.

Tuy nhiên, trong bài giảng này, chúng ta hạn chế trong một phạm vi tiếp cận là mọi công việc của hệ điều hành bắt đầu từ hệ thống phần cứng có sẵn và hệ điều hành cần phải hoạt động nhằm phát huy cao nhất năng lực của hệ thống phần cứng đó và vì vậy chúng ta chỉ đề cập đến tài nguyên về phần cứng (có thể kể tới một phần về tài nguyên phần mềm) và định hướng tới vấn đề phát huy hiệu quả khai thác các tài nguyên đó.

Để định hướng tới mục tiêu phát huy hiệu quả các thành phần trong tài nguyên phần cứng, cần xem xét một số đặc trưng cơ bản và đánh giá giá trị của mỗi thành phần trong hệ thống phần cứng, hướng tới mục đích đưa ra được các chiến lược ưu tiên thích đáng (hoặc khả dụng) đối với mỗi thành phần khi xây dựng hệ thống các chương trình điều khiển sự hoạt động của máy tính.

Theo cách tiếp cận của hệ điều hành, các tài nguyên điển hình thuộc phần cứng bao gồm: *thiết bị xử lý trung tâm (CPU), bộ nhớ trong, và hệ thống vào – ra* (kênh, thiết bị điều khiển thiết bị vào ra và thiết bị vào ra, bộ nhớ ngoài v.v…). CPU và bộ nhớ trong thuộc và khu vực trung tâm còn hệ thống vào – ra thường được xếp vào khu vực ngoại vi của hệ thống máy tính.

Trong các thiết bị nói trên, đáng chú ý nhất phải kể đến là CPU và bộ nhớ trong.

Bộ xử lý trung tâm (Central Processing Unit-CPU)

Trước hết chúng ta xem xét về các đặc trưng liên quan đến CPU. Việc đánh giá tài nguyên CPU về cơ bản cũng dựa trên các đặc trưng này: tốc độ xử lý, độ dài từ máy, phương pháp thiết kế hệ lệnh máy trong CPU.

Tốc độ xử lý là thông số thể hiện mức độ làm việc nhanh chậm của CPU dựa trên các đơn vị biểu diễn tốc độ. Tốc độ xử lý của CPU thường được tính theo *tần số đồng hồ nhịp* (với đơn vị là MHz-triệu nhịp trong 1 giây) khi xem xét tần số đồng hồ nhịp *hoặc số lượng phép tính cơ bản được thực hiện trong một giây* (với đơn vị là MIPS – Million Instruction Per Second – triệu phép tính cơ bản trong một giây) khi xem xét theo tốc độ thực hiện phép tính (phép cộng tĩnh – không dấu của một CPU thường được coi là phép tính cơ bản của CPU đó). Thông thường, đơn vị đo MHz được dùng cho một CPU cụ thể hoặc một máy vi tính còn đơn vị đo MIPS được dùng cho một hệ thống CPU của một máy tính lớn.

Độ dài từ máy: Từ máy là lượng thông tin đồng thời mà CPU xử lý trong một nhịp làm việc. Độ dài từ máy chính là số lượng bit nhị phân của toán hạng đối số trong phép tính cơ bản của CPU. Trong thời gian gần đây, chúng ta đã quen thuộc với các CPU 8 bit, 16 bit, 32 bit, 64 bit, … và số lượng bit nói trên chính là độ dài từ máy.

Độ dài của từ máy có quan hệ với tốc độ xử lý. Khi nói đến năng lực hoạt động (tốc độ xử lý thông tin) thực sự của một CPU mà chỉ nói đến tốc độ xử lý mà không nói kèm theo độ dài từ máy là chưa hoàn toàn đầy đủ. Điều đó có thể được diễn giải theo phát biểu như sau "năng lực hoạt động thực sự củaCPU được đánh giá thông qua tốc độ xử lý và độ dài từ máy".

Bộ nhớ trong (Operative Memory-OM) có một số đặc trưng tiêu biểu như sau:

Dung lượng bộ nhớ: Khả năng đồng thời lưu trữ thông tin của bộ nhớ trong. Hiện tại dung lượng của bộ nhớ trong từ vài MB đến vài GB.

Đặc trưng tiếp theo của bộ nhớ trong phù hợp với nguyên lý thứ hai theo Von Neumann là: *Bộ nhớ được địa chỉ hóa để truy nhập.* Đa số các máy tính được địa chỉ hóa theo byte và trong một số trường hợp lại được địa chỉ hóa theo từ máy.

Địa chỉ đầu tiên trong bộ nhớ là địa chỉ 0. Lý do của việc chọn địa chỉ đầu tiên là 0 liên quan đến tính chia hết, bởi số 0 chia hết cho mọi số. Khi phân phối bộ nhớ trong cho một đối tượng, trong nhiều trường

hợp, *địa chỉ vùng bộ nhớ trong của đối tượng phải chia hết cho độ dài vùng bộ nhớ* dành cho đối tượng đó hoặc chia hết cho số nào đó (ví dụ, phân phối cho một chương trình trong MS-DOS được bắt đầu bởi địa chỉ đoạn là địa chỉ chia hết cho 16).

Một đặc trưng (hay cũng vậy là một yêu cầu) mang tính bản chất đối với bộ nhớ trong là: *Thời gian truy cập bộ nhớ trong tới mọi địa chỉ nhớ phải đồng nhất;* không thể có sự khác biệt giữa thời gian truy cập tới địa chỉ cao với thời gian truy cập tới địa chỉ thấp. Từ đặc trưng này dẫn đến việc đặt ra một yêu cầu là phải tổ chức bộ nhớ trong theo các khối phân cấp để cục bộ dần và việc cục bộ dần như vậy sẽ làm cho việc truy nhập được cân bằng. Nguồn gốc của yêu cầu này liên quan đến tính xác định của thuật toán, hay nói cách khác đi, yêu cầu này nhằm mục tiêu đảm bảo độ tin cậy của hệ thống máy tính. Chúng ta thường thấy bộ nhớ được cấu trúc từ các " thanh bộ nhớ", mỗi thanh bộ nhớ lại có thể được phân nhỏ hơn và việc truy nhập bộ nhớ theo cách phân cấp dần theo từng thanh, trong mỗi thanh lại theo từng bộ phận nhỏ hơn có trong thanh đó v.v… cho đến khi truy nhập tuần tự trong phần nhỏ nhất chỉ có sai khác thời gian không đáng kể.

Để tăng tốc độ truy nhập của CPU đối với bộ nhớ trong, người ta thường gắn CPU với bộ nhớ tạm thời của CPU (được gọi là bộ nhớ cache của CPU). Bộ nhớ cache là thiết bị nhớ đặc biệt với tốc độ truy cập của CPU tới cache của nó cao hơn rất nhiều so với tốc độ truy cập vào bộ nhớ trong. Trong cache chứa một phần nội dung của bộ nhớ trong thường là phần bộ nhớ hiện thời (chương trình và dữ liệu) được CPU đang hướng tới. Quá trình hướng truy nhập bộ nhớ (theo địa chỉ) của CPU được bắt đầu từ việc hướng tới cache, nếu cache chứa phần bộ nhớ đó thì việc hướng địa chỉ kết thúc và thực hiện công việc, ngược lại thực hiện việc hướng tới bộ nhớ trong theo quy tắc thông thường.

Chương trình chỉ chạy được khi chương trình và dữ liệu tương ứng của chương trình đó phải có mặt tại bộ nhớ trong (chính xác hơn là chỉ cần bộ phận hiện thời của chương trình và dữ liệu liên quan đến bộ phận đó nằm trong bộ nhớ trong). Cách thức sử dụng bộ nhớ trong đóng vai trò quan trọng nhằm đảm bảo chất lượng hoạt động của hệ thống và vì vậy, bài toán điều khiển bộ nhớ trong có độ ưu tiên cao chỉ sau bài toán điều khiển CPU.

Hệ thống ngoại vi

Hệ thống ngoại vi đảm bảo việc chuyển đổi thông tin giữa môi trường ngoài và khu vực trung tâm. Có sự phân cấp trong hệ thống ngoại vi: gần khu vực trung tâm nhất là kênh, sau đó là thiết bị điều khiển thiết bị ngoại vi và ngoài cùng là thiết bị ngoại vi.

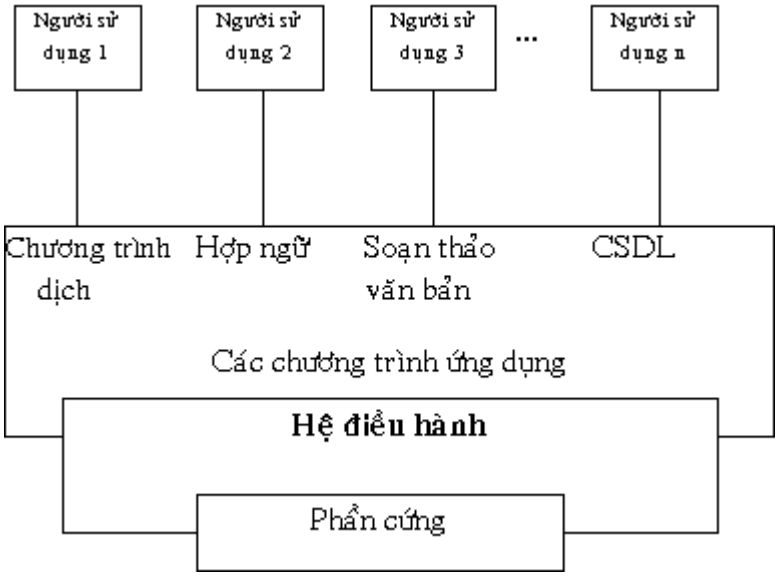
1.2. Khái niệm hệ điều hành

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

-Xét về phía người sử dụng thì hệ điều hành cần phải tạo được môi trường giao diện giữa người sử dụng và máy tính.Thông qua môi trường này cho phép người sử dụng đưa ra các lệnh, chỉ thị điều khiển hoạt động của máy tính.

-Về phía các chương trình ứng dụng thì hệ điều hành phải tạo môi trường để các chương trình hoạt động, cung cấp các cơ chế cho phép kích hoạt hoặc loại bỏ các chương trình ứng dụng.

-Về phía phần cứng thì hệ điều hành phải quản lý các thiết bị một cách có hiệu quả, khai thác được hết các khả năng của thiết bị, cung cấp cho các chương trình và người sử dụng các tài nguyên phần cứng khi có yêu cầu, thu hồi khi cần thiết.



Hình 1.1 Mô hình trừu tượng của hệ thống máy tính

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

2. Phân loại hệ điều hành

Mục tiêu: Nắm được chức năng, phân loại và các thành phần cơ bản trong hệ điều hành.

Nắm được cách thức tải hệ điều hành.

2.1. Các thành phần của hệ điều hành

Theo định nghĩa trên đây, hệ điều hành là một tập hợp các chương trình đã được cài đặt sẵn, mỗi chương trình đảm nhận một chức năng nào đó trong hệ thống chức năng của hệ điều hành. Một trong những nguyên tắc phổ biến nhất để nhận biết các thành phần của hệ điều hành là dựa theo chức năng của các chương trình có trong thành phần đó. Theo nguyên tắc này thì các thành phần cơ bản của hệ điều hành là thành phần điều khiển (control programs), thành phần ứng dụng (application programs, còn gọi thành phần xử lý) và các chương trình tiện ích (utilities).

Thành phần điều khiển

Thành phần điều khiển liên quan đến chức năng điều khiển, phân phối công việc của hệ điều hành. Khi một chương trình điều khiển hoạt động, nó không cho ra một sản phẩm mới (sản phẩm ở đây là các File trên đĩa từ, một kết quả được in ra) mà cho tác động đối với sự điều khiển hoạt động của máy tính. Dưới đây liệt kê một số môdun chương trình điển hình thuộc vào thành phần điều khiển:

-Chương trình dẫn dắt (điều phối chính, monitor, chương trình giải thích lệnh): Tiếp nhận các nhiệm vụ (yêu cầu của người dùng) từ dòng vào các nhiệm vụ, sắp xếp phân phối lịch thực hiện v.v…đối với từng nhiệm vụ, sau đó trả lại kết quả cho người dùng.

-Điều khiển quá trình (bài toán): Thực hiện luân phiên các quá trình (process; bài toán –task) đang tồn tại trong bộ nhớ, mỗi bài toán có các khối chứa thông tin liên quan để chuyển việc thực hiện từ quá trình này sang quá trình khác sao cho việc sử dụng CPU đạt hiệu quả.

-Điều khiển dữ liệu: Tổ chức lưu trữ dữ liệu trên vật dẫn ngoài và đảm bảo truy nhập dữ liệu theo yêu cầu của chương trình người dùng. Công việc vào – ra giữa bộ nhớ ngoài và bộ nhớ trong cũng như do liên quan đến một hệ thống thiết bị ngoại vi đa dạng và phong phú nên điều khiển dữ liệu cũng rất đa dạng.

-Môdun chương trình tải (loader): Đảm bảo việc đưa các môdun chương trình từ bộ nhớ ngoài vào bộ nhớ trong tại một địa chỉ trong bộ nhớ trong. Trong nhiều trường hợp, môdun chương trình tải còn đảm bảo việc chuyển điều khiển để thực hiện (execute) chương trình được tải vào;

-và nhiều thành phần khác…

Thành phần ứng dụng

Thành phần ứng dụng có trong hệ điều hành bao gồm những chương trình mà khi được thực hiện sẽ tạo ra một sản phẩm mới. Các bộ dịch ngôn ngữ lập trình (compiler), các chương trình tính toán, các chương trình soạn thảo v.v…được các hệ điều hành cung cấp nhằm giúp cho người sử dụng có thể chọn lựa phần mềm thích hợp trong lĩnh vực khai thác máy tính của mình. Thành phần ứng dụng rất đa dạng do tính chất đa dạng của yêu cầu người dùng.

Thành phần điều khiển hướng đích là sự hiệu quả khai thác máy tính; còn thành phần ứng dụng hướng đích là việc thỏa mãn nhu cầu của người dùng, tăng hiệu suất sử dụng máy tính đối với từng lớp người dùng.

Các chương trình tiện ích

Các chương trình tiện ích cung cấp thêm cho người dùng các phương tiện phần mềm làm việc với hệ thống máy tính thuận tiện hơn. Các chương trình liên quan đến cách thức thâm nhập hệ thống, các chương trình sao chép, in ấn nội dung của File, các chương trình làm việc với đĩa v.v…được xếp vào thành phần tiện ích.

2.2. Phân loại hệ điều hành

Một trong những nguyên tắc phổ biến nhất để phân loại các hệ điều hành (truyền thống) là dựa theo tính chất hoạt động của thành phần điều

khiển và việc phân loại hệ điều hành ở đây được thực hiện theo nguyên tắc đó. Tính chất hoạt động của chương trình điều khiển liên quan đến cách thức đưa chương trình vào bộ nhớ trong, chọn chương trình đã có ở bộ nhớ trong ra thực hiện v.v… Theo cách thức phân loại này, có thể kể đến hệ điều hành đơn chương trình, hệ điều hành đa chương trình và hệ điều hành thời gian thực.

a. Hệ điều hành đơn chương trình

Trong hệ điều hành đơn chương trình, toàn bộ hệ thống máy tính phục vụ một chương trình từ lúc bắt đầu khi chương trình đó được đưa vào bộ nhớ trong cho đến thời điểm kết thúc chương trình đó. Khi một chương trình người dùng đã được đưa vào bộ nhớ thì nó chiếm giữ mọi tài nguyên của hệ thống và vì vậy chương trình của người dùng khác không thể được đưa vào bộ nhớ trong.

Do các thiết bị vào ra có tốc độ làm việc chậm, nên người ta đã cải tiến chế độ đơn chương trình theo hướng sử dụng cách thức đặc biệt (có tên gọi là SPOOLING: Simultaneous Peripheral Operation OnLine; đôi lúc dùng thuật ngữ chế độ SPOOLING cũng với nghĩa là cách thức này), mà theo cách thức này, mọi vấn đề vào ra liên quan đến chương trình được thực hiện thông qua đĩa từ. Chương trình người dùng, thông qua hệ điều hành, chỉ thực hiện vào ra với đĩa từ, còn việc vào ra giữa đĩa từ với các thiết bị khác lại do cơ chế khác đảm nhận và do vậy, thời gian giải bài toán (thời gian chương trình thực hiện) giảm đi.

b. Hệ điều hành đa chương trình

Đối với hệ điều hành đa chương trình thì trong máy tính, tại mỗi thời điểm có nhiều chương trình đồng thời có mặt ở bộ nhớ trong. Các chương trình này đều có nhu cầu được phân phối bộ nhớ và CPU để thực hiện. Như vậy, bộ nhớ, CPU, các thiết bị ngoại vi…là các tài nguyên của hệ thống được chia sẽ cho các chương trình đó. Đặc điểm quan trọng cần lưu ý là các chương trình này phải được "bình đẳng" khi giải quyết các đòi hỏi về tài nguyên. Khái niệm chương trình nói trong chế độ đa chương trình được dùng để chỉ cả chương trình người dùng lẫn chương trình của hệ điều hành.

Khi so sánh với hệ điều hành đơn chương trình, có thể nhận thấy ngay một điều là đối với một chương trình cụ thể thì trong chế độ đơn chương trình, chương trình đó sẽ kết thúc nhanh hơn (thời gian chạy ngắn hơn) so với khi nó chạy trong chế độ đa chương trình, nhưng bù lại, trong một khoảng thời gian xác định thì theo chế độ đa chương trình sẽ hoàn thiện được nhiều chương trình (giải được nhiều bài toán) hơn, do đó hiệu quả sử dụng máy tính cao hơn.

Như đã đánh giá ở phần trên, một trong những tài nguyên quan trọng nhất của hệ thống máy tính là CPU và việc chia sẽ CPU là một trong những dạng điển hình của việc chia sẽ tài nguyên.

Hệ điều hành hoạt động theo chế độ mẻ

Đây là loại hệ điều hành định hướng tới mục tiêu làm cực đại số lượng các bài toán được giải quyết trong một khoảng đơn vị thời gian (có nghĩa là trong một khoảng thời gian thì hướng mục tiêu hoàn thiện được càng nhiều chương trình càng tốt). Ở nước ta những năm trước đây, các máy tính dùng hệ điều hành OS, DOS phổ biến hoạt động theo chế độ mẻ (batch).

Các hệ điều hành theo chế độ mẻ lại có thể phân biệt thành hai loại điển hình là MFT và MVT.

MFT: *Multiprogramming with Fixed number of Tasks*

Khi hệ thống làm việc, đã quy định sẵn một số lượng cố định các bài toán đồng thời ở bộ nhớ trong: bộ nhớ trong được chia thành một số vùng nhớ cố định, các vùng này có biên cố định mà mỗi vùng được dùng để chứa một chương trình. Mỗi chương trình người dùng chỉ được đưa vào một vùng nhớ xác định tương ứng với chương trình đó. Một chương trình chỉ có thể làm việc trong giới hạn của vùng bộ nhớ trong đang chứa nó.

MVT: *Multiprogramming with Variable number of Tasks*

Khác với chế độ MFT, trong chế độ MVT, bộ nhớ trong không bị chia sẵn thành các vùng, việc nạp chương trình mới vào bộ nhớ trong còn được tiếp diễn khi mà bộ nhớ trong còn đủ để chứa nó.

Chế độ phân chia thời gian (Time Shared System:TSS)

Chế độ phân chia thời gian là chế độ hoạt động điển hình của các hệ điều hành đa người dùng (multi-users). Hệ điều hành hoạt động theo chế độ này định hướng phục vụ trực tiếp người dùng khi chương trình của người dùng đó đang thực hiện, làm cho giao tiếp của người dùng với máy tính là hết sức thân thiện. Liên quan đến hệ điều hành hoạt động theo chế độ này là các khái niệm lượng tử thời gian, bộ nhớ ảo v.v…

Trong hệ TSS, tại cùng thời điểm có nhiều người dùng đồng thời làm việc với máy tính: mỗi người làm việc với máy thông qua trạm cuối (terminal) và vì vậy, hệ thống đã cho phép máy tính thân thiện với người dùng.

Hệ điều hành phân phối CPU lần lượt cho từng chương trình người dùng, mỗi chương trình được chiếm giữ CPU trong một khoảng thời gian như nhau (khoảng thời gian đó được gọi là lượng tử thời gian): có thể thấy phổ biến về lượng tử thời gian điển hình là khoảng 0,05s. Máy tính làm việc với tốc độ cao, chu kỳ quay lại phục vụ cho từng chương trình người

dùng là rất nhanh, mỗi người đều có cảm giác rằng mình chiếm toàn bộ tài nguyên hệ thống.

Bộ nhớ luôn chứa chương trình của mọi người dùng, vì vậy xảy ra tình huống toàn bộ bộ nhớ trong không đủ để chứa tất cả chương trình người dùng hiện đang thực hiện, vì thế đối với hệ điều hành TSS nảy sinh giải pháp sử dụng bộ nhớ ảo: sử dụng đĩa từ như vùng mở rộng không gian nhớ của bộ nhớ trong.

c. Hệ điều hành thời gian thực

Nhiều tài nguyên trong lĩnh vực điều khiển cần được giải quyết không muộn hơn một thời điểm nhất định, và vì vậy, đối với các máy tính trong lĩnh vực đó cần hệ điều hành thời gian thực (RT: Real Time). Trong hệ thời gian thực, mỗi bài toán được gắn với một thời điểm thời gian (deadtime) và bài toán phải được giải quyết không muộn hơn thời điểm đã cho đó: Nếu bài toán hoàn thiện muộn hơn thời điểm đó thì việc giải quyết nó trở nên không còn có ý nghĩa nữa. Hệ thời gian thực có thể được coi như một trường hợp của hệ đa chương trình hoạt động theo chế độ mẻ có gắn thêm thời điểm kết thúc cho mỗi bài toán.

2.3. Tính chất cơ bản của hệ điều hành

a)Tin cậy

Mọi hoạt động, mọi thông báo của HĐH đều phải chuẩn xác, tuyệt đối. chỉ khi nào biết chắc chắn là đúng thì HĐH mới cung cấp thông tin cho người sử dụng. Để đảm bảo được yêu cầu này, phần thiết bị kỹ thuật phải có những phương tiện hỗ trợ kiểm tra tính đúng đắn của dữ liệu trong các phép lưu trữ và xử lý. Trong các trường hợp còn lại HĐH thông báo lỗi và ngừng xử lý trao quyền quyết định cho người vận hành hoặc người sử dụng.

b) An toàn

Hệ thống phải tổ chức sao cho chương trình và dữ liệu không bị xoá hoặc bị thay đổi ngoài ý muốn trong mọi trường hợp và mọi chế độ hoạt động. Điều này đặc biệt quan trọng khi hệ thống là đa nhiệm. Các tài nguyên khác nhau đòi hỏi những yêu cầu khác nhau trong việc đảm bảo an toàn.

c) Hiệu quả

Các tài nguyên của hệ thống phải đợc khai thác triệt để sao chon gay cả điều kiện tài nguyên hạn chế vẫn có thể giải quyết những yêu cầu phức tạp. Một khía cạnh quan trọng của đảm bảo hiệu quả là duy trì đồng bộ trong toàn bộ hệ thống, không để các thiết bị tốc độ chậm trì hoãn hoạt động của toàn bộ hệ thống.

d) Tổng quát theo thời gian

HĐH phải có tính kế thừa, đồng thời có khả năng thích nghi với những thay đổi cso thể cso trong tương lai. Tính thừa kế là rất quan trọng ngay cả với các hệ điều hành thế hệ mới. Đối với việc nâng cấp, tính kế thừa là bắt buộc. Các thao tác, thông báo là không được thay đổi, hoặc nếu có thì không đáng kể và phải được hướng dẫn cụ thể khi chuyển từ phiên bản này sang phiên bản khác, bằng các phương tiện nhận biết của hệ thống. Đảm bảo tính kế thừa sẽ duy trì và phát triển đội ngũ người sử dụng-một nhân tố quan trọng để HĐH có thể tồn tại. Ngoài ra người sử dụng cũng rất quan tâm, liệu những kinh nghiệm và kiến thức của mình về HĐH hiện tại còn được sử dụng bao lâu nữa. Khả năng thích nghi với những thay đổi đòi hỏi HĐH phải được thiết kế theo một số nguyên tắc nhất định.

e) Thuận tiện

Hệ thống phải dễ dàng sử dụng, có nhiều mức hiệu quả khác nhau tuỳ theo kiến thức và kinh nghiệm người dùng. Hệ thống trợ giúp phong phú để người sử dụng có thể tự đào tạo ngay trong quá trình khai thác.

Trong một chừng mực nào đó, các tính chất trên mâu thuẫn lẫn nhau. Mỗi HĐH có một giải pháp trung hoà, ưu tiên hợp lý ở tính chất này hay tính chất khác.

2.4. Phân lớp các chương trình trong thành phần điều khiển

Một trong những cách phân lớp các chương trình thuộc thành phần điều khiển là dựa theo bài toán mà lớp chương trình dó giải quyết. Các bài toán cơ bản nhất nảy sinh trong quá trình điều khiển hệ thống máy tính được liệt kê như dưới đây.

Điều khiển dữ liệu

Điều khiển dữ liệu (điều khiển file, điều khiển vào ra) bao gồm các môdun chương trình của hệ điều hành liên quan đến việc tổ chức lưu trữ và quản lý dữ liệu trên vật dẫn ngoài, chuyển dữ liệu từ bộ nhớ ngoài vào bộ nhớ trong và ngược lại. Quá trình chuyển dữ liệu thường được thực hiện qua hai giai đoạn: chuyển đổi dữ liệu thực sự giữa khu vực ngoại vi với bộ nhớ trong và chuyển đổi dữ liệu nội bộ bộ nhớ trong. Tính đa dạng của thiết bị ngoài dẫn tới việc có nhiều cách tổ chức, lưu trữ, cập nhật dữ liệu v.v…

Điều khiển CPU, điều khiển quá trình

Để tối ưu hóa sự làm việc của CPU thì hoạt động của CPU được đảm bảo từ hệ thống điều khiển CPU: làm như thế nào để thời gian hoạt động có ích của CPU là cao nhất. Có thể tiếp cận theo khía cạnh điều khiển quá trình (chương trình, bài toán) với việc phân chia tài nguyên dùng chung, đồng bộ hóa, xử lý song song khi quan tâm đến mối quan hệ giữa các quá trình đang đồng thời tồn tại.

Điều khiển bộ nhớ

Việc quản lý bộ nhớ trong để nắm vững vùng nhớ nào rỗi, vùng nhớ nào bận và việc phân phối bộ nhớ cho một chương trình và giải phóng bộ nhớ khi nó thực hiện xong là chức năng chính của điều khiển bộ nhớ. Điều khiển bộ nhớ làm sao đạt mục tiêu sử dụng bộ nhớ càng tối ưu càng tốt.

2.5. Chức năng cơ bản của hệ điều hành

a) Quản lý tiến trình

Một ***tiến trình*** là một chương trình đang được thi hành. Một tiến trình phải sử dụng tài nguyên như thời gian sử dụng CPU, bộ nhớ, tập tin, các thiết bị nhập xuất để hoàn tất công việc của nó. Các tài nguyên này được cung cấp khi tiến trình được tạo hay trong quá trình thi hành.

Một tiến trình là hoạt động (active) hoàn toàn-ngược lại với một tập tin trên đĩa là thụ động (passive)-với một bộ đếm chương trình cho biết lệnh kế tiếp được thi hành.Việc thi hành được thực hiện theo cơ chế tuần tự , CPU sẽ thi hành từ lệnh đầu đến lệnh cuối.

Một tiến trình được coi là một đơn vị làm việc của hệ thống. Một hệ thống có thể có nhiều tiến trình cùng lúc , trong đó một số tiến trình là của hệ điều hành, một số tiến trình là của người sử dụng. các tiến trình này có thể diễn ra đồng thời.

Vai trò của hệ điều hành trong việc quản lý tiến trình là :

- Tạo và hủy các tiến trình của người sử dụng và của hệ thống.

- Tạm dừng và thực hiện tiếp một tiến trình.

- Cung cấp các cơ chế đồng bộ tiến trình.

- Cung cấp các cơ chế giao tiếp giữa các tiến trình.

- Cung cấp cơ chế kiểm soát deadlock

b) Quản lý bộ nhớ chính :

Trong hệ thống máy tính hiện đại, ***bộ nhớ chính*** là trung tâm của các thao tác, xử lý. Bộnhớ chính có thể xem như một mảng kiểu byte hay kiểu word. Mỗi phần tử đều có địa chỉ. Đó là nơi lưu dữ liệu được CPU truy xuất một cách nhanh chóng so với các thiết bị nhập/xuất. CPU đọc những chỉ thị từ bộ nhớ chính. Các thiết bị nhập/xuất cài đặt cơ chế DMA cũng đọc và ghi dữ liệu trong bộ nhớ chính. Thông thường bộ nhớ chính chứa các thiết bị mà CPU có thể định vị trực tiếp. Ví dụ CPU truy xuất dữ liệu từ đĩa, những dữ liệu này được chuyển vào bộ nhớ qua lời gọi hệ thống nhập/xuất.

Một chương trình muốn thi hành trước hết phải được ánh xạ thành địa chỉ tuyệt đối và nạp vào bộ nhớ chính.Khi chương trình thi hành, hệ thống truy xuất các chỉ thị và dữ liệu của chương trình trong bộ nhớ chính. Ngay cả khi tiến trình kết thúc , dữ liệu vẫn còn trong bộ nhớ cho đến khi một tiến trình khác được ghi chồng lên.

Hệ điều hành có những vai trò như sau trong việc quản lý bộ nhớ chính :

- Lưu giữ thông tin về các vị trí trong bộ nhớ đã được sử dụng và tiến trình nào đang sử dụng.

- Quyết định tiến trình nào được nạp vào bộ nhớ chính, khi bộ nhớ đã có thể dùng được.

- Cấp phát và thu hồi bộ nhớ khi cần thiết.

c) Quản lý bộ nhớ phụ :

Bộ nhớ chính quá nhỏ để có thể lưu giữ mọi dữ liệu và chương trình, ngoài ra dữ liệu sẽ mất khi không còn được cung cấp năng lượng. Hệ thống máy tính ngày nay cung cấp ***hệ thống lưu trữ phụ***. Đa số các máy tính đều dùng đĩa để lưu trữ cả chương trình và dữ liệu. Hầu như tất cả chương trình : chương trình dịch, hợp ngữ, thủ tục, trình soạn thảo, định dạng... đều được lưu trữ trên đĩa cho tới khi nó được thực hiện, nạp

vào trong bộ nhớ chính và cũng sử dụng đĩa để chứa dữ liệu và kết quả xử lý. Vai trò của hệ điều hành trong việc quản lý đĩa :

- Quản lý vùng trống trên đĩa.

- Định vị lưu trữ.

- Lập lịch cho đĩa.

d) Quản lý hệ thống vào/ ra :

Một trong những mục tiêu của hệ điều hành là *che dấu* những đặc thù của các thiết bị phần cứng đối với người sử dụng thay vào đó là một lớp thân thiện hơn, người sử dụng dể thao tác hơn. Một hệ thống vào/ra bao gồm :

- Thành phần quản lý bộ nhớ chứa vùng đệm (buffering), lưu trữ (caching) và spooling (vùng chứa).

- Giao tiếp điều khiển thiết bị (device drivers) tổng quát.

- Bộ điều khiển cho các thiết bị xác định.

Chỉ có bộ điều khiển cho các thiết bị xác định mới hiểu đến cấu trúc đặc thù của thiết bị mà nó mô tả.

e) Quản lý hệ thống tập tin :

Máy tính có thể lưu trữ thông tin trong nhiều dạng thiết bị vật lý khác nhau : băng từ, đĩa từ, đĩa quang, ... Mỗi dạng có những đặc thù riêng về mặt tổ chức vật lý. Mỗi thiết bị có một bộ kiểm soát như bộ điều khiển đĩa (disk driver) và có những tính chất riêng. Những tính chất này là tốc độ, khả năng lưu trữ, tốc độ truyền dữ liệu và cách truy xuất.

Để cho việc sử dụng hệ thống máy tính thuận tiện, hệ điều hành cung cấp một cái nhìn logic đồng nhất về hệ thống lưu trữ thông tin. Hệ điều hành định nghĩa một đơn vị lưu trữ logic là tập tin. Hệ điều hành tạo một ánh xạ từ tập tin đến vùng thông tin trên đĩa và truy xuất những tập tin này thông qua thiết bị lưu trữ.

Một tập tin là một tập hợp những thông tin do người tạo ra nó xác định. Thông thường một tập tin đại diện cho một chương trình và dữ liệu. Dữ liệu của tập tin có thể là số, là ký tự, hay ký số.

Vai trò của hệ điều hành trong việc quản lý tập tin :

- Tạo và xoá một tập tin.

- Tạo và xoá một thư mục.

- Hỗ trợ các thao tác trên tập tin và thư mục.

- Ánh xạ tập tin trên hệ thống lưu trữ phụ.

- Sao lưu dự phòng các tập tin trên các thiết bị lưu trữ.

f) Hệ thống bảo vệ :

Trong một hệ thống nhiều người sử dụng và cho phép nhiều tiến trình diễn ra đồng thời, các tiến trình phải được bảo vệ đối với những hoạt động khác.Do đó, hệ thống cung cấp cơ chế để đảm bảo rằng tập tin, bộ nhớ, CPU, và những tài nguyên khác chỉ được truy xuất bởi những tiến trình có quyền. Ví dụ, bộ nhớ đảm bảo rằng tiến trình chỉ được thi hành trong phạm vi địa chỉ của nó. Bộ thời gian đảm bảo rằng không có tiến trình nào độc chiếm CPU. Cuối cùng các thiết bị ngoại vi cũng được bảo vệ.

Hệ thống bảo vệ là một cơ chế kiểm soát quá trình truy xuất của chương trình, tiến trình, hoặc người sử dụng với tài nguyên của hệ thống. Cơ chế này cũng cung cấp cách thức để mô tả lại mức độ kiểm soát.

Hệ thống bảo vệ cũng làm tăng độ an toàn khi kiểm tra lỗi trong giao tiếp giữa những hệ thống nhỏ bên trong.

g) Hệ thống thông dịch lệnh :

Một trong những phần quan trọng của chương trình hệ thống trong một hệ điều hành là hệ thống thông dịch lệnh, đó là giao tiếp giữa người sử dụng và hệ điều hành. Một số hệ điều hành đặt cơ chế dòng lệnh bên trong hạt nhân, số khác như MS-DOS và UNIX thì xem hệ điều hành như

là một chương trình đặt biệt, được thi hành khi các công việc bắt đầu hoặc khi người sử dụng login lần đầu tiên.

Các lệnh đưa vào hệ điều hành thông qua *bộ điều khiển lệnh*. Trong các hệ thống chia xẻ thời gian một chương trình có thể đọc và thông dịch các lệnh điều khiển được thực hiện một cách tự động. Chương trình này thường được gọi là bộ thông dịch điều khiển card, cơ chế dòng lệnh hoặc Shell. Chức năng của nó rất đơn giản đó là lấy lệnh kế tiếp và thi hành.

Mỗi hệ điều hành sẽ có những giao tiếp khác nhau, dạng đơn giản theo cơ chế dòng lệnh, dạng thân thiện với người sử dụng như giao diện của Macintosh có các biểu tượng, cửa sổ thao tác dùng chuột.

Các lệnh có quan hệ với việc tạo và quản lý các tiến trình, kiểm soát nhập xuất, quản lý bộ lưu trữ phụ, quản lý bộ nhớ chính, truy xuất hệ thống tập tin và cơ chế bảo vệ.

2.6. Nhân của hệ điều hành, tải hệ điều hành

Nhân của hệ điều hành (Kernel)

Hệ điều hành là bộ bao gồm một số lượng lớn các chương trình, trong nhiều trường hợp đó là một bộ chương trình đồ sộ và vì vậy không thể đưa tất cả các chương trình của hệ điều hành vào bộ nhớ trong được.

Nhân của hệ điều hành thông thường bao gồm:

-Môdun chương trình tải (Loader). Chức năng chủ yếu của môdun chương trình tải là đưa một chương trình vào bộ nhớ trong bắt đầu từ địa chỉ nào đó để sau đó cho phép chương trình đã được tải nhận điều khiển để chạy hoặc không.

-Môdun chương trình dẫn dắt (monitor). Việc chọn lựa các bước làm việc của toàn bộ hệ thống do môdun này đảm nhiệm.

-Môdun chương trình lập lịch (scheduler): chọn chương trình tiếp theo để chạy.

-Một số môdun chương trình khác

-Cùng một số thông tin hệ thống là các tham số hệ thống.

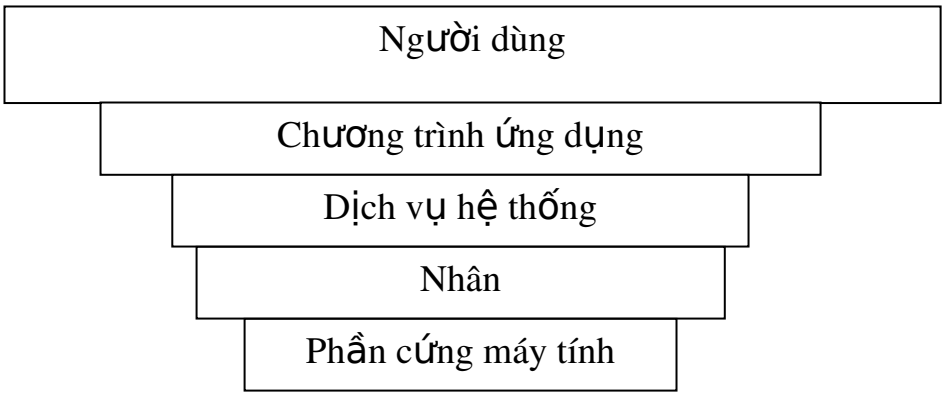
Nảy sinh một số vấn đề liên quan đến nhân hệ điều hành:

Vấn đề (bài toán) đầu tiên liên quan nhân là: chọn những môdun hệ thống nào để đưa vào nhân? Nhân quá lớn (nhân chứa nhiều môdun chương trình) thì đỡ tốn thời gian tải các môdun trong hệ điều hành vào bộ nhớ trong song do chiếm nhiều bộ nhớ trong nên lại giảm dung

lượng bộ nhớ trong có thể sử dụng được cho chương trình của người sử dụng. Nhân quá bé, thì công việc tải sẽ thường xuyên hơn, thời gian dành cho việc nạp môdun của hệ điều hành vào bộ nhớ trong sẽ tăng lên và như thế thời gian CPU dành cho chương trình người dùng giảm đi, hiệu suất sử dụng CPU thấp.

Vấn đề thứ hai đối với nhân là: Phân phối bộ nhớ trong cho nhân như thế nào? Phân phối liên tục hay rời rạc? Một trong những nguyên lý cơ bản của việc nạp nhân là *phân phối bộ nhớ cho nhân phải đảm bảo vùng bộ nhớ liên tục còn lại đạt lớn nhất* có thể có và không gây cản trở cho việc nâng cấp hệ điều hành.

Các mức giao tiếp trong hệ thống máy tính



Hình 1.2. Cấu trúc mức của hệ thống máy tính

Tải hệ điều hành

Hệ điều hành không thể tự đặt ngay trong máy tính được. Do hệ điều hành là tập hợp các chương trình được cài đặt sẵn nhưng ở trên vật dẫn ngoài (các file từ đĩa cứng) và muốn máy tính hoạt động được phải qua một giai đoạn đưa hệ điều hành vào máy để làm việc. Giai đoạn tải hệ điều hành (còn gọi là tải hệ thống) có thể được phân ra các bước sau đây:

Khởi động chương trình tải nguyên thủy. Trong máy tính thường có đoạn chương trình nguyên thủy với tên IPL (Initial Program Loader) đã được cứng hóa (thường đặt trong EPROM) sẽ tự bị kích hoạt để thực hiện mỗi khi bật máy gây xung điện. IPL bắt đầu làm việc.

IPL kiểm tra tính sẵn sàng của hệ thống thiết bị. Tương ứng với mỗi thiết bị, IPL lập ra khối điều khiển thiết bị UCB (Unit Control Block) chứa các thông số về thiết bị đó.

IPL tải đoạn chương trình "mồi", thường đặt ở sector đầu tiên ở đĩa chứa hệ điều hành, vào bộ nhớ trong tại những địa chỉ định sẵn và truyền điều khiển cho đoạn chương trình mồi. Trong một số hệ điều

hành người ta gọi đoạn chương trình mồi là chương trình khởi động nhân. Đoạn chương trình thực hiện chức năng tải nhân của hệ điều hành vào.

Sau khi tải nhân xong, chương trình mồi sẽ trao điều khiển cho chương trình dẫn dắt để hệ thống bắt đầu làm việc.

3. Sơ lược lịch sử phát triển của HĐH

Mục tiêu: nắm được lịch sử phát triển hệ điều hành.

Thế hệ 1 (1945 – 1955)

Vào khoảng giữa thập niên 1940, Howard Aiken ở Havard và John von Neumann ở Princeton, đã thành công trong việc xây dựng máy tính dùng ống chân không. Những máy này rất lớn với hơn 10000 ống chân không nhưng chậm hơn nhiều so với máy rẻ nhất ngày nay.

Mỗi máy được một nhóm thực hiện tất cả từ thiết kế, xây dựng lập trình, thao tác đến quản lý. Lập trình bằng ngôn ngữ máy tuyệt đối, thường là bằng cách dùng bảng điều khiển để thực hiện các chức năng cơ bản. Ngôn ngữ lập trình chưa được biết đến và hệ điều hành cũng chưa nghe đến.

Vào đầu thập niên 1950, phiếu đục lổ ra đời và có thể viết chương trình trên phiếu thay cho dùng bảng điều khiển.

Thế hệ 2 (1955 – 1965)

Sự ra đời của thiết bị bán dẫn vào giữa thập niên 1950 làm thay đổi bức tranh tổng thể. Máy tính trở nên đủ tin cậy hơn. Nó được sản xuất và cung cấp cho các khách hàng. Lần đầu tiên có sự phân chia rõ ràng giữa người thiết kế, người xây dựng, người vận hành, người lập trình, và người bảo trì.

Để thực hiện một công việc (một chương trình hay một tập hợp các chương trình), lập trình viên trước hết viết chương trình trên giấy (bằng hợp ngữ hay FORTRAN) sau đó đục lỗ trên phiếu và cuối cùng đưa phiếu vào máy. Sau khi thực hiện xong nó sẽ xuất kết quả ra máy in.

Hệ thống xử lý theo lô ra đời, nó lưu các yêu cầu cần thực hiện lên băng từ, và hệ thống sẽ đọc và thi hành lần lượt. Sau đó, nó sẽ ghi kết quả lên băng từ xuất và cuối cùng người sử dụng sẽ đem băng từ xuất đi in.

Hệ thống xử lý theo lô hoạt động dưới sự điều khiển của một chương trình đặc biệt là tiền thân của hệ điều hành sau này. Ngôn ngữ lập trình sử dụng trong giai đoạn này chủ yếu là FORTRAN và hợp ngữ.

Thế hệ 3 (1965 – 1980)

Trong giai đoạn này, máy tính được sử dụng rộng rãi trong khoa học cũng như trong thương mại. Máy IBM 360 là máy tính đầu tiên sử dụng mạch tích hợp (IC). Từ đó kích thước và giá cả của các hệ thống máy giảm đáng kể và máy tính càng phổ biến hơn. Các thiết bị ngoại vi dành cho máy xuất hiện ngày càng nhiều và thao tác điều khiển bắt đầu phức tạp.

Hệ điều hành ra đời nhằm điều phối, kiểm soát hoạt động và giải quyết các yêu cầu tranh chấp thiết bị. Chương trình hệ điều hành dài cả triệu dòng hợp ngữ và do hàng ngàn lập trình viên thực hiện.

Sau đó, hệ điều hành ra đời khái niệm ***đa chương***. CPU không phải chờ thực hiện các thao tác nhập xuất. Bộ nhớ được chia làm nhiều phần, mỗi phần có một công việc (job) khác nhau, khi một công việc chờ thực hiện nhập xuất CPU sẽ xử lý các công việc còn lại. Tuy nhiên khi có nhiều công việc cùng xuất hiện trong bộ nhớ, vấn đề là phải có một cơ chế bảo vệ tránh các công việc ảnh hưởng đến nhau. Hệ điều hành cũng cài đặt thuộc tính spool.

Giai đoạn này cũng đánh dấu sự ra đời của ***hệ điều hành chia xẻ thời gian*** như CTSS của MIT. Đồng thời các hệ điều hành lớn ra đời như MULTICS, UNIX và hệ thống các máy mini cũng xuất hiện như DEC PDP-1.

Thế hệ 4 (1980 - nay)

Giai đoạn này đánh dấu sự ra đời của máy tính cá nhân, đặc biệt là hệ thống IBM PC với hệ điều hành MS-DOS và Windows sau này. Bên cạnh đó là sự phát triển mạnh của các hệ điều hành tựa Unix trên nhiều hệ máy khác nhau như Linux. Ngoài ra, từ đầu thập niên 90 cũng đánh dấu sự phát triển mạnh mẽ của ***hệ điều hành mạng*** và ***hệ điều hành phân tán***.

Hệ điều hành tập trung | **Hệ điều hành phân tán** | **Hệ điều hành tự trị phối hợp** | **Hệ điều hành mạng**

Hình 1.3. Phân bố các hệ điều hành

CÂU HỎI VÀ BÀI TẬP

1. Trình bày khái niệm về tài nguyên hệ thống.Cho ví dụ.
2. Nêu khái niệm hệ điều hành.
3. Nêu các chức năng cơ bản hệ điều hành.
4. Các hệ điều hành MSDos, Window XP, Window 7, Linux thuộc loại hệ điều hành nào.
5. Nêu các thành phần của hệ điều hành.

HƯỚNG DẪN TRẢ LỜI

1. Có ba tài nguyên hệ thống: *tài nguyên về phần cứng, tài nguyên về phần mềm và tài nguyên về nguồn nhân lực (ví dụ về phần cứng có CPU,Ram,máy in; về phần mềm:hệ điều hành Window server 2008, chương trình quản lý bán hàng,…)*

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

3. Chức năng hệ điều hành: *quản lý tiến trình, quản lý bộ nhớ, quản lý hệ thống vào-ra, quản lý hệ thống tập tin, hệ thông dịch lệnh.*

4. MSDos là hệ điều hành đơn chương trình; Window XP, Window 7,Linux là hệ điều hành đa chương trình.
5. Các thành phần hệ điều hành: *thành phần điều khiển, thành phần ứng dụng, các chương trình tiện ích.*

CHƯƠNG 2: ĐIỀU KHIỂN DỮ LIỆU

Chương 2: Điều khiển dữ liệu **Mã chương**:..M2

Mục tiêu:

Sau khi học xong bài học này, sinh viên có khả năng:

- Nắm được cách thức hệ điều hành (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 vào – ra dữ liệu
- Nắm được cách thức tổ chức lưu trữ dữ liệu, các phương pháp quản lý trên bộ nhớ ngoài.

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. Các phương pháp tổ chức dữ liệu

Khái niệm file (bộ dữ liệu)

Dữ liệu được xử lý trong máy tính được bảo quản lâu dài trên băng từ, đĩa từ, đĩa quang v.v… và dữ liệu được tập hợp lại một cách có tổ chức thành các file dữ liệu theo mục đích sử dụng. File có thể là chương trình của người dùng, một chương trình của hệ điều hành, một văn bản, một tập hợp dữ liệu. Trong một số hệ điều hành, một số thiết bị ngoại vi cũng được quan niệm như file dữ liệu.

Theo góc độ quan sát của người dùng, dữ liệu trong một file lại được tổ chức thành các bản ghi lôgic (gọi tắt bản ghi), mà mỗi bản ghi lôgic có thể là một byte hoặc một cấu trúc dữ liệu nào đó. Bản ghi chính là đơn vị dữ liệu mà chương trình người dùng quan tâm đến và xử lý theo mỗi nhịp làm việc: file là tập hợp (được người dùng quan niệm là một dãy) các bản ghi có tổ chức. Thông thường, trong file tồn tại một thứ tự

giữa các bản ghi, thứ tự đó thể hiện vị trí logic giữa các bản ghi với nhau (chẳng hạn như thứ tự đưa bản ghi vào file).

Phụ thuộc vào mục đích sử dụng (sắp xếp, tìm kiếm…bản ghi trong file), người ta đưa ra bốn kiểu tổ chức phổ biến với file. Đó là tổ chức kế tiếp, tổ chức chỉ số kế tiếp, tổ chức thư viện và tổ chức trực tiếp. Ngoài ra có thể kể đến việc tổ chức file trong chế độ bộ nhớ ảo trên đĩa từ.

Tổ chức kế tiếp

Bản ghi lôgic sắp xếp đúng theo trình tự làm việc đối với chúng: thứ tự trình bày trên vật dẫn ngoài trùng với thứ tự đưa bản ghi vào trong file. Tổ chức kế tiếp chính là kiểu tổ chức duy nhất đối với các file đặt trên bìa đục lỗ, băng từ v.v… Rõ ràng là đối với file gồm các bản ghi trên chồng bìa hay băng từ, việc đi đến một bản ghi nào đó phải vượt qua các bản ghi được xếp trước nó. Tuy nhiên, trên các thiết bị trực truy như đĩa từ, đĩa quang cũng cho phép file được tổ chức kế tiếp.

Tổ chức chỉ số kế tiếp

Trong nhiều trường hợp, việc sắp xếp và tìm kiếm các bản ghi không theo thứ tự phát sinh ra chúng, mà theo một thứ tự nào đó gắn với bản ghi: mỗi bản ghi được tương ứng với một chỉ số và việc sắp xếp, tìm kiếm bản ghi theo chỉ số. Liên quan đến file được tổ chức theo chỉ số kế tiếp có một hệ thống chỉ số bao gồm chỉ số chính, chỉ số trụ, chỉ số rãnh.

Hệ thống chỉ số: để làm việc với file tổ chức chỉ số kế tiếp cần có hệ thống chỉ số gắn với file đó. Một file có thể được lưu trữ trên nhiều đĩa từ, và vì vậy, chỉ số chính cho biết trên mỗi đĩa từ chứa dữ liệu của file bao gồm các bản ghi có các chỉ số thuộc khoảng nào.

Bảng dưới đây biểu diễn một bảng chỉ số chính đối với một file được tổ chức theo dạng chỉ số kế tiếp.

Tên đĩa	Chỉ số max
VOL1	1000
VOL2	2000
VOL3	4000
….	….

Theo bảng này, các bản ghi có chỉ số không vượt quá 1000 nằm trong đĩa có tên là VOL1, các bản ghi có chỉ số từ 1001 tới 2000 nằm trên đĩa có tên VOL2,…Việc tìm kiếm bản ghi không tiến hành lần lượt qua từng bản ghi mà được hạn chế theo không gian chỉ số: bước đầu tiên xác định đĩa từ chứa bản ghi đó và đi tới tìm kiếm trên đĩa từ đó.

Với cấu trúc tương tự nhằm mục đích định vị dần được bản ghi theo chỉ số, trên mỗi đĩa chứa file lại có một bảng chỉ số trụ, cho biết mỗi trụ chứa các bản ghi thuộc khoảng chỉ số nào.

Thấp hơn nữa, trên mỗi trụ lại có một bảng chỉ số rãnh cho biết cụ thể các chỉ số trên mỗi rãnh thuộc trụ nói trên.

Tổ chức truy nhập trực tiếp (trực truy)

Trong file được tổ chức trực truy, tồn tại sự tương ứng giữa định vị bản ghi của file với địa chỉ thực sự trên đĩa từ mà không phải qua một hệ thống chỉ số nào cả. Trong quá trình làm việc với file dữ liệu, người dùng chủ động làm việc theo sự tương ứng nói trên và tổ chức trực truy đảm bảo cho người dùng khả năng linh hoạt trong xử lý bản ghi.

Tổ chức thư viện

File tổ chức thư viện bao gồm một thư mục và một tập hợp file thành phần mà mỗi file thành phần lại được tổ chức kế tiếp. Mỗi file thành phần được gọi là một chương và mỗi chương có tên để truy nhập đến.

Ví dụ về file được tổ chức thư viện có thể kể đến như các file thư viện (đuôi .LIB) trong các ngôn ngữ lập trình, mỗi một môdun chương trình mẫu như một chương của file thư viện.

Tổ chức theo bộ nhớ ảo

Trong chế độ phân chia thời gian phải sử dụng bộ nhớ ảo. Các file có liên quan đến bộ nhớ ảo được tổ chức theo những quy cách riêng, tiện lợi cho việc luân chuyển, trao đổi giữa bộ nhớ thực và bộ nhớ ảo.

Chú ý rằng, khi thiết kế hệ điều hành, với mỗi phương pháp tổ chức file, các chuyên gia tạo ra hệ điều hành phải xây dựng được các môdun chương trình tương ứng. Nếu một hệ điều hành cho phép có nhiều phương pháp tổ chức file có thể làm cho việc tạo lập và thi hành hệ điều hành trở nên quá phức tạp. Vì vậy, một hệ điều hành không cần thiết phải có tất cả các cách chức tổ chức file nêu trên.

1.2. Các phương pháp truy nhập dữ liệu

Tồn tại hai cách thức truy nhập dữ liệu phổ biến nhất: truy nhập tuần tự và truy nhập cơ sở.

Cách thức truy cập tuần tự

Lần lượt các bản ghi theo đúng trình tự trong file (từ bản ghi đầu tiên đến bản ghi cuối cùng) được "xem xét" và "xử lý". Theo cách thức truy nhập tuần tự thì hoàn toàn biết trước được bản ghi lôgic tiếp theo được xem xét xử lý là bản ghi nào, và vì vậy, hệ điều hành biết được vị trí trên vật dẫn ngoài của bản ghi cần xử lý tiếp theo. Khi đã biết được vị trí, để có sẵn bản ghi tiếp theo cho chương trình xử lý, cách tốt nhất là đọc trước bản ghi cần xử lý vào bộ nhớ trong.

Cách thức truy nhập tuần tự cho mức độ tự động hóa cao, tuy nhiên chỉ áp dụng được với các file được tổ chức kế tiếp hoặc chỉ số kế tiếp.

Để đảm bảo được tính tự động hóa cao như thế, chương trình hệ thống phải đảm bảo thực hiện mọi công việc chuẩn bị liên quan đến bản ghi cho chương trình người dùng xử lý.

Cách thức truy nhập cơ sở

Theo cách thức truy nhập cơ sở, hệ thống hoàn toàn không có trước thông tin về bản ghi nào là bản ghi tiếp theo để xử lý nên mức độ tự động hóa thấp: Người lập trình tự mình phải xác định bản ghi cần xử lý, và để tìm được nó, mọi vấn đề về đồng bộ hóa phải được đặt ra.

Tuy mức độ tự động hóa thấp, nhưng bù lại, cách thức truy nhập cơ sở cho chương trình người sử dụng làm việc với file dữ liệu hết sức mềm dẻo và linh hoạt, đạt được mức độ chủ động cao của chương trình người dùng đối với file.

Các phương pháp tổ chức và truy nhập dữ liệu phổ biến

Trong hệ điều hành cần có các chương trình làm việc với các kiểu tổ chức file và cách thức truy nhập dữ liệu. Giải pháp tạo ra một chương trình của hệ điều hành làm việc phù hợp đối với tất cả các kiểu tổ chức và truy nhập file là không thực tiễn (vì làm như vậy quá trình vào ra dữ liệu trở nên rất phức tạp, không linh hoạt) vì vậy phương án thích hợp hơn là chia ra một số phương pháp tổ chức và truy nhập dữ liệu cụ thể, đối với mỗi phương pháp này có chương trình riêng phục vụ nó. Có sáu phương pháp tổ chức truy nhập dữ liệu cụ thể phổ biến, đó là:

Phương pháp truy nhập tuần tự cho file tổ chức kế tiếp (QSAM: Queue Sequel Access Method)

Phương pháp truy nhập cơ sở cho File tổ chức kế tiếp (BSAM)

Phương pháp truy nhập tuần tự cho file tổ chức chỉ số kế tiếp (QISAM:Queue Index Sequel Access Method)

Phương pháp truy nhập cơ sở cho file tổ chức chỉ số kế tiếp (BISAM)

Phương pháp truy nhập cơ sở cho file tổ chức trực tiếp (BDAM: Basic Direct Access Method);

Phương pháp truy nhập cơ sở cho file tổ chức thư viện (BPAM: Basic Partition Access Method).

1.3 Chức năng của hệ thống điều khiển dữ liệu

Các chức năng cơ bản của hệ thống điều khiển dữ liệu bao gồm các chức năng liên quan đến việc tổ chức bảo quản dữ liệu trên vật dẫn ngoài và truy nhập chúng để chương trình người dùng thao tác. Dưới đây sẽ trình bày chi tiết hơn về các chức năng của hệ thống điều khiển dữ liệu.

Bảo quản dữ liệu trên vật dẫn ngoài

Trên vật dẫn ngoài, mà phổ dụng nhất là trên băng từ và đĩa từ, các file dữ liệu (và chương trình) được lưu trữ để sau này có thể dễ dàng làm việc với chúng. Việc lưu trữ các file theo những quy định chặt chẽ của hệ điều khiển dữ liệu để không ảnh hưởng lẫn nhau, không chồng chéo. Như vậy, trên vật dẫn ngoài, việc tổ chức thông tin phải theo những quy định của hệ điều hành. Tồn tại một số phương pháp quản lý vùng nhớ trên vật dẫn ngoài để phân phối cho các file (sử dụng bảng FAT, sử dụng Bitmap, …). Thực tế là không gian trên vật dẫn ngoài không dành cho việc lưu trữ nội dung của các file mà còn phải có các vùng không gian để chứa thông tin liên quan đến đặc trưng của vật mang tin và tình trạng phân phối không gian cho các file trên vật mang tin đó. Trên đĩa từ, các hệ điều hành thường cung cấp cho người dùng một hệ thống phân cấp dạng cấu trúc cây tổ chức các file có trong đĩa đó. Ngoài khái niệm file, hệ thống còn sử dụng các thư mục cho việc quản lý lưu trữ file.

Đảm bảo cách thức tổ chức khác nhau đối với dữ liệu và định vị chúng

Như đã nói, mỗi cách tổ chức dữ liệu kèm theo những mặt mạnh và mặt yếu riêng và phù hợp với mục tiêu sử dụng nhất định. Người dùng mong muốn file dữ liệu mà mình làm việc được tổ chức theo cách thức mà người đó mong muốn. Hệ điều hành cần đảm bảo việc lưu trữ file trên vật dẫn ngoài phù hợp với cách tổ chức nội tại (các bản ghi lôgic có trong file) nhằm thỏa mãn yêu cầu phong phú của người dùng.

Thực hiện các phương pháp truy nhập khác nhau tới dữ liệu phụ thuộc vào phương pháp tổ chức chúng

Mỗi phương pháp truy nhập bao gồm các môdun chương trình liên quan đến việc tổ chức nội tại của file và cách thức truy nhập của chúng. Khi cho phép các cách tổ chức khác nhau đối với file trên vật dẫn ngoài đồng thời cần cho phép chúng được truy nhập theo các phương pháp phong phú. Ngay cả đối với một file trên vật dẫn ngoài, tại thời điểm này người dùng có thể sử dụng cách thức QSAM song một lúc khác, người dùng có thể sử dụng cách thức BDAM (nếu điều đó có thể được). Chức năng này đòi hỏi hệ điều hành xử lý mềm dẻo theo yêu cầu người dùng.

Catalog dữ liệu và thực hiện việc tìm kiếm tự động hóa dữ liệu đã được catalog theo tên ký hiệu mà không cần theo địa chỉ

Trong nhiều hệ điều hành, các file có thể được catalog hóa. Những file thường xuyên được sử dụng (các chương trình ứng dụng, các file dữ liệu người dùng v.v…) nếu được catalog hóa thì việc tìm kiếm chúng khá dễ dàng. Trong một đĩa từ đặc biệt, được gọi là đĩa thường trực, có bảng catalog: bảng này cho thấy sự tương ứng giữa tên một file đã catalog hóa với địa chỉ tìm kiếm trực tiếp nó (tên đĩa từ chứa file đó, vị trí cụ thể đặt

file nói trên trên đĩa từ). Mỗi khi làm việc với một file đã catalog, hệ thống yêu cầu đặt đĩa từ có tên xác định vào và tự động tìm ngay được file cho người dùng.

Cho phép sự độc lập cao nhất của chương trình đối với dữ liệu của chương trình đó

Việc lưu trữ các file trên vật dẫn ngoài do hệ thống tự động đặt, không phụ thuộc vào chương trình người dùng. Điều đó cho phép sự độc lập cao nhất có thể có giữa chương trình và dữ liệu. Con người không quan tâm đến việc hệ điều hành đã đặt file ở đâu trên vật dẫn ngoài, chỉ quan tâm đến tên của nó. Sự độc lập nói trên còn cho phép nhiều người dùng có thể sử dụng chung tài nguyên dữ liệu và như vậy, giá trị của dữ liệu vì thế càng được nâng cao.

2. Bản ghi và khối

Mục tiêu: nắm được khái niệm bản ghi logic và bản ghi vật lý;

Nắm được khái niệm kết khối và tách khối.

Vận dụng được để đưa ra ví dụ cụ thể của quá trình kết khối và tách khối trong hệ điều hành.

2.1. Bản ghi lôgic và bản ghi vật lý

a.Bản ghi lôgic 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 lôgic. 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 được gọi là Khối). Thông thường, khối đơn vị bộ nhớ ngoài mà hệ điều hành thực hiện việc đọc/ghi đối với file. Chẳng hạn, trong MS DOS, một cluster chính là một khối trên đĩa từ và file được lưu trữ trên một tập hợp các cluster 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 nắm 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 số 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 cả 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 tương ứng.

b. Bản ghi theo tổ chức của File: có ba dạng tổ chức bản ghi lôgic

Thông thường có ba dạng phổ biến là dạng cố định, dạng động và dạng không xác định. Dạng 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 liệu 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 ghi trong một file có kiểu trong ngôn ngữ lập trình PASCAL thuộc 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.

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 và V.

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 vào ra vậy lý.

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

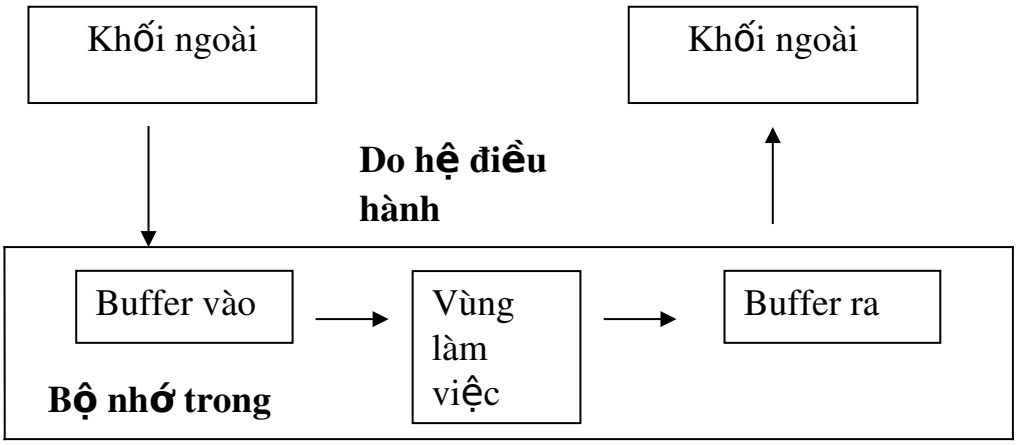
Tách khối là quá trình từ các khối đưa ra được các bản ghi cần tìm có liên quan đến khối đó. Quá trình này diễn ra sau khi hệ điều hành đã đọc

một khối từ vật dẫn ngoài 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 lâu dài. Như đã biết, hệ điều hành ghi thông tin lên thiết bị nhớ 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 thiết bị nhớ ngoài.

Chương trình người dùng xử lý dữ liệu tại 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 gian được gọi là vùng đệm (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.



Hình 2.1 Sơ đồ tách khối / kết khối

Sơ đồ trong hình 2.1 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 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.

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

(môdun chương trình có thể được phát triển thành nhóm môdun chương trình):

-Môdun chương trình đảm bảo chức năng tổ chức lưu trữ và định vị trên vật dẫn ngoài;

-Môdun chương trình đảm bảo vào/ra mỗi khối (bản ghi vật lý) đối với mỗi khối xác định;

-Môdun chương trình đảm bảo việc tách/kết khối theo bản ghi đối với file xác định.

3. Điều khiển buffer

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) và chương trình người dùng trong quá trình vào – ra dữ liệu

3.1. Vai trò của buffer

Như đã trình bày trong mục 2.2, quá trình vào – ra luôn cần các vùng nhớ trung gian làm nơi đặt nội dung các bản ghi vật lý (khối). Tại các vùng nhớ nói trên sẽ diễn ra các quá trình tách khối (khi đọc dữ liệu từ ngoài vào) hay kết khối (khi ghi dữ liệu lên vật dẫn ngoài). Như vậy, buffer chính là vùng bộ nhớ trong lưu trữ tạm thời các dữ liệu, thuận tiện cho việc vào-ra.

Chương trình người dùng có thể làm việc với một hoặc nhiều file ngoài, tốc độ xử lý của chương trình và tốc độ đọc dữ liệu của chương trình của phương pháp truy nhập là nhanh chậm khác nhau và trong nhiều trường hợp để hiệu quả hơn trong việc vào – ra ,các buffer có thể được liên kết nhau và tạo thành một xâu các buffers.

Tùy thuộc vào chương trình người dùng được viết trên ngôn ngữ lập trình nào, mà vùng nhớ đệm được tạo ra hoặc do chương trình dịch hoặc do chính chương trình người dùng. Nếu chương trình được viết trên ngôn ngữ bậc cao thì do chương trình dịch đảm nhận, còn nếu nó được viết trên assembler thì do chính chương trình người dùng phải đảm nhận.

Trong một số hệ điều hành còn có quy định về số lượng cực đại các buffer có thể được dùng trong hệ thống;mặt khác, thông tin liên quan đến các buffer nói trên được đặt vào các vùng nhớ đã được định sẵn (liên hệ với dòng lệnh buffers = n của CONFIG.SYS trong MS DOS).

3.2. Sử dụng buffers

Có hai phương pháp điển hình khi sử dụng buffer: sử dụng buffer theo khẳng định và sử dụng buffer theo đòi hỏi.

a.Buffer theo khẳng định

Áp dụng cho các file dữ liệu được mở để làm việc theo hai phương pháp truy nhập QSAM và QISAM: theo hai phương pháp này, chương trình của phương pháp truy nhập đã biết trước bản ghi cần xử lý và vì vậy mức độ tự động hóa cao, tốc độ nhanh.

Mức độ tự động hóa cao được thể hiện ở chỗ mọi khâu tách khối, kết khối, đồng bộ hóa, kiểm tra sai sót dều do chương trình hệ thống đảm nhận, lệnh vào ra của chương trình người dùng chỉ thực hiện công việc hết sức đơn giản và do đó đạt tốc độ làm việc nhanh.

Lệnh vào ra chương trình người dùng (do chương trình dịch ra) chỉ làm công việc truyền dữ liệu từ vùng nhớ này (từ buffer) sang vùng làm việc (khi vào) và theo hướng ngược lại (khi ra).

Sử dụng buffer theo khẳng định tương ứng với cách thức truy nhập file tuần tự. Ngay lệnh mở file để đọc, khối đầu tiên của file đã được đọc vào bộ nhớ và bản ghi đầu tiên đã được tách ra sẵn sàng đáp ứng yêu cầu của chương trình người dùng. Sau khi bản ghi được xử lý xong (bao gồm cả trường hợp bản ghi được tạo mới), vị trí của nó hoàn toàn đã được biết, và vì vậy, nó sẽ được kết khối để chuẩn bị đưa ra bộ nhớ ngoài.

b. Buffer theo đòi hỏi

Sử dụng buffer theo đòi hỏi được dùng đối với mọi phương pháp truy nhập dữ liệu. Trong chế độ này, người dùng xác định chương trình của mình sẽ chủ động làm việc với bản ghi nào vì vậy hệ điều hành không thể tự động đọc (ghi) khối tương ứng vào (từ) bộ nhớ trong được. Chỉ sau khi người sử dụng đã đưa ra yêu cầu làm việc với khối nào thì chương trình hệ thống mới vào ra vật lý với khối đó. Mọi công việc tách khối, kết khối, kiểm tra tính đúng đắn của thao tác vào ra, đồng bộ hóa các công việc đều do chương trình người dùng phải đảm nhận.

Tuy rằng mức độ tự động hóa thấp, song khi sử dụng buffer theo đòi hỏi, sử chủ động của chương trình người dùng đối với dữ liệu lại cao hơn cách thức sử dụng buffer theo khẳng định và quan trong hơn là nó không bị hạn chế phạm vi sử dụng như buffer theo khẳng định.

3.3. Điều khiển buffer (vào ra dữ liệu)

Trong sơ đồ vào ra, chúng ta đã được giới thiệu về vùng làm việc, đó là vùng bộ nhớ mà chương trình người dùng trực tiếp xử lý dữ liệu trên đó. Tuy nhiên, trong nội dung của phần dưới đây, các buffer có thể đóng vai trò của vùng làm việc. Điều khiển buffer cho biết cách thức mà chúng ta sẽ sử dụng các buffer đó.

Đối với buffer theo khẳng định tồn tại hai phương pháp sử dụng buffer: buffer đơn giản và buffer trao đổi. Buffer theo khẳng định làm việc với lệnh GET (khi vào) và PUT (khi ra).

Buffer theo đòi hỏi làm việc với lệnh READ (khi vào) và WRITE (khi ra) ngoài ra đòi hỏi các lệnh CHECK (kiểm tra sự kiện) và WAIT (chờ đợi một sự kiện).

a. Buffer đơn giản

Trong buffer đơn giản, các đoạn (tương ứng với một đoạn làm việc: bản ghi lôgic) trong buffer là kề cận nhau và luôn liên quan tới cùng một file. Trong buffer đơn giản, hệ thống sử dụng cùng một lệnh kênh đối với mọi buffer trong xâu buffer. Bản ghi có thể được xử lý hoặc tại miền làm việc, hoặc tại buffer vào, hoặc tại buffer ra. Phương pháp sử dụng buffer đơn giản lại được chia ra một số chế độ sử dụng là chế độ gửi, chế độ dữ liệu, chế độ chỉ dẫn.

Chế độ gửi

Khi vào: theo lệnh GET, bản ghi lôgic lần lượt được gửi từ buffer vào tới vùng làm việc để chương trình xử lý. Động từ "gửi" dùng để chỉ tồn tại thực sự việc gửi dữ liệu từ buffer vào tới vùng làm việc (buffer vào và vùng làm việc là hai vùng nhớ khác nhau hoàn toàn).

Khi ra: theo lệnh PUT, bản ghi lôgic lần lượt từ vùng làm việc chuyển tới buffer ra. Tương tự khi vào dữ liệu, quá trình chuyển dữ liệu từ vùng làm việc tới buffer thực sự được xảy ra.

Chế độ dữ liệu (chỉ áp dụng phương pháp QSAM)

Đối với bản ghi có độ dài mở rộng (trong trường hợp này một bản ghi lôgic chưa nhiều bản ghi vật lý, trên mỗi bản ghi vật lý, ngoài thông tin dữ liệu thực sự lại có thêm các thông tin điều khiển liên quan đến sự liên kết các bản ghi vật lý trong bản ghi lôgic đó). Quá trình hoạt động trong chế độ dữ liệu tương tự như trong chế độ gửi, ngoại trừ việc không truyền gửi các thông tin liên quan đến việc mô tả bản ghi.

Chế độ chỉ dẫn

Không dùng miền làm việc: lấy ngay buffer vào hay buffer ra làm vùng làm việc.

Theo lệnh GET, địa chỉ của bản ghi lôgic tiếp theo trong buffer vào được trao cho chương trình để buffer vào đóng vai trò của vùng làm việc. Như vậy lệnh GET chỉ chuyển giao địa chỉ của đoạn buffer vào cho chương trình người dùng để chương trình người dùng xử lý trên vùng có địa chỉ đã truyền (thực chất chương trình người dùng xử lý trên buffer vào).

Theo lệnh PUT, bản ghi cũng không được gửi: địa chỉ của vùng làm việc (chương trình vừa xử lý) đó trở thành địa chỉ đoạn của buffer ra.

b. Buffer trao đổi

Trong buffer trao đổi, các đoạn trong buffer không nhất thiết kề cận nhau, ngoài ra tất cả các đoạn có thể liên kết với các File khác nhau. Miền

làm việc phải tương thích về độ dài và giới hạn như buffer vào. Buffer ra phải tương thích buffer vào về kích cỡ và giới hạn, điều đó cho phép thay đổi vai trò của buffer vào, buffer ra và vùng làm việc.

Buffer trao đổi có cả ba chế độ điều khiển buffer (gửi , dữ liệu, chỉ dẫn) như buffer đơn giản.

Ngoài ra, sử dụng buffer trao đổi còn có chế độ đặt: chế độ đặt cũng giống như chế độ gửi.

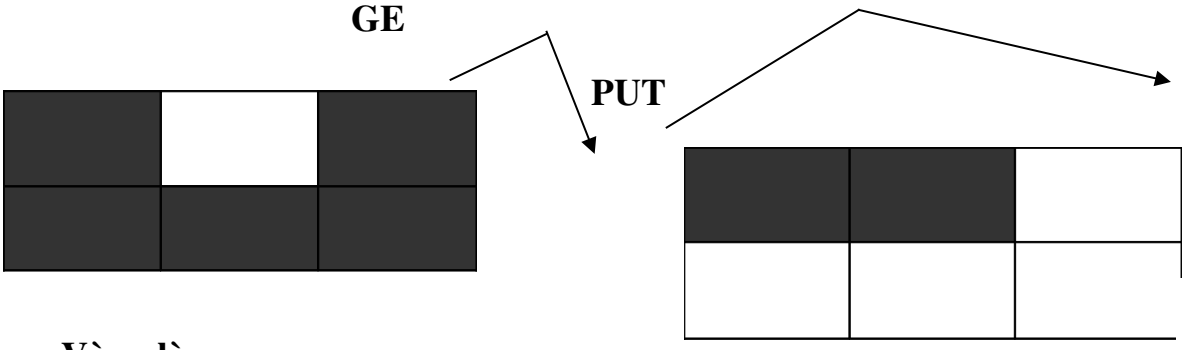
Vai trò của ba đối tượng vùng làm việc, buffer vào, buffer ra là bình đẳng.

c. Buffer theo đòi hỏi

Buffer theo đòi hỏi làm việc theo chế độ trực tiếp: trước mỗi lệnh READ, WRITE, phải thiết lập được buffer rỗi trong xâu buffer. Theo lệnh READ, khối từ bộ nhớ ngoài (được xác định trong lệnh READ) được tải vào buffer nói trên. Để đồng bộ hóa phải sử dụng lệnh CHECK và WAIT trong chương trình người dùng và như vậy người lập trình phải đảm bảo chương trình của mình hoạt động chính quy.

d. Một số ví dụ trong điều khiển Buffer

GET ở chế độ gửi, PUT ở chế độ gửi (Hình 2.2)



Hình 2.2 Điều khiển buffer GET gửi (vào) và PUT gửi (ra)

4. Quy trình điều khiển chung vào ra

Mục tiêu: Nắm được quy trình điều khiển vào ra.

4.1 Các khối điều khiển dữ liệu

Các chương trình hệ thống phải quản lý được các thông tin về các File đang làm việc trong hệ thống, cách thức truy nhập chúng, thông tin về vật dẫn ngoài chứa nội dung các File đó. Để có thể thực hiện được các chức năng của mình, hệ thống xây dựng (hoặc đòi hỏi) một số khối điều khiển dữ liệu. Các khối điều khiển dữ liệu điển hình được giới thiệu ở dưới đây.

Khối FCB (File Control Block): chứa thông tin quản lý làm việc đối với File. Trong một số hệ điều hành thuật ngữ "thẻ File" có ý nghĩa thay

thế tương đương. Trong khối này có những thông tin cụ thể về File tương ứng: số lượng bản ghi, bản ghi hiện thời, địa chỉ các khối liên kết v.v…

Khối DCB (Data Control Block): chương trình người dùng được viết theo ngôn ngữ bậc cao thì chương tình dịch tạo DCB, còn nếu được viết theo hợp ngữ thì chương trình người dùng tạo DCB. Khối DCB chứa mọi thông tin liên quan đến điều khiển vào ra: tổ chức File, phương pháp truy nhập, địa chỉ các khối điều khiển liên quan v.v…

Khối UCB (Unit Control Block): chứa thông tin về thiết bị vào ra, vật dẫn ngoài tương ứng, giúp cho quá trình điều khiển thiết bị.

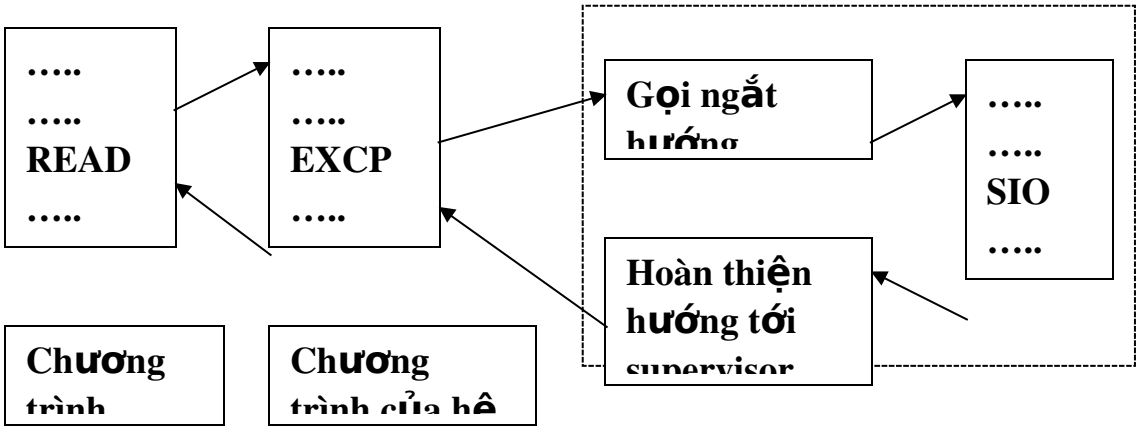
Ngoài ra còn có một số khối mở rộng khác cho điều khiển dữ liệu.

4.2 Một số ví dụ về sơ đồ chung điều khiển vào ra trong OS (sử dụng lệnh READ: buffer theo đòi hỏi)

Qua xem xét sơ đồ ở hình 2.5 chúng ta thấy:

Chương trình người dùng và chương trình của phương pháp truy nhập ở vùng bộ nhớ RAM (địa chỉ của chúng tùy theo trạng thái máy trước khi chúng được nạp vào).

Các chương trình gọi ngắt vào ra, thân ngắt và kết thúc ngắt được đặt trên những địa chỉ xác định. Trong thân ngắt có chứa lệnh bắt đầu vào/ra(SIO: start Input/Output). Như đã biết điều khiển vào ra do kênh đảm nhận và kênh hoạt động theo hệ thống lệnh riêng (lệnh kênh).



(*Trong sơ đồ trên, vùng nằm trong đường rời nét là thuộc các vùng nhớ cố định của bộ nhớ trong)*

Hình 2.3. Một ví dụ về điều khiển vào – ra trong hệ điều hành OS

Chi tiết quá trình được tóm tắt như sau (xét các máy theo hệ OS):

-chuẩn bị một chương trình kênh (dãy các lệnh kênh)

-xây dựng từ địa chỉ kênh (CAW: channel address Word)

-gửi từ địa chỉ kênh nói trên vào một địa chỉ quy định sẵn

-đưa ra lệnh SIO và tải chương trình kênh (theo kênh và thiết bị tương ứng)

-phân tích kết quả việc tải chương trình kênh

-sau khi tải thành công chương trình kênh, CPU và kênh làm việc song song

-sau khi kết thúc (tốt hay không tốt) công việc vào ra, kênh đưa ra tín hiệu cho ngắt vào/ra. Chương trình xử lý ngắt sẽ phân tích tín hiệu trên để biết thành công hay không và dấu hiệu sai sót.

Chương trình người dùng, dựa vào kiểm tra kết quả vào/ra để xử lý: nếu hoàn thiện thì công việc tiếp tục; nếu có sai sót sẽ tùy từng ngữ cảnh để xử lý.

Nếu chỉ ra rằng, thao tác vào ra không thể kết thúc ngay được thì chương trình sẽ chuyển sang trạng thái chờ đợi.

5. Tổ chức lưu trữ dữ liệu trên bộ nhớ ngoài

Mục tiêu: Nắm được cách thức tổ chức lưu trữ dữ liệu, các phương pháp quản lý trên bộ nhớ ngoài.

5.1. Các khái niệm cơ bản

Yêu cầu quản lý bộ nhớ ngoài

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

Quản lý không gian nhớ tự do trên bộ nhớ ngoài (free space manage)

Cấp phát không gian nhớ tự do (allocation methods)

Cung cấp các khả năng định vị bộ nhớ ngoài

Lập lịch cho bộ nhớ ngoài

Cấu trúc vật lý đĩa từ

Đĩa từ bao gồm một hoặc nhiều lá đĩa đặt đồng trục. Mỗi mặt đĩa chia thành các rãnh tròn đồng tâm gọi là track, mỗi track được chia thành các cung gọi là sector. Trên mỗi mặt đĩa có đầu đọc ghi dữ liệu.

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

5.2. Các phương pháp quản lý không gian tự do

Vì không gian trống là giới hạn nên chúng ta cần dùng lại không gian từ các tập tin bị xoá cho các tập tin mới nếu có thể. Để giữ vết của không gian đĩa trống, hệ thống duy trì một danh sách không gian trống. Danh sách không gian trống ghi lại tất cả khối đĩa trống. Để tạo

tập tin, chúng ta tìm trong danh sách không gian trống lượng không gian được yêu cầu và cấp phát không gian đó tới tập tin mới. Sau đó, không gian này được xoá từ danh sách không gian trống. Khi một tập tin bị xoá, không gian đĩa của nó được thêm vào danh sách không gian trống. Mặc dù tên của nó là danh
sách nhưng danh sách không gian trống có thể không được cài như một danh sách.

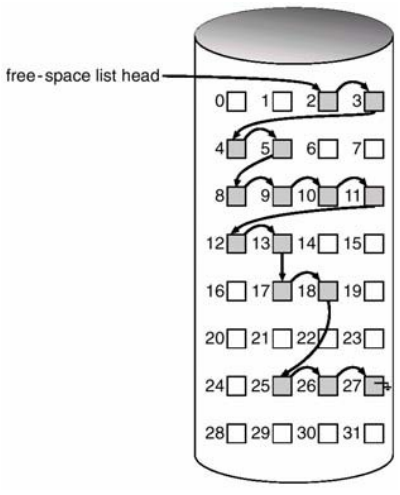
a) Bit vector

Thường thì danh sách không gian trống được cài đặt như một bản đồ bit (bit map) hay một vector bit (bit vector). Mỗi khối được biểu diễn bởi 1 bit. Nếu khối là trống, bit của nó được đặt là 1, nếu khối được cấp phát bit của nó được đặt là 0.
Thí dụ, xét một đĩa khi các khối 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, và 27 là trống và các khối còn lại được cấp phát. Bản đồ bit không gian trống sẽ là:
00111100111111000110000001110000 0…

Lợi điểm chính của tiếp cận này là tính tương đối đơn giản và hiệu quả của nó trong việc tìm khối trống đầu tiên, hay n khối trống tiếp theo trên đĩa.

Một lần nữa, chúng ta thấy các đặc điểm phần cứng định hướng chức năng phần mềm. Tuy nhiên, các vector bit là không đủ trừ khi toàn bộ vector được giữ
trong bộ nhớ chính. Giữ nó trong bộ nhớ chính là có thể cho các đĩa nhỏ hơn, như trên các máy vi tính nhưng không thể cho các máy lớn hơn. Một đĩa 1.3 GB với khối 51 bytes sẽ cần một bản đồ bit 332 KB để ghi lại các khối trống. Gom bốn khối vào một nhóm có thể giảm số này xuống còn 83 KB trên đĩa.

b) Danh sách liên kết



Hình 2.4 danh sách không gian trống được liên kết trên đĩa

Một tiếp cận khác để quản lý bộ nhớ trống là liên kết tất cả khối trống, giữ một con trỏ tới khối trống đầu tiên trong một vị trí đặc biệt trên đĩa và lưu nó trong bộ nhớ. Khối đầu tiên này chứa con trỏ chỉ tới khối đĩa trống tiếp theo,..Trong thí dụ trên, chúng ta có thể giữ một con trỏ chỉ tới khối 2 như là khối trống đầu tiên. Khối 2 sẽ chứa một con trỏ chỉ tới khối 3, khối này sẽ chỉ tới khối 4,…(như hình X-10). Tuy nhiên, cơ chế này không hiệu quả để duyệt danh sách, chúng ta phải đọc mỗi khối,
yêu cầu thời gian nhập/xuất đáng kể. Tuy nhiên, duyệt danh sách trống không là hoạt động thường xuyên. Thường thì, hệ điều hành cần một khối trống để mà nó có thể cấp phát khối đó tới một tập tin, vì thế khối đầu tiên trong danh sách trống được dùng. Phương pháp FAT kết hợp với đếm khối trống thành cấu trúc dữ liệu cấp phát.

c) Nhóm

Thay đổi tiếp cận danh sách trống để lưu địa chỉ của n khối trống trong khối trống đầu tiên. n-1 khối đầu tiên này thật sự là khối trống. Khối cuối cùng chứa địa chỉ của n khối trống khác, …Sự quan trọng của việc cài đặt này là địa chỉ của một số lượng lớn khối trống có thể được tìm thấy nhanh chóng, không giống như trong tiếp cận danh sách liên kết chuẩn.

d) Bộ đếm

Một tiếp cận khác đạt được lợi điểm trong thực tế là nhiều khối kề có thể được cấp phát và giải phóng cùng lúc, đặc biệt khi không gian được cấp phát với giải thuật cấp phát kề hay thông qua nhóm. Do đó, thay vì giữ một danh sách n địa chỉ đĩa trống, chúng ta có thể giữ địa chỉ của khối trống đầu tiên và số n khối kề trống theo sau khối đầu tiên. Mỗi mục từ trong danh sách không gian trống sau đó chứa một địa chỉ đĩa và bộ đếm. Mặc dù mỗi mục từ yêu cầu nhiều không gian hơn một địa chỉ đĩa đơn,
nhưng toàn bộ danh sách sẽ ngắn hơn với điều kiện là bộ đếm lớn hơn 1.

5.3. Các phương pháp cấp phát không gian tự do

Tính tự nhiên của truy xuất trực tiếp đĩa cho phép chúng ta khả năng linh hoạt trong việc cài đặt tập tin. Trong hầu hết mọi trường

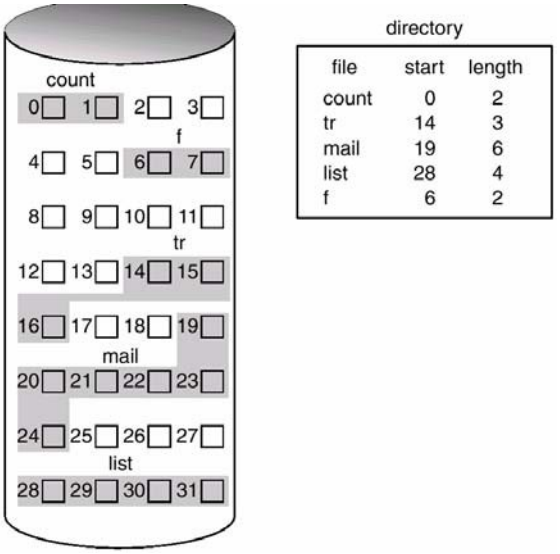
hợp, nhiều tập tin sẽ được lưu trên cùng đĩa. Vấn đề chính là không gian cấp phát tới các tập tin này như thế nào để mà không gian đĩa được sử dụng hiệu quả và các tập tin có thể được truy xuất nhanh chóng. Ba phương pháp quan trọng cho việc cấp phát không gian đĩa được sử dụng rộng rãi: cấp phát kề, liên kết và chỉ mục. Mỗi phương pháp có ưu và nhược điểm.

Một số hệ thống hỗ trợ cả ba. Thông dụng hơn, một hệ thống sẽ dùng một phương pháp cụ thể cho tất cả tập tin.

a) Cấp phát kề

Phương pháp cấp phát kề yêu cầu mỗi tập tin chiếm một tập hợp các khối kề nhau trên đĩa. Các địa chỉ đĩa định nghĩa một thứ tự tuyến tính trên đĩa. Với thứ tự này, giả sử rằng chỉ một công việc đang truy xuất đĩa, truy xuất khối b+1 sau khi khối b không yêu cầu di chuyển trước. Khi di chuyển đầu đọc được yêu cầu (từ cung từ cuối cùng của cylinder tới cung từ đầu tiên của cylinder tiếp theo), nó chỉ di chuyển một rãnh (track). Do đó, số lượng tìm kiếm đĩa được yêu cầu cho truy xuất kề tới các tập tin được cấp phát là nhỏ nhất.

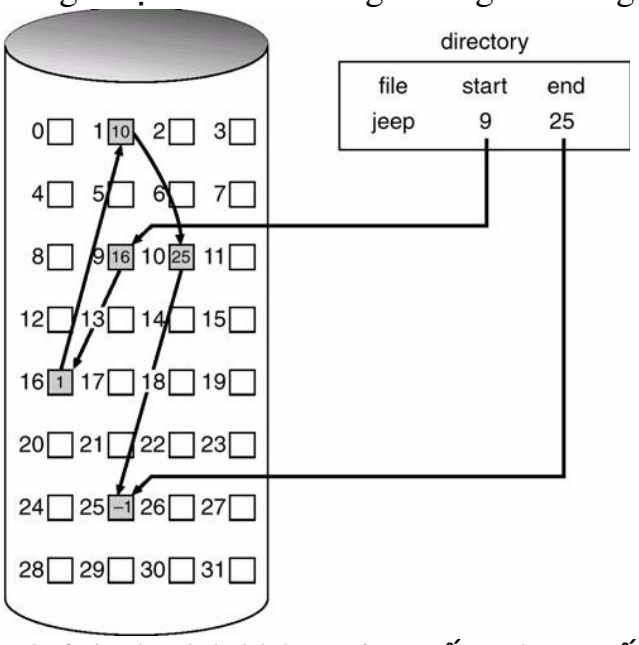
Cấp phát kề của một tập tin được định nghĩa bởi địa chỉ đĩa và chiều dài (tính bằng đơn vị khối) của khối đầu tiên. Nếu tập tin có n khối và bắt đầu tại khối b thì nó chiếm các khối b, b+1, b+2,..,b+n-1. Mục từ thư mục cho mỗi tập tin hiển thị địa chỉ của khối bắt đầu và chiều dài của vùng được cấp phát cho tập tin này



Hình 2.5 danh sách không gian trống được cấp phát kề

b) Cấp phát liên kết

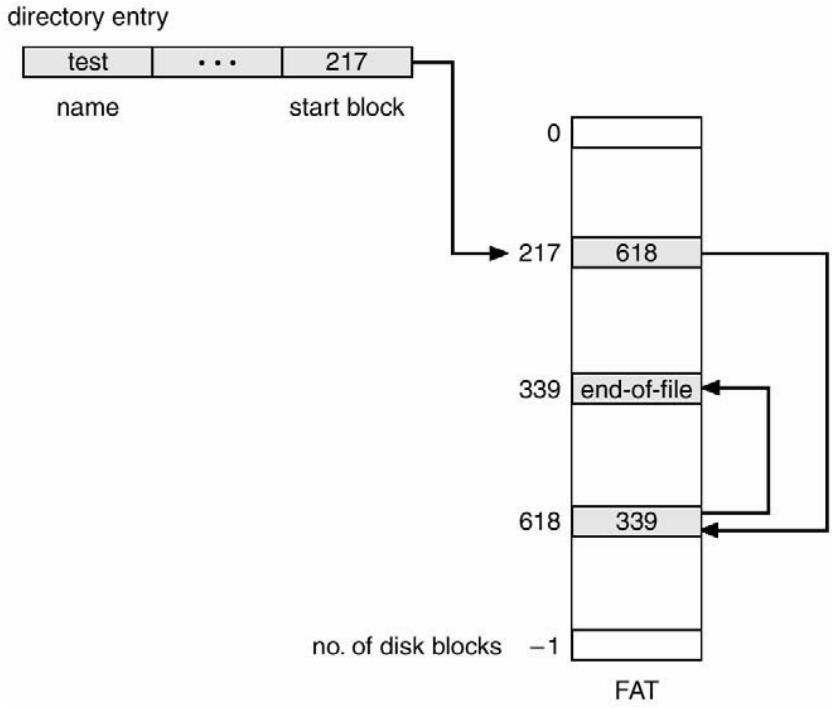
Cấp phát liên kết giải quyết vấn đề của cấp phát kề. Với cấp phát liên kết, mỗi tập tin là một danh sách các khối đĩa được liên kết; các khối đĩa có thể được phân tán khắp nơi trên đĩa. Thư mục chứa một con trỏ chỉ tới khối đầu tiên và các khối cuối cùng của tập tin. Thí dụ, một tập tin có 5 khối có thể bắt đầu tại khối số 9, tiếp tục là khối 16, sau đó khối 1, khối 10 và cuối cùng khối 25 . Mỗi khối chứa một con trỏ chỉ tới khối kế tiếp. Các con trỏ này không được làm sẵn dùng cho người dùng.



Hình 2.6 danh sách không gian trống được cấp phát liên kết

Một thay đổi quan trọng trên phương pháp cấp phát liên kết là dùng bảng cấp phát tập tin (file allocation table-FAT). Điều này đơn giản nhưng là phương pháp
cấp phát không gian đĩa hiệu quả được dùng bởi hệ điều hành MS-

DOS và OS/2. Một phần đĩa tại phần bắt đầu của mỗi phân khu được thiết lập để chứa bảng này. Bảng này có một mục từ cho mỗi khối đĩa và được lập chỉ mục bởi khối đĩa. FAT được dùng nhiều như là một danh sách liên kết. Mục từ thư mục chứa số khối của khối đầu tiên trong tập tin. Mục từ bảng được lập chỉ mục bởi số khối đó sau đó chứa số khối của khối tiếp theo trong tập tin. Chuỗi này tiếp tục cho đến khi khối cuối cùng, có giá trị cuối tập tin đặc biệt như mục từ bảng. Các khối không được dùng được hiển thị bởi giá trị bảng 0. Cấp phát một khối mới tới một tập tin là một vấn đề đơn giản cho việc tìm mục từ bảng có giá trị 0 đầu tiên và thay thế giá trị kết thúc tập tin trước đó với địa chỉ của khối mới. Sau đó, số 0 được thay thế với giá trị kết thúc tập tin. Một thí dụ minh hoạ là cấu trúc FAT của hình X-7 cho một tập tin chứa các khối đĩa 217, 618 và 339.



Hình 2.7 Bảng cấp phát tập tin

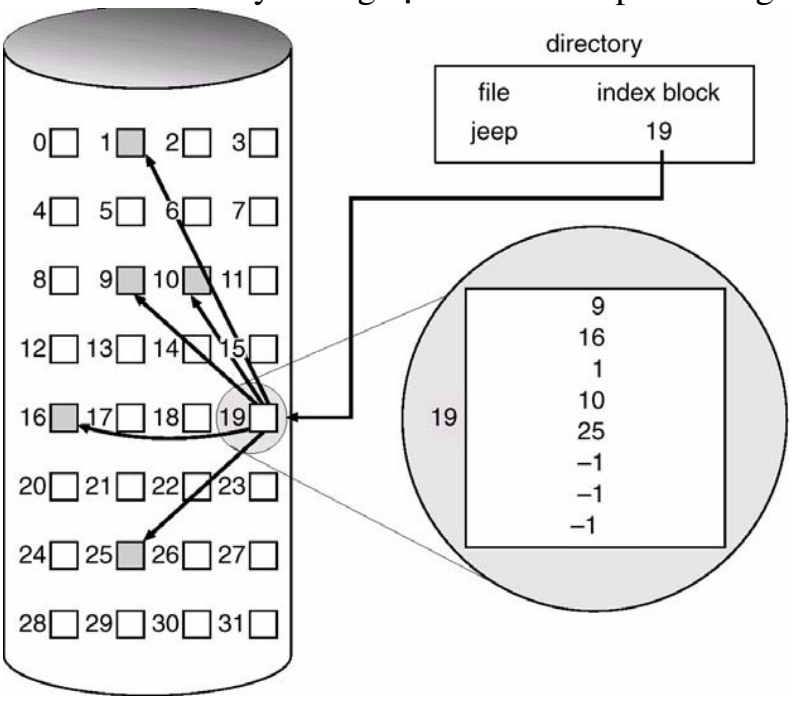
Cơ chế cấp phát FAT có thể dẫn tới số lượng lớn tìm kiếm đầu đọc đĩa nếu FAT không được lưu trữ(cache). Đầu đọc đĩa phải di chuyển tới điểm bắt đầu của phân khu để đọc FAT và tìm vị trí khối sau đó di chuyển tới vị trí của chính khối đĩa đó. Trong trường hợp xấu nhất, cả hai di chuyển xảy ra cho mỗi khối đĩa. Lợi điểm là thời gian truy xuất ngẫu nhiên được cải tiến vì đầu đọc đĩa có thể tìm vị trí của bất cứ khối nào bằng cách đọc thông tin trong FAT.

c) Cấp phát được lập chỉ mục

Cấp phát liên kết giải quyết việc phân mãnh ngoài và vấn đề khai báo kích thước của cấp phát kề. Tuy nhiên, cấp phát liên kết

không hỗ trợ truy xuất trực tiếp hiệu quả vì các con trỏ chỉ tới các khối được phân tán với chính các khối đó qua đĩa và cần được lấy lại trong thứ tự. Cấp phát được lập chỉ mục giải quyết vấn đề này bằng cách mang tất cả con trỏ vào một vị trí: khối chỉ mục (index block).

Mỗi tập tin có khối chỉ mục của chính nó, khối này là một mảng các địa chỉ khối đĩa. Mục từ thứ i trong khối chỉ mục chỉ tới khối i của tập tin. Thư mục chứa địa chỉ của khối chỉ mục (như hình X-8). Để đọc khối i, chúng ta dùng con trỏ trong mục từ khối chỉ mục để tìm và đọc khối mong muốn. Cơ chế này tương tự như cơ chế phân trang.



Hình 2.8 Cấp phát không gian đĩa được lập chỉ mục

5.4. Lập lịch cho đĩa

Khái niệm về lập lịch cho đĩa

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

Một số phương pháp lập lịch

-FCFS
-SSTF
-Scan
-C-Scan
-Look
-C-Look

5.5. Hệ file

Dữ liệu được xử lý trong máy tính được bảo quản lâu dài trên băng từ, đĩa từ, đĩa quang v.v… và dữ liệu được tập hợp lại một cách có tổ chức thành các file dữ liệu theo mục đích sử dụng. File có thể là chương trình của người dùng, một chương trình của hệ điều hành, một văn bản, một tập hợp dữ liệu. Trong một số hệ điều hành, một số thiết bị ngoại vi cũng được quan niệm như file dữ liệu.

Theo góc độ quan sát của người dùng, dữ liệu trong một file lại được tổ chức thành các bản ghi lôgic (gọi tắt bản ghi), mà mỗi bản ghi lôgic có thể là một byte hoặc một cấu trúc dữ liệu nào đó. Bản ghi chính là đơn vị dữ liệu mà chương trình người dùng quan tâm đến và xử lý theo mỗi nhịp làm việc: file là tập hợp (được người dùng quan niệm là một dãy) các bản ghi có tổ chức. Thông thường, trong file tồn tại một thứ tự giữa các bản ghi, thứ tự đó thể hiện vị trí logic giữa các bản ghi với nhau (chẳng hạn như thứ tự đưa bản ghi vào file).

CÂU HỎI VÀ BÀI TẬP

1. Nêu các phương pháp tổ chức và truy nhập dữ liệu.
2. Nêu chức năng của hệ thống điều khiển dữ liệu.
3. Khái niệm về kết khối và tách khối.
4. Nêu vai trò buffer.
5. Trình bày các khối điều khiển dữ liệu
6. Trình bày các phương pháp của quản lý và cấp phát không gian nhớ trên bộ nhớ ngoài của hệ điều hành.
7. Khái niệm file.

HƯỚNG DẪN TRẢ LỜI

1. Các phương pháp tổ chức : *tổ chứ kế tiếp, tổ chức chỉ số kế tiếp, tổ chức truy nhập trực tiếp, tổ thư viện, tổ chức theo bộ nhớ ảo*.

Các phương pháp truy nhập dữ liệu: *cách thức truy nhập tuần tự, cách thức truy nhập cơ sở.*

2. Chức năng của hệ thống điều khiển dữ liệu: Bảo quản dữ liệu trên thiết bị ngoài, đảm bảo cách thức tổ chức khác nhau đối với dữ liệu, thực hiện các phương pháp truy nhập khác nhau tới dữ liệu, catalog

dữ liệu và thực hiện việc tìm kiếm tự động hóa dữ liệu theo tên kí hiệu mà không cần theo địa chỉ.

3. ***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 thiết bị nhớ ngoài.

   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 sau khi hệ điều hành đã đọc một khối từ vật dẫn ngoài vào bộ nhớ trong và trước khi chương trình người dùng xử lý bản ghi.

4. Buffer là vùng nhớ đệm trung gian lưu trữ tạm thời, thuận tiện cho việc vào –ra.
5. Có các khối điều khiển: *Khối FCB (File Control Block), Khối DCB (Data Control Block), Khối UCB (Unit Control Block)*.
6. Các phương pháp quản lý không gian nhớ : *quản lý bằng bit vectơ (bitmap), quản lý bằng danh sách móc nối*.

   Các phương pháp cấp phát không gian nhớ: *cấp phát kề (liên tục), cấp phát liên kết, cấp phát chỉ số.*

7. Xem phần khái niệm file.

CHƯƠNG 3: ĐIỀU KHIỂN BỘ NHỚ

Chương 3: **Điều khiển bộ nhớ** **Mã chương**:..M3

- Nắm được

Mục tiêu:

Sau khi học xong bài học này, sinh viên có khả năng: nguyên lý điều khiển bộ nhớ của HĐH, phương thức tối ưu hóa việc phân phối bộ nhớ, tránh lãng phí và chia sẻ tài nguyên bộ nhớ**.**

1. Quản lý và bảo vệ bộ nhớ

Mục tiêu : Nắm được các khái niệm về bộ nhớ, quản lý phân phối bộ nhớ và vấn đề bảo vệ bộ nhớ.

1.1. Một số khái niệm liên quan đến bộ nhớ

Đơn vị lưu trữ và địa chỉ hóa bộ nhớ trong được chọn là byte hoặc từ máy song phổ biến nhất là byte. Địa chỉ được bắt đầu từ 0.

Trong các lệnh, địa chỉ (của chương trình, tạo ra không gian địa chỉ) được cho theo một dạng sau đây :

Địa chỉ tuyệt đối: địa chỉ thực sự trong bộ nhớ. Ví dụ về việc truy nhập địa chỉ tuyệt đối xảy ra trong chương trình là khi cần chuyển điều khiển từ đơn vị chương trình này sang đơn vị chương trình khác. Địa chỉ tuyệt đối thường được cho theo độ dài từ máy, chẳng hạn, với từ máy 32 bit không gian địa chỉ lên đến 4 GB. Trường hợp ngoại lệ như trong máy vi tính 16 bit, nếu dùng một từ máy địa chỉ hóa chỉ tới 64KB, thì địa chỉ tuyệt đối được cho bằng hai từ máy : một từ máy được dùng để chỉ segment, một từ dùng để chỉ offset.

Các toán hạng trong một lệnh có thể là địa chỉ của một vùng nhớ nào đó (một, hai và thậm chí ba địa chỉ vùng nhớ) nếu chỉ dùng địa chỉ tuyệt đối thì độ dài của lệnh máy sẽ dài và kéo theo sự tăng đáng kể độ dài của toàn bộ chương trình. Đó là một trong những lý do chính dẫn tới cần dùng giải pháp sử dụng địa chỉ tương đối.

Địa chỉ tương đối : Có nhiều cách thức để biểu thị địa chỉ tương đối. Một trong những cách điển hình là đối với địa chỉ liên tiếp nhau sẽ sử dụng chung một thanh ghi (được gọi là thanh ghi cơ sở) chứa địa chỉ đầu tiên trong dãy đó, các địa chỉ còn lại được quy chiếu bằng một gia số so với địa chỉ đầu (nội dung của thanh ghi cơ sở). Gia số chính là khoảng cách của địa chỉ đang tính với địa chỉ đầu. Khi quy định một thanh ghi xác định nào đó là thanh ghi cơ sở thì trong lệnh không cần thiết nêu thanh ghi cơ sở nữa mà chỉ cần chỉ ra gia số địa chỉ, mà gia số thường nhỏ nên số bit dành cho nó trong lệnh là rất ít (việc dùng các thanh ghi CS, DS, SS, ES trong máy vi tính là ví dụ đáp ứng mục đích này).

Một phương pháp dùng địa chỉ tương đối thường hay gặp là ngoài thanh ghi cơ sở, thì địa chỉ các thành phần trong một cấu trúc dữ liệu còn được tương ứng với một thanh ghi chỉ số : địa chỉ các thành phần trong cấu trúc dữ liệu đó được biểu diễn bằng gia số đối với địa chỉ ở thanh ghi chỉ số. Như vậy, trong lệnh có thể có thêm số hiệu của thanh ghi chỉ số song mỗi máy tính lại chỉ có rất ít thanh ghi nên số bit dành cho địa chỉ cũng giảm đi. Chẳng hạn, câu lệnh WITH trong PASCAL đã sử dụng cơ chế nói trên hoặc mode địa chỉ [BX + SI +4] trong ngôn ngữ assembler trên PC.

Hệ thống cần phân phối hay giải phóng bộ nhớ đối với chương trình người sử dụng : đơn vị cung cấp hay giải phóng bộ nhớ thường là trang (page) hoặc một đơn vị bộ nhớ nào đó được hệ thống quy định (ví dụ, trong MS-DOS đơn vị đó là 1 đoạn –paragraph). Trang đó có độ dài 2 KB, 4 KB v.v… song phổ biến nhất là 4KB. Địa chỉ của trang phù hợp với độ dài của trang theo nghĩa địa chỉ chia hết cho độ dài.

Phân phối bộ nhớ cho chương trình còn được phân biệt là phân phối tĩnh hay phân phối động : liên quan đến thời điểm phân phối bộ nhớ là trước (khi tải) hay trong thời gian thực hiện chương trình. Việc phân phối tĩnh hay động được thực hiện đối với cả chương trình lẫn dữ liệu.

Truy nhập tới bộ nhớ cũng phân biệt là truy nhập tuần tự hay truy nhập trực tiếp. Truy nhập tuần tự theo các địa chỉ kế tiếp nhau tương ứng với địa chỉ tương đối (ví dụ, truy nhập tới các phần tử của mảng là tuần tự bắt đầu từ phần tử đầu tiên), còn truy nhập trực tiếp tương ứng với địa chỉ tuyệt đối.

1.2. Quản lý phân phối bộ nhớ. Vấn đề bảo vệ bộ nhớ

Bài toán cơ bản của điều phối bộ nhớ là :

-Phân phối các vùng nhớ cho chương trình và dữ liệu để có thể thực hiện được một cách chính quy, không ảnh hưởng đến các chương trình khác đang tồn tại trong bộ nhớ ;

-Bảo vệ chương trình và dữ liệu không bị xóa hoặc chồng chéo bởi những chương trình khác ;

-Sử dụng bộ nhớ hiệu quả nhất có thể được.

Như vậy, khi điều phối bộ nhớ, đòi hỏi thỏa mãn hai yêu cầu : phân rã được không gian địa chỉ và chia xẻ bộ nhớ. Để đảm bảo được các chức năng cơ bản trên, chương trình điều khiển bộ nhớ phải giải quyết một số nội dung sau đây :

Phân rã không gian địa chỉ để tránh chồng chéo, xâm phạm lẫn nhau giữa các chương trình.

Cũng giống như trong chương trình PASCAL, các biến local (cục bộ) và global (toàn bộ) dù giống nhau về tên nhưng lại được phân phối các vùng địa chỉ hoàn toàn khác nhau, khi xem xét tình trạng bộ nhớ đang có một số chương trình người dùng, cần phân rã các địa chỉ để không chồng chéo. Để làm được điều đó có thể đưa ra một số chiến lược. Một chiến lược điển hình dùng trong một số hệ điều hành là phân lớp cho các vùng bộ nhớ và gắn lớp bộ nhớ cho mỗi chương trình. Một miền bộ nhớ chỉ có thể phân phối cho một số lớp chương trình, cũng như vậy, một chương trình có thể được phân phối vào một số lớp bộ nhớ nào đó. Đối với mỗi trang, có hai trạng thái áp dụng : đã được phân phối hay còn rỗi. Ví dụ, với các hệ đơn chương trình, như MS-DOS trên máy tính PC, quản lý bộ nhớ

đơn giản : sử dụng con trỏ để xác định cận của các vùng bộ nhớ còn rỗi. Tuy nhiên, cũng với máy tính PC với các bộ vi xử lý từ 386 trở đi, cho phép chạy được trong chế độ đa chương trình. Người ta đã phân chia các mức bộ nhớ và các mức chương trình và đã tính đến quyền thâm nhập địa chỉ và cung cấp bộ nhớ : Chỉ khi mức của chương trình cho phép thâm nhập đến một vùng bộ nhớ theo quyền hạn thì mới cập nhật được.

Chia xẻ bộ nhớ liên quan đến việc dùng chung các phần bộ nhớ mà không ảnh hưởng đến nhau. Trong chế độ đa chương trình, một số chương trình người dùng, có thể cùng hướng đến một chương trình hay dữ liệu chung nào đó. Để dùng chung được, các môdun chương trình có thể có chế độ chạy nhiều lần và các chương trình người dùng có thể gọi chương trình nói trên theo yêu cầu của mình độc lập với các chương trình của người sử dụng khác thâm nhập vào chương trình nói trên. Chú ý rằng, để một môdun chương trình được nhiều người dùng chung thì một điều kiện dễ nhận biết là trong khi môdun đó chạy, không gây ra sự biến đổi nội dung các lệnh thuộc môdun đó.

Trong phân phối bộ nhớ cho chương trình, cần chú ý tới hai cách thức là : phân phối liên tục và phân phối rời rạc.

Tùy thuộc vào hệ điều hành và chế độ hoạt động của nó mà phân phối bộ nhớ có thể tiến hành theo một trong hai yêu cầu : phân phối liên tục và phân phối rời rạc.

Phân phối liên tục là một chương trình sẽ chiếm một vùng nhớ liên tục ; nội dung từ đầu chương trình đến cuối chương trình nằm trọn trong vùng nhớ đó và không cho phép chương trình khác sử dụng vùng nhớ chèn giữa vùng nhớ dành cho chương trình. Phân phối liên tục làm đơn giản việc cung cấp bộ nhớ cho chương trình cũng như việc quản lý bộ nhớ.

Phân phối rời rạc là một chương trình có thể được phân chia thành một số đoạn, các đoạn này nằm ở các vùng nhớ rời rạc nhau, giữa các vùng nhớ này có thể có các vùng nhớ được phân phối cho các chương trình khác. Phân phối rời rạc sẽ làm cho bài toán quản lý và phân phối bộ nhớ phức tạp hơn.

Phân phối một vùng nhớ cho chương trình được thực hiện theo một trong hai cách thức : chọn cái đầu tiên và chọn cái tốt nhất. Sự khác nhau của hai cách thức đó được giải thích như sau. Thông thường, tại một thời điểm bất kỳ bộ nhớ trong có một số vùng bộ nhớ rỗi rời rạc nhau do việc giải phóng các chương trình nào đó trong bộ nhớ. Các vùng nhớ này có kích thước khác nhau được quản lý trong một danh sách có thứ tự nào đó. Giả sử nảy sinh nhu cầu cần phân phối một dung lượng n đơn vị bộ nhớ (trang, paragraph…) cho việc thực hiện một chương trình nào đó. Theo cách thức « chọn cái đầu tiên » thì vùng nhớ rỗi đầu tiên trong danh sách có dung lượng lớn hơn hoặc bằng n đơn vị nhớ sẽ được sử dụng để phân phối. Theo cách thức « chọn cái tốt nhất » thì vùng nhớ rỗi có dung lượng

lớn hơn hay bằng n đơn vị nhớ mà độ dư thừa ít nhất sẽ được sử dụng để phân phối cho nhu cầu nói trên.

2. Điều khiển bộ nhớ liên tục theo đa bài toán

Mục tiêu: Nắm được các phương pháp điều khiển bộ nhớ liên tục.

2.1. Chiến lược giới hạn tĩnh (cận cố định)

Một trong những phương pháp điển hình phân phối bộ nhớ liên tục là chiến lược giới hạn tĩnh còn gọi là chiến lược phân chương (tương ứng với chế độ MFT của hệ điều hành). Bộ nhớ được chia thành các chương: gán tên chương, địa chỉ, dung lượng trong quá trình khởi tạo hệ điều hành. Hình 3.4 cho một hình ảnh phân chương bộ nhớ và việc phân phối bộ nhớ cho một số chương trình.

Hình 3.1 Bộ nhớ được phân chương

Đối với ví dụ theo hình vẽ 3.4, bộ nhớ được phân ra thành 5 chương: P0 (32K), P1 (40K), P2 (40K), P3 (72K), P4 (72K). Chương P0 được dành cho nhân, mỗi chương còn lại đã có một chương trình được tải (load). Kích cỡ (dung lượng) trung bình của mỗi chương phụ thuộc vào dung lượng của bộ nhớ và số lượng chương. Các chương trình được gán số hiệu để chỉ có thể tải vào những chương trình nhất định. Nảy sinh trường hợp có thể có những chương rỗi mà không tải được chương trình: lớp gắn với nó bị bận hoặc độ rộng của chương không đủ để tải. Lúc đó hoặc hệ thống hoặc thao tác viên thực hiện việc thay đổi lớp gắn cho chương trình hoặc thay đổi số lượng chương, kích cỡ chương song phổ biến là thao tác viên dùng lệnh để thực hiện công việc đó. Tuy điều đó xem ra có vẻ thủ công song tránh được sự phức tạp cho chương trình điều khiển.

184K	P4(72K)
112K	P3(72K)
72K	P2(40K)
32K	P1(40K)
0K	P0(32K)
Địa chỉ	*Chương bộ nhớ*

Để quản lý bộ nhớ trong trường hợp này, sử dụng bảng mô tả chương (partition description table: PDT), có dạng:

Số hiệu chương	Địa chỉ	Độ dài	Tình trạng
0	0K	32K	Đã load
1	32K	40K	Đã load
2	72K	40K	Đã load
3	112K	72K	Đã load
4	184K	72K	Đã load

Đối với một bài toán, nó được gắn với một vài chương bộ nhớ, chiến lược phân phối bộ nhớ cho nó có thể được kể làm hai hướng: phân phối nhanh nhất (gặp chương được gắn, đủ độ rộng đầu tiên), phân phối tối ưu (chọn chương với vùng nhớ dư thừa là ít nhất). Trở lại vấn đề vướng mắc khi phân phối bộ nhớ:

-Không có chương nào đủ để phân phối cho chương trình;

-Mọi chương đã được tải;

-Một số chương rỗi, mỗi chương rỗi không đủ chứa bài toán song nối vài chương rỗi tạo ra một vùng nhớ đủ để tải bài toán.

Việc phân phối bộ nhớ cho bài toán (quá trình) được coi như gắn với mỗi chương có 1 dòng xếp hàng các bài toán cần được phân phối bộ nhớ đối với nó. Mỗi bài toán lại có thể gắn với một vài chương, có sự chung nhau giữa một số dòng xếp hàng. Việc phân phối bộ nhớ cho một bài toán liên quan tới việc thao tác đối với các dòng xếp hàng nói trên.

Mối liên kết giữa chương và lớp bài toán không phải là luôn chặt chẽ. Như trên đã thấy, tồn tại một số cách thức thay đổi mối liên kết nói trên (hoặc do chương trình hệ thống hoặc do thao tác viên v.v…).

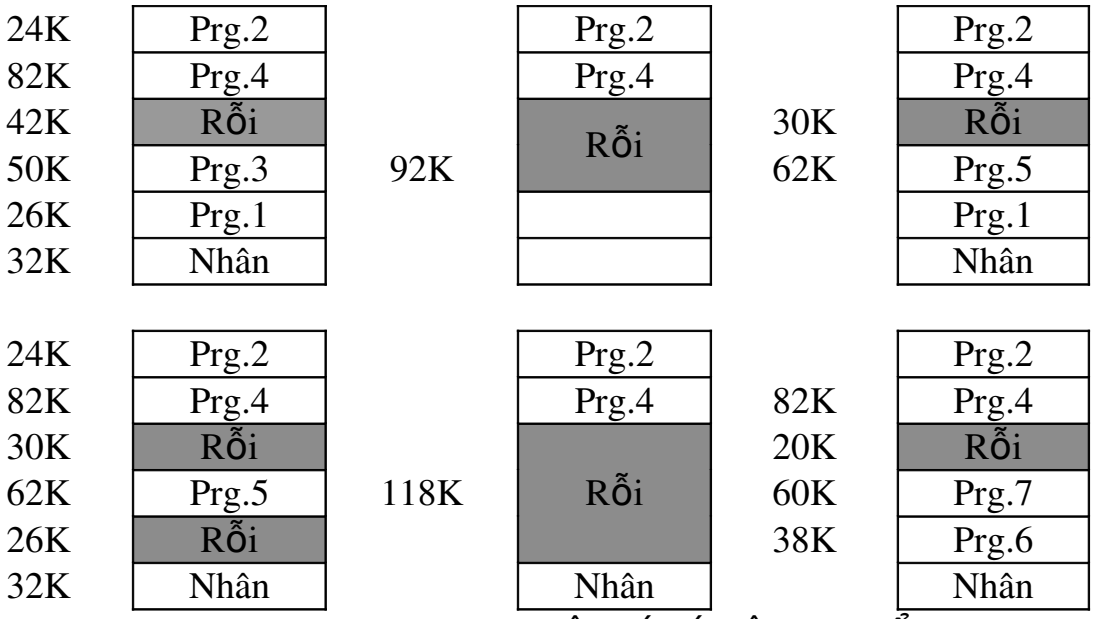
2.2 Chiến lược giới hạn động (cận thay đổi)

Như trên đã thấy, chế độ phân phối cận cố định (phân phối tĩnh) nảy sinh một số vấn đề trong việc sử dụng tối ưu bộ nhớ, với phương án khắc phục đưa vào lệnh của thao tác viên. Trong cách thức phân phối liên tục bộ nhớ, chế độ giới hạn thay đổi được áp dụng.

Trong chế độ này (tương ứng với chế độ MVT của hệ điều hành), bộ nhớ không chia thành các chương giống như ở chế độ giới hạn cố định. Các chương trình nạp liên tục vào bộ nhớ cho đến khi còn nạp được. Một ví dụ về hình ảnh của bộ nhớ trong được cho trong hình 3.5.

Trong quá trình làm việc, các chương trình được thực hiện và giải phóng, các vùng bộ nhớ giải phóng đó có thể liên tục hoặc rời rạc. Sử dụng vùng bộ nhớ đó ra làm sao. Một số tình huống nảy sinh (hình 3.5).

Trên hình vẽ thứ 6, chương trình 4 (Prg 4) được giải phóng đầu tiên. Ngay trước chương trình 4, một vùng nhớ rỗi với dung lượng 20K. Khi giải phóng chương trình 4, có một vùng rỗi liên tục với dung lượng 102K. Chương trình 8 với độ dài 52K được tải vào trong bộ nhớ trong và sau đó chương trình 6 được giải phóng. Hiện tại, trên dòng đợi, đến lượt chương trình Pr9 có độ dài 80K. Mỗi vùng rỗi riêng rẽ trong bộ nhớ không thể chứa nối chương trình 9, trong khi đó dung tích rỗi tổng cộng là 88K. Hệ thống cần nhập hai vùng nhớ rỗi trên để nạp được chương trình Pr9.



Hình 3.2. Các hình trạng bộ nhớ với cận thay đổi

Điều khiển bộ nhớ theo cận thay đổi sử dụng linh hoạt tối ưu bộ nhớ, tránh được một số hạn chế so với cận cố định (cho phép độ dài của môdun chương trình lớn) và miền nhớ rỗi được sử dụng linh hoạt. Tuy vậy, công việc phân phối bộ nhớ là phức tạp.

-quản lý bộ nhớ luôn thay đổi

-định vị lại bộ nhớ cho các chương trình.

Khi chương trình đang hoạt động, nó đang ở trạng thái trung gian, nếu không có những cơ chế thích hợp thì việc định vị lại sẽ ảnh hưởng đến sự thực hiện chương trình. Điều này cũng liên quan đến vấn đề địa chỉ hóa trong chương trình: sử dụng địa chỉ cở sở không tường minh. Chỉ khi có thể quy chiếu trên địa chỉ không tường minh mới có thể giải quyết được bài toán định vị lại như trên.

Mặc khác, không phải thời điểm nào cũng cho phép định vị lại. Chương trình đang đợi kết quả của công việc vào/ra thì việc định vị lại

gặp trở ngại lớn trong vấn đề liên kết kết quả công việc vào/ra với chương trình.

Vấn đề định vị lại có ý nghĩa không chỉ trong phân phối bộ nhớ liên tục mà cả trong phân phối bộ nhớ gián đoạn. Việc sử dụng địa chỉ tương đối là một hình thức phù hợp với việc định vị lại. Có một số cách thức liên quan đến định vị lại: định vị tĩnh và định vị động.

2.3. Cách thức Overlay và swapping

a) Cách thức OVERLAY

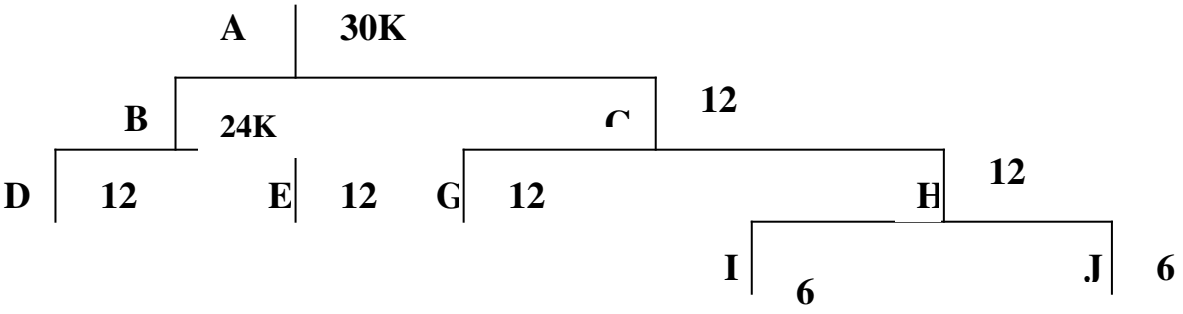
Trong các trường hợp điều khiển bộ nhớ nói trên, để khắc phục hiện tượng thiếu bộ nhớ khi phân phối liên tục, một số hệ thống cho phép chương trình hoạt động theo chế độ OVERLAY.

Chế độ OVERLAY cho phép tổ chức chương trình thành các đơn vị chương trình và đảm bảo các điều kiện sau:

Phân phối bộ nhớ cho chương trình trong một miền liên tục,

Môdun tải bao gồm một số đơn vị chương trình (segment chương trình) mà các segment chương trình được tải đồng thời (phân phối liên tục). Một số môdun tải có thể được tải vào cùng 1 vùng bộ nhớ. Các môdun tải như vậy được tập hợp trong các File trên đĩa.

Trong tập hợp các môdun chương trình sẽ nảy sinh quan hệ độc lập/phụ thuộc: sự có mặt của một nhóm môdun trong bộ nhớ *không đòi hỏi/có đòi hỏi* sự có mặt của một nhóm môdun khác.



Hình 3.3 Cấu trúc chương trình OVERLAY

Trong các môdun nói trên, có một môdun luôn tồn tại trong quá trình chương trình thực hiện: đó là chương trình chính, mọi môdun chương trình đều phụ thuộc vào nó sẽ được tổ chức dưới dạng hình cây. Hình trên đây (hình 3.8) cho một ví dụ cấu trúc một chương trình: bộ nhớ đòi hỏi của môdun A:30KB; B:24KB, C:12KB, D, E, G, H: 12KB; I, J: 6KB. Theo cấu trúc đó: A (môdun chính) sử dụng hai môdun B và C. B và C độc lập nhau: chúng có thể được lưu trữ trên cùng một vùng nhớ. B sử

dụng hai môdun độc lập là D và E; C sử dụng hai môdun độc lập là G và H. H sử dụng hai môdun độc lập là I và J.

Như vậy, cây chương trình có gốc B cần 36KB; cây chương trình có gốc C cần 30KB. Vậy chương trình cần vùng nhớ liên tục là 30KB +36KB = 66KB. Trong khi đó nếu không sử dụng chế độ overlay, chương trình nói trên cần vùng nhớ liên tục là 30KB +24KB+5*12KB+2*6KB=126KB.

Ví dụ, trong các ngôn ngữ lập trình nói chung cho phép chế độ này. Chẳng hạn, khai báo OVERLAY trong PASCAL, FORTRAN v.v…Cụ thể, trong TP3.0, các môdun overlay được cho vào File cùng phần tên với file chương trình nhưng phần mở rộng là 001, 002 v.v…;Một số phần mềm, có file đi kèm có phần mở rộng OVL, chứa các môdun cùng mức khi tải vào bộ nhớ trong.

b) Cách thức swapping

Swapping là cách thức hệ thống thực hiện việc chuyển giao nội dung một số phần bộ nhớ ra đĩa từ để giải phóng bộ nhớ cho một yêu cầu phân phối hiện tại. Phần nội dung bộ nhớ được chuyển ra ngoài chứa cả nội dung các chương trình đang tồn tại trong bộ nhớ; sau đó khi chương trình nói trên được chọn thực hiện, thì phần bộ nhớ được đưa từ đĩa từ vào bộ nhớ trong.

Swapping áp dụng chủ yếu cho điều phối bộ nhớ liên tục song trong một số trường hợp cũng được các hệ điều hành hoạt động điều phối bộ nhớ gián đoạn sử dụng.

Về hình thức, có thể coi swapping là một biến thể của overlay: Trong overlay, môdun chương trình chính thuộc chương trình người dùng còn trong swapping, môdun chương trình chính thuộc về hệ điều hành, còn môdun overlay là các chương trình người dùng, trong một số trường hợp thì thậm chí đấy là các bộ phận các chương trình người dùng.

Trong các hệ điều hành dùng cách thức swapping, tồn tại một môdun hệ thống tên là swapper có chức năng như sau:

Chọn quá trình (chương trình người dùng) để đưa ra đĩa từ (swap out)

Chọn quá trình để đưa trở lại từ đĩa vào bộ nhớ trong (swap in)

Định vị và quản lý không gian swap (trong bộ nhớ trong cũng như trên đĩa từ).

Swap out

Swapper định hướng chọn quá trình được đưa ra đĩa từ (quá trình bị swap out) là quá trình đang bị đình chỉ mà đang chiếm một vùng nhớ đủ lớn để có thể phân phối bộ nhớ cho quá trình đang được nạp vào bộ nhớ trong.

Trong những quá trình thóa mãn điều kiện trên, swapper sẽ chọn lựa quá trình có độ ưu tiên thấp nhất, chờ đợi một sự kiện xảy ra chậm và quá trình này thường xuyên bị đình chỉ khi thống kê trong một khoảng thời gian dài. Một số điều cần chú ý khi chọn quá trình bị swap out là tính đến là thời gian quá trình đó đã tải (hoặc nhận) vào bộ nhớ trong, tính chất thực hiện trong bộ nhớ trong của quá trình đó v.v… Cần tránh trường hợp một quá trình vừa bị swap out xong thì lại cần gửi nó vào lại bộ nhớ trong (swap in).

Swap in

Swapper chọn quá trình đang ở bộ nhớ ngoài (do swap out) nhận lại vào bộ nhớ trong phụ thuộc vào một số thông số: thời gian quá trình đã ở bộ nhớ ngoài, độ ưu tiên của quá trình v.v… Mục tiêu của công việc chọn lựa này là đảm bảo sao cho thời gian dành cho swap out và swap in là ít nhất có thể được.

Định vị và quản lý không gian swap

Các quá trình trong trạng thái swap out được hệ điều hành lưu trữ dưới dạng thực hiện được cùng với dữ liệu có liên quan lên một File trên đĩa được gọi là File swap. Không những thế, File này còn chứa các thuộc tính của quá trình bị swap, chẳng hạn như độ ưu tiên của quá trình và yêu cầu bộ nhớ đối với quá trình đó. Trong một số trường hợp File như trên còn được gọi là *ảnh của quá trình.* Do quá trình khi thực hiện làm thay đổi stack và dữ liệu cho nên ảnh- dạng đang thực hiện của quá trình khác với ảnh lưu trữ thông thường khi chưa được tải. Nói chung, cũng giống như việc bảo vệ trạng thái quá trình khi chuyển điều khiển, khi swap out, vùng bảo vệ tương ứng với quá trình đó cũng được lưu trên ảnh của nó.

Tồn tại hai lựa chọn cơ sở đối với việc định vị File swap:

- 1 file swap cho toàn bộ hệ thống
- 1 số file swap chuyên dụng theo quá trình.

Lựa chọn chỉ 1 file swap cho toàn bộ hệ thống: Một file rất lớn được khởi tạo, thường xảy ra tại thời điểm khởi tạo hệ thống, chứa mọi ảnh swap out của mọi quá trình. Nói chung, File swap chung đó đặt trên "bộ nhớ ngoài" tốc độ cao; File đó thường có địa chỉ và độ rộng tĩnh. Một điều quan trọng đối với sự lựa chọn này là kích cỡ của file swap chung đó. Nếu kích cỡ của nó quá lớn thì vùng trên "bộ nhớ ngoài" dành cho các mục đích khác sẽ bé, ảnh hưởng đến tốc độ hoạt động chung của hệ thống. Ngược lại, nếu kích cỡ của file nhỏ thì có thể xảy ra tình huống sai sót khi swap out.

Lựa chọn một số file swap chuyên dụng: mỗi một quá trình bị swap out sẽ tương ứng với một file trên bộ nhớ ngoài (nhiều file ảnh). File swap (ảnh của quá trình) được khởi tạo hoặc dạng tĩnh (ngay khi quá trình được nạp vào bộ nhớ trong) hoặc động (khi cần swap out mới tạo file).

Việc swap out và swap in đối với quá trình ảnh hưởng đến thời gian thực hiện quá trình và liên quan đến tốc độ vào ra với bộ nhớ ngoài.

2.4. Các phương thức phân phối vùng nhớ (first fit, best fit, worst fit)

Tập hợp các lỗ trống được tìm thấy để xác định lỗ nào là tốt nhất để cấp phát. Các chiến lược first-fit, best-fit, worst-fit là những chiến lược phổ biến nhất được dùng để chọn một lỗ trống từ tập hợp các lỗ trống.

- First-fit: cấp phát lỗ trống đầu tiên đủ lớn. Tìm kiếm có thể bắt đầu tại đầu tập hợp các lỗ trống hay tại điểm kết thúc của tìm kiếm first-fit trước đó. Chúng ta dừng tìm kiếm ngay khi chúng ta tìm thấy một lỗ trống đủ lớn.
- Best-fit: cấp phát lỗ trống nhỏ nhất đủ lớn. Chúng ta phải tìm toàn bộ danh sách, trừ khi danh sách được xếp thứ tự theo kích thước. Chiến lược này tạo ra lỗ trống nhỏ nhất còn thừa lại.
- Worst-fit: cấp phát lỗ trống lớn nhất. Chúng ta phải tìm toàn danh sách trừ khi nó được xếp theo thứ tự kích thước. Chiến lược này tạo ra lỗ trống còn lại lớn nhất mà có thể có ích hơn lỗ trống nhỏ từ tiếp cận best-fit.

Các mô phỏng hiển thị rằng cả first-fit và best-fit là tốt hơn worst-fit về việc giảm thời gian và sử dụng lưu trữ. Giữa first-fit và best-fit không thể xác định rõ chiến lược nào tốt hơn về sử dụng lưu trữ, nhưng first-fit có tốc độ nhanh hơn.

Tuy nhiên, các giải thuật này gặp phải vấn đề phân mãnh ngoài (external fragmentation). Khi các quá trình được nạp và được xoá khỏi bộ nhớ, không gian bộ nhớ trống bị phân rã thành những mãnh nhỏ. Phân mãnh ngoài tồn tại khi tổng không gian bộ nhớ đủ để thoả mãn một yêu cầu, nhưng nó không liên tục; vùng lưu trữ bị phân mãnh thành một số lượng lớn các lỗ nhỏ. Vấn đề phân mãnh này có thể rất lớn. Trong trường hợp xấu nhất, chúng có thể có một khối bộ nhớ trống nằm giữa mỗi hai quá trình. Nếu tất cả bộ nhớ này nằm trong một khối trống lớn, chúng ta có thể chạy nhiều quá trình hơn.

3. Điều khiển bộ nhớ gián đoạn

Mục tiêu: nắm được phương thức tối ưu hóa việc phân phối bộ 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 trong bộ nhớ trong để làm việc và nói

chung thì số chương trình này không giảm đi như đã xét theo chế độ mẻ. 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 như chế độ overlay 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, cho phép các tên logic là kí 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 tên được phép dùng là "vô hạn".

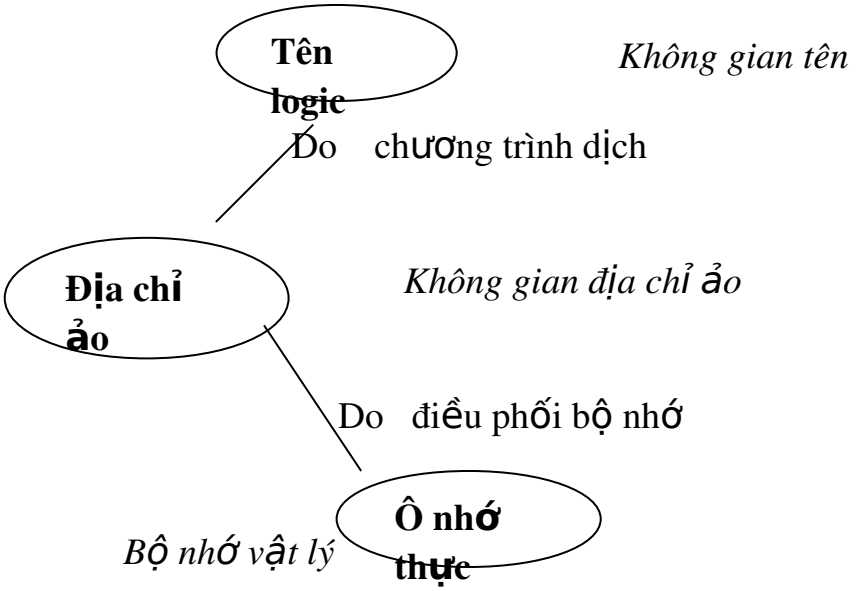
Hệ thống chương trình cần phải định vị được "bộ nhớ" đối với mỗi tên trong chương trình: cần ánh xạ không gian tên vào địa chỉ vật lý và trong ánh xạ đó nảy sinh khái niệm không gian địa chỉ ảo. Ánh xạ từ không gian tên tới bộ nhớ vật lý được chia làm hai bước (hình3.9).

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 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 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 tên nảy sinh 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.



Hình 3.4. Ánh xạ bộ nhớ ảo

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

Bộ dịch 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 cập trực tiếp tới địa chỉ thực.

Biện pháp giải quyết mềm dẻo 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 hiện 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 segment và theo trang như được trình bày dưới đây.

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. Phân đoạn

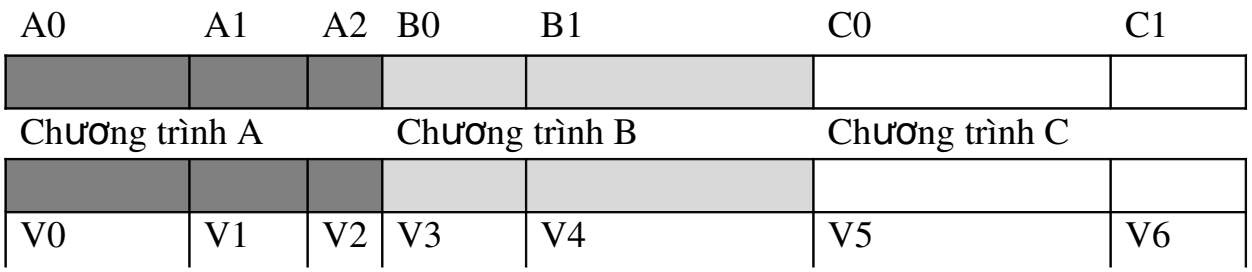
a. Khái niệm segment

Người sử dụng không nhất thiết quan niệm không gian tên là liên tục, mà họ có thể quan niệm chương trình là một tập hợp các phần lôgic (được gọi là segment) mỗi từ chúng hoặc là miền dữ liệu, thủ tục hay một bộ thủ tục (như vậy, khái niệm segment liên quan đến bộ phận của chương trình mà không là bộ nhớ). Người dùng hướng tới ô nhớ thông qua tên (thực tế sau khi dịch là số hiệu của segment và gia số tương đối so với đầu segment). Cho phép khả năng độ dài segment biến động trong thời gian sử dụng. Việc định ra các segment do người lập trình phải làm. Địa chỉ nội tại trong segment là liên tục, một số segment của một chương trình người dùng không phải tạo thành một vùng liên tục trong bộ nhớ trong; hơn nữa, không phải mọi segment của một chương trình đều nằm trong bộ nhớ trong. Nguyên lý cơ bản của điều khiển bộ nhớ rời rạc theo segment là ở chỗ: ánh xạ bộ nhớ thực hiện việc chuyển dịch từ ô bộ nhớ ảo vào ô nhớ vật lý mỗi khi hướng tới bộ nhớ (định vị bộ nhớ cho segment).

Nếu tất cả segment một chương trình đều đang nằm ở bộ nhớ trong thì việc phân phối segment giống như các đoạn động, ánh xạ thực hiện được nhờ chỉ các thanh ghi chỉ dẫn, liên kết với mỗi đoạn có hướng đến địa chỉ. Nếu có chỉ 1 thanh ghi thì giống như MVT. Thực sự tồn tại kiểu máy tính với nhiều thanh ghi cho định vị segment (Univac có hai: một cho lệnh, một cho dữ liệu).

Tổng quát hơn là các segment nằm cả ở bộ nhớ ngoài, nằm cả ở bộ nhớ trong. Chính hệ điều hành đảm bảo thực hiện việc phủ không tường minh độc lập với việc người lập trình có chỉ ra cấu trúc của việc phủ hay không.

Hình 3.10 cho ví dụ về segment không gian bộ nhớ ảo của các chương trình và việc tải chúng trong bộ nhớ trong. Trong hình vẽ 3.10, cho biết chương trình A (người dùng A) bao gồm 3 segment A0, A1, A2; chương trình B (người dùng B) bao gồm 2 segment B0 và B1; chương trình C (người dùng C) bao gồm 2 segment C0 và C1,…



Hình 3.5. các segment không gian bộ nhớ ảo của các chương trình

Trong bộ nhớ ảo các chương trình này được phân phối bộ nhớ ảo liên tục, mỗi chương trình nằm trên một vùng liên tục. Hệ thống cũng quan niệm rằng, trong không gian bộ nhớ ảo, tập hợp các segment của mọi

chương trình người dùng xếp trên đó được đánh thứ tự theo trình tự xuất hiện (trên hình vẽ, chúng có tên là V0, V1, V2,..). Quản lý toàn bộ bộ nhớ ảo thông qua việc quản lý các segment ảo nói trên.

Hình 3.11 cho biết hình ảnh của bộ nhớ trong quá trình máy tính hoạt động: các segment của các chương trình người dùng được nạp vào bộ nhớ trong theo yêu cầu. Chú ý là các segment của một chương trình có thể đặt ở mọi vị trí rỗi cho phép và các segment của cùng một chương trình nằm ở các vùng nhớ rời rạc nhau (hai segment B0 và B1 của chương trình B).

Hình 3.6 Các segment trong bộ nhớ thực của các chương trình

b.Điều khiển bộ nhớ theo segment

Bảng segment: Toàn bộ không gian bộ nhớ ảo được thể hiện trong một bảng segment tổng thể. Mỗi phần tử trong bảng này tương ứng với một segment trong một chương trình người dùng nào đó. Bảng segment được dùng để thực hiện được việc định vị cho segment. Từ bảng segment tổng thể có thể biết được hình ảnh của toàn bộ không gian bộ nhớ ảo.

Đối với mỗi chương trình của người dùng tương ứng có một bảng segment người dùng để định vị việc phân phối bộ nhớ thực cho chương trình người dùng mỗi khi nảy sinh sự hướng tới. Như hình vẽ phía trên, mỗi bảng segment người dùng thực chất chỉ là một vùng con liên tục của bảng segment tổng thể.

0	0	…	…
	1	4400	1000
	0	…	…
	1	5400	1000
	1	400	4000
3	0	…	…
	1	6400	1000
5			

0	1	5400	1000
1	1	400	4000

0 … …

Bảng segment tổng thể Bảng segment của chương trình B

Bảng 3.7 Các bảng segment

Tồn tại thanh ghi bảng segment để chỉ đầu của bảng segment cho chương trình hiện tại. Chỉ dẫn bộ nhớ có dạng (s, d) trong đó: s là số hiệu segment, còn d là gia số. Khi bổ sung s tới thanh ghi bảng segment, hệ điều hành xác định được vị trí vật lý, vị trí phần tử bảng segment đối với segment cần hướng tới.

Cấu trúc phần tử trong các bảng segment: bảng segment tổng thể (hay cũng vậy bảng segment của chương trình người dùng) bao gồm một số bản ghi cùng kiểu, mỗi bản ghi có ba trường:

Trường đầu tiên là dấu hiệu segment: nhận giá trị 0 hay 1 tùy thuộc vào hiện tại segment có mặt ở trong bộ nhớ trong hay không: 0 hiện không có mặt trong bộ nhớ trong, 1 là hiện đang có mặt.

Nếu nó đang ở bộ nhớ trong thì nội dung hai trường sau mới có ý nghĩa. Trường thứ hai chứa địa chỉ của vùng định vị segment đó; trường thứ ba chứa độ dài hiện tại của segment.

Cộng nội dung trường thứ hai với gia cố d nêu trên sẽ cho địa chỉ cần hướng tới. Việc ánh xạ hai bước được cho như trình bày ở hình 3.13.

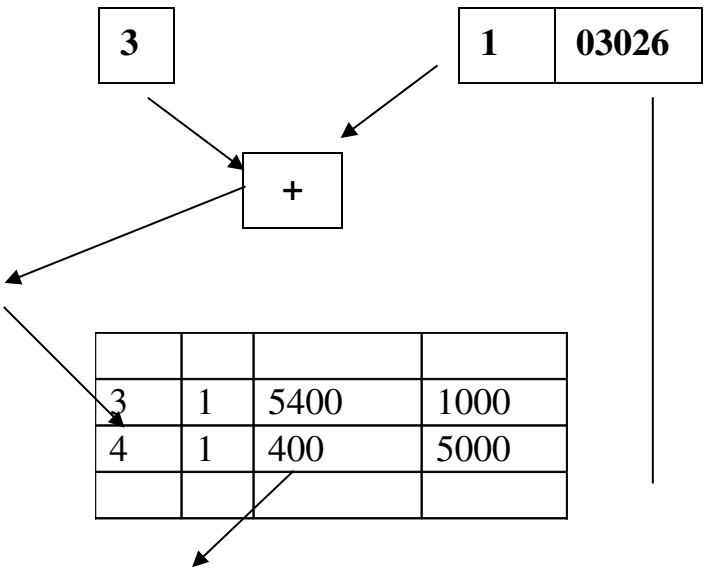
Giải thích: với chương trình hiện tại, thanh ghi bảng segment có giá trị 3 có nghĩa là chương trình này đòi hỏi các segment từ 3 trở di.

Khi xuất hiện việc hướng tới địa chỉ (1, 03026), cho phép định vị địa chỉ thực sự của đối tượng đang quan tâm.

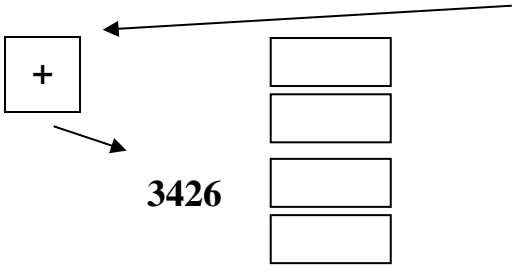
Phép cộng đầu tiên 3 với 1 cho số hiệu segment thực sự (là 4) trong hệ thống phân segment hiện có (trong bảng segment tổng thể).

Khi tra tới bản ghi có số hiệu nói trên thấy nội dung 1 400 4000 có nghĩa là: segment đang ở bộ nhớ trong, độ dài segment là 4000 và địa chỉ đầu là 400. Một phép cộng thứ 2 là 400+3026 cho địa chỉ thực hiện cần hướng tới là 3426.

(vẽ hình)



3	1	5400	1000
4	1	400	5000



Hình 3.8. Ví dụ hướng địa chỉ ảo

Trong ví dụ trên có thể đưa ra một số nhận xét sơ bộ sau:

Hệ thống chỉ cần quản lý bảng segment tổng thể mà mỗi chương trình chiếm một vùng con liên tục trong bảng tổng thể đó. Bảng segment chương trình người dùng được tính đến về mặt hình thức mà thực sự là hệ thống không quan tâm đến. Tuy nhiên hệ thống cần quản lý các vị trí đặt segment đầu tiên của chương trình người dùng trong bảng tổng thể: sử dụng thanh ghi chỉ số segment của chương trình người dùng.

Việc tham chiếu tới một địa chỉ liên quan tới hai phép cộng như trên.

Địa chỉ cần hướng tới (segment cần hướng tới) đang có mặt tại bộ nhớ trong. Trường hợp segment không tìm thấy trong bộ nhớ (còn gọi là thiếu vắng segment) được giải quyết nhờ cách thức sinh ra một ngắt để gọi một chương trình đặt segment (là chương trình phân phối bộ nhớ).

Chức năng của chương trình đặt segment:

1.Đưa một số segment ra bộ nhớ ngoài để giải phóng bộ nhớ (khi cần thiết).

2.Chuyển dịch CPU sang phục vụ chương trình khác vì chương trình này đang trong trạng thái chờ đợi.

3.Khi đọc segment vào bộ nhớ trong thì đồng thời thực hiện việc biến đổi phần tử của bảng segment: đầu segment và dấu hiệu bộ nhớ trong. Tồn tại những phương pháp xác định có phải "gỡ" segment nào đó ra ngoài và chất lượng các segment nào.

c. Ưu điểm của segment

Đảm bảo việc tải các segment (của nhiều chương trình) vào bộ nhớ trong không cần sự can thiệp của người dùng. Khi liên kết, chương trình LINK có thể không thiết lập cấu trúc với việc phủ và để cài đặt không cần chương trình tải hay supervisor phủ. Có hai chiến lược xây dựng cấu trúc chương trình cấu trúc tĩnh và cấu trúc động.

Cấu trúc tĩnh dùng trong trường hợp người lập trình mong muốn trong thời gian link các môdun đưa ra các segment cần đồng thời tìm thấy ở bộ nhớ trong. Cấu trúc tĩnh có điểm bất lợi do người lập trình không phòng ngừa hết các khả năng gọi lẫn nhau giữa segment. Mặt khác, có thể động chạm tới sử dụng đệ quy môdun. Không những thế,

cấu trúc tĩnh gặp trở ngại khi có sự phụ thuộc tương đối của chương trình vào dữ liệu.

Như vậy cần có giải pháp cấu trúc động cho phép hình thức hóa liên kết các segment theo dạng địa chỉ ảo.

Cho phép hiệu đính được liên kết. Khi hướng tới một segment, tên của nó được sử dụng ngay trong thời gian sử dụng: không phải ánh xạ mọi địa chỉ các segment khác nhau tới không gian địa chỉ thực.

Bộ nhớ được sử dụng khá hiệu quả.

3.3. Phân trang

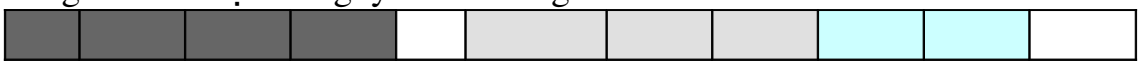
a.Điều khiển trang

Tổ chức trang là trường hợp đặc biệt của segment.

Tổ chức trang đơn giản hơn tổ chức segment: trang là các đơn vị nhớ đồng nhất cỡ. Không gian bộ nhớ ảo được chia thành các trang cùng cỡ, được đánh số để xác định. Địa chỉ trong chương trình trong điều khiển trang có dạng (p, i) với p là số hiệu của trang còn i là gia số so với đầu trang. Cỡ của trang là lũy thừa của 2. Địa chỉ ảo là một số: các bit già cho trang, các bit thấp là gia số. Không gian địa chỉ thực cũng được phân theo trang (trang vật lý cùng cỡ với trang ảo) với số hiệu trang f. Ánh xạ từ p vào f do chương trình điều khiển bộ nhớ đảm nhận.

Có một sự phân biệt giữa tổ chức trang với tổ chức segment: việc phân chia segment do người dùng đảm nhận còn việc phân chương trình ra thành các trang lại do chương trình dịch đảm nhận: trang tương ứng như cấu trúc lệnh hoặc dữ liệu.

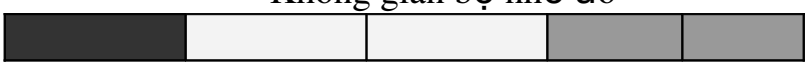
Khác với phân phối không gian bộ nhớ ảo cho segment, việc phân chia bộ nhớ ảo theo trang là không "tiết kiệm", mỗi chương trình người dùng chiếm một số nguyên các trang.



A0 A1 A2 A3 B0 B1 B2 C0 C1

Chương trình A chương trình B chương trình C

Không gian bộ nhớ ảo



A1 C0 B0 A3

Các trang trong bộ nhớ vật lý

Hình 3.9. Các chương trình trong không gian bộ nhớ ảo và đặt trang

Hiện nhiên số lượng các trang vật lý là tùy thuộc vào dung tích bộ nhớ trong và cỡ của trang trong khi đó số lượng trang ảo là không hạn

chế. Trang ảo nằm ở bộ nhớ trong hoặc trên đĩa từ. Trên đĩa từ, chúng cần phải ghi nhận trên những vùng bộ nhớ liên tục song với BNT không đòi hỏi. Cũng như segment, các trang của một chương trình không đòi hỏi một vùng nhớ liên tục. Ví dụ xem hình 3.15.

Phổ biến hệ thống sử dụng tổ chức trang dùng bộ nhớ ngoài, tuy vậy cũng có hệ thống sử dụng bộ nhớ trong.

b.Cài đặt

Chú ý trường hợp sử dụng bộ nhớ ngoài. Để tương ứng giữa trang ảo và trang vật lý sử dụng bảng trang, mỗi phần tử gồm có hai trường: dấu hiệu và chỉ số trang vật lý (nếu ở bộ nhớ trong). Thanh ghi trang, chứa địa chỉ bảng trang của chương trình hiện tại. Tương tự như segment, có chương trình đặt trang, một thành phần của phân phối bộ nhớ.

Chương trình đặt trang có chức năng

Tìm vị trí đặt trang (có sự thay đổi nội dung phần tử tương ứng trong bảng trang).

Chuyển CPU cho chương trình khác, chương trình này về trạng thái chờ đợi. Quá trình tính toán địa chỉ (p, i) được biểu thị bằng một số sau khi được tách có (71,638) và thanh ghi trang chương trình hiện tại là 550. Tương tự như đã làm ở segment, trang cần tìm kiếm có chỉ số 621 (550+71). Trong bảng trang tổng quát, phần tử 621 có giá trị (1,24) biểu thị rằng trang nói trên đang đặt trong bộ nhớ trong tại trang vật lý 24.

Giả sử độ dài của trang là 1000 vậy địa chỉ đầu trang 24 sẽ là 24000 và địa chỉ cần truy nhập sẽ là 24638 (24000+638). Hình vẽ 3.15 thể hiện quá trình đã được xem xét.

a. Chiến lược đặt trang

Chiến lược đặt trang định hướng tới việc làm tối thiểu sự vắng trang trong bộ nhớ trong. Chiến lược đầu tiên là tải các trang trước khi sử dụng sẽ loại trừ được việc vắng trang. Đơn giản nhất là không sử dụng bộ nhớ ngoài như là sự mở rộng bộ nhớ trong nữa. Có những hệ thống sử dụng hình thức này. Mặt khác, trong chế độ đa chương trình cần đảm bảo điều kiện: không bắt buộc chương trình này phải đợi sự hoàn thiện của chương trình khác.

Chiến lược thứ hai cho phép không phải toàn bộ trang bộ nhớ trong: người dùng sử dụng bộ nhớ ảo mà dung tích vượt quá bộ nhớ trong hoặc hệ thống đảm bảo đa chương trình với số lượng lớn công việc.

Nguyên lý đặt trước (tương tự như buffer theo khẳng định: nghiên cứu từ những năm 1960) dựa trên trình bày liên kết trong chương trình.

Nguyên lý đặt trang theo đòi hỏi chỉ đặt khi trang được thực tế hướng đến (buffer theo đòi hỏi: theo những năm 1970).

b. Chiến lược giải phóng trang

Tồn tại một số chiến lược chọn trang để giải phóng. Tương ứng với đặt trang theo đòi hỏi thì cần tối thiểu đặt/giải phóng trang cũng định hướng tới việc tối thiểu tình trạng thiếu vắng trang, hay cũng vậy, tối thiểu sự trao đổi ngoài/trong.

Trong vấn đề giải phóng trang có thể chọn:

- *Cơ chế FIFO (First In First Out):*

Trang nào được đưa vào bộ nhớ trong sớm nhất sẽ được giải phóng để nhường chỗ cho việc nạp một trang mới vào.

Để giải thích xét một ví dụ về lời gọi hướng tới các trang như dưới đây:

144	A1	144	263	144	168	144	A1	179	A1	A2	263

Trong trường hợp này, các trang được hướng địa chỉ theo thứ tự từ trái sang phải và đáp ứng sự hướng đến này, có các trang không nạp vào do đã được nạp sẵn (các ô bị bôi sẫm màu). Dòng tương ứng dưới đây cho biết tình trạng các trang bị giải phóng khỏi bộ nhớ khi cần phải nạp trang mới vào. Ví dụ, tại thời điểm cần nạp trang 168, trong bộ nhớ đã có các trang 144, A1, 263 trong đó 144 là trang nạp vào đầu tiên nên bị giải phóng ra khỏi bộ nhớ trong. Việc loại bỏ các trang A1, 263…tiếp theo là hoàn toàn tương tự.

					144	A1	263	168		144	A1

- *Cơ chế LRU (least-recent-used):*

Định hướng tới việc làm tối thiểu số lần loại bỏ và nạp trang, cơ chế FIFO cần phải cải tiến để nhận được các cơ chế có hiệu quả hơn và một trong các cơ chế như thế là cơ chế LRU. Cơ chế LRU sử dụng một stack để kiểm tra xem trang thực sự đang nằm trong bộ nhớ trong mà thời gian chưa có sự hướng địa chỉ tới là dài nhất. Xét trường hợp tương tự như ở ví dụ trước, khi cần nạp trang 168 vào bộ nhớ trong, tuy trang 144 được nạp vào sớm nhất nhưng sau đó đã có 2 sự hướng địa chỉ tới. Trong khi đó, trang A1 đang tồn tại ở bộ nhớ trong mà có thời gian lâu nhất chưa có sự hướng tới nên thuật toán LRU sẽ chọn giải phóng A1 thay vì cho giải phóng 144.

					A1		263	168		144	A1

Có thể nêu sơ lược về tư tưởng của LRU là chọn các trang ít thường xuyên hướng tới nhất để loại và hy vọng là đã giữ lại bộ nhớ trong những trang thường xuyên hơn thì như vậy việc trao đổi trong/ngoài là ít nhất có thể được. Có thể thấy nói chung LRU tốt hơn FIFO, chẳng hạn ở ví dụ cụ thể trên, FIFO mất 6 lần loại trang, còn LRU chỉ mất có 5 lần.

Cơ chế LFU (least frequently used)

Định hướng theo thời đoạn ngắn hay dài. Tương tự như LRU song tính toán tần suất có sự hướng tới ít nhất trong một khoảng thời gian đủ lâu nào đó.

Một số cơ chế cho điều khiển trang

Bộ nhớ cho điều khiển trang: bộ nhớ lưu giữ bảng map (bảng đồ) trang tổng thể (không gian bộ nhớ ảo): giả sử trang ảo lên đến 16MB, với trường hợp mỗi trang có độ dài 4KB, thì phải dùng tới 4K phần tử; trong trường hợp mỗi trang có độ dài 256B thì phải dùng 64K phần tử…

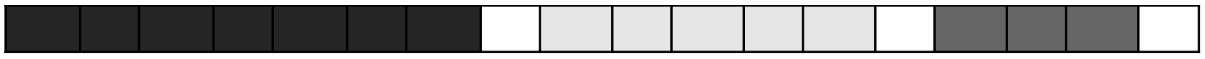
Thanh ghi bảng trang cho quá trình hiện tại: thanh ghi bảng trang cho trong chương trình hiện tại.

3.4. Kết hợp phân đoạn và phân trang

Điều khiển trang thuận tiện, dễ thể hiện, song mắc một nhược điểm là nếu độ dài của trang quá bé thì tăng trao đổi vào – ra, còn nếu độ dài của trang lớn có thể gây ra lãng phí cả về trao đổi và bộ nhớ.

Điều khiển theo segment có tính linh hoạt hơn về độ dài các segment song do độ dài đa dạng cũng tạo ra phức tạp trong thực hiện việc điều phối bộ nhớ.

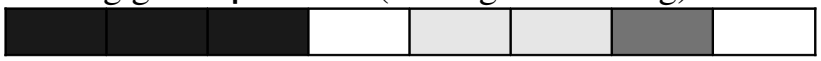
Giải pháp trộn (trang – segment) cố gắng phát huy ưu điểm từ trong các giải pháp nói trên.



A0 A1 A2 A3 B0 B1 B2 C0 C1

Chương trình A chương trình B chương trình C

Không gian bộ nhớ ảo (các segment –trang)

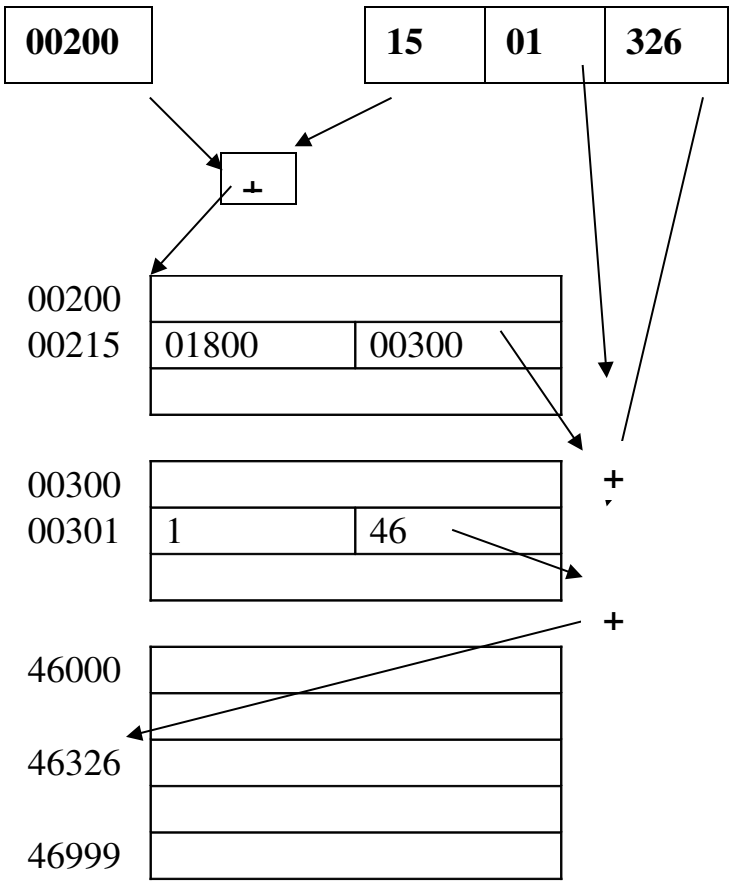


A1 B0 C1

Các segment – trang trong bộ nhớ vật lý

Hình 3.10. Phân phối trên bộ nhớ ảo và bộ nhớ thực trong chế độ segment – trang.

Trong giải pháp trang- segment, gia cố d trong cặp (s, d) được thay thế bởi cặp (p, i) trong giải pháp trang. Địa chỉ ảo (s, d) được thay thế bởi bộ ba (s, p, i) trong đó mỗi segment sẽ bao gồm một số nguyên các trang: phần tử chỉ không những segment mà còn cả trang trong segment đó (độ dài segment tính theo trang mà không theo byte). Phần tử để tham chiếu trong trường hợp này không phải là segment mà là bảng segment: nó chỉ dẫn đến bảng này và độ dài hiện tại của segment tính theo trang. Một vài công việc có thể cùng sử dụng một segment. Có thể biểu diễn hướng tới bộ nhớ như dưới đây.



Hình 3.11. Ví dụ trong điều khiển bộ nhớ segment- trang

Như hình vẽ 3.17, sự hướng tới bộ nhớ qua ba giai đoạn:

Đầu tiên tới bảng segment: thanh ghi segment của chương trình cộng với chỉ số segment trong địa chỉ (15+200=215).

Chỉ số trang trong địa chỉ với trang bắt đầu của segment 215 (00300+1=00301)

Chỉ số trong nội tại trang: 46*1000+326=46326

CÂU HỎI VÀ BÀI TẬP

1. So sánh sự khác biệt giữa địa chỉ vật lý và địa chỉ logic
2. Giả sử bộ nhớ trong được chia thành các vùng nhớ có kích thước 600Kb, 500Kb, 200Kb, 300Kb, (theo thứ tự), cho biết các chương trình có kích thước 212Kb, 417Kb, 112Kb, 426Kb (theo thứ tự) sẽ được cấp phát bộ nhớ như thế nào nếu sử dụng phương pháp First – Fit và Best Fit. Phương pháp nào cho phép sử dụng bộ nhớ hiệu quả.
3. Trình bày điều khiển bộ nhớ vật lý theo chiến lược cận cố định.Cho

ví dụ minh họa.

4. Trình bày điều khiển bộ nhớ vật lý theo chiến lược cận thay đổi.Cho ví dụ minh họa.
5. Trình bày điều khiển bộ nhớ logic theo cấu trúc Overlay.Cho ví dụ minh họa.
6. Trình bày điều khiển bộ nhớ theo kĩ thuật phân đoạn
7. Trình bày điều khiển bộ nhớ theo kĩ thuật phân trang

HƯỚNG DẪN TRẢ LỜI

1. Dựa vào khái niệm địa chỉ vật lý và địa chỉ logic để phân biệt
2. Sắp xếp các vùng nhớ rỗi theo thứ tự sau đó đưa các chương trình lần lượt vào bộ nhớ theo các phương pháp First Fit là chọn cái đầu tiên (chọn vùng nhớ đầu tiên đủ để chứa chương trình), Best Fit là chọn cái tốt nhất (chọn vùng nhớ đủ chứa chương trình nhưng có độ dư thừa là ít nhất).
3. Dựa vào phần điều khiển bộ nhớ chiến lược giới hạn tĩnh để trình bày và cho ví dụ.
4. Dựa vào phần điều khiển bộ nhớ chiến lược giới hạn động để trình bày và cho ví dụ.
5. Overlay là chương trình chia thành các modul nhỏ và một số modul sẽ dùng chung vùng nhớ. Vẽ cây chương trình .
6. Dựa vào phần điều khiển bộ nhớ theo phân đoạn ở giáo trình để trình bày tóm tắt kĩ thuật này.
7. Dựa vào phần điều khiển bộ nhớ theo phân trang ở giáo trình để trình bày tóm tắt kĩ thuật này.

CHƯƠNG 4: ĐIỀU KHIỂN CPU, ĐIỀU KHIỂN QUÁ TRÌNH

Chương 4: **Điều khiển CPU, điều khiển quá trình Mã chương**:..M4

Mục tiêu:

- Nắm

Sau khi học xong bài học này, sinh viên có khả năng:

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.

1. Các khái niệm cơ bản

Mục tiêu: nắm được khái niệm quá trình, quan hệ giữa các quá trình.

1.1.khái niệm quá trình

Công việc không thể được tiến hành nếu nó không được bộ xử lý tiếp nhận và thực hiện: bộ xử lý là một tài nguyên của hệ thống được sử dụng để hoàn thành công việc. Có thể coi chương trình cần thực hiện như một *quá trình* (các hệ điều hành khác nhau có thể sử dụng các thuật ngữ khác nhau cùng nghĩa với thuật ngữ quá trình, mà phổ biến hơn cả là các thuật ngữ tiến trình, bài toán): *quá trình* là đối tượng được tiếp nhận bởi bộ xử lý (D.L. Parnas, 1974). Cần phân biệt khái niệm quá trình với khái niệm chương trình: quá trình là một lần thực hiện một chương trình nào đó kể từ khi bắt đầu cho đến khi kết thúc. Ví dụ, cùng một lúc trong chế độ đa người dùng, có ba người dùng đều gọi chương trình dịch ngôn ngữ C: hệ thống chỉ có 1 chương trình C, trong khi đó tại thời điểm đang xét có 3 quá trình đang tồn tại và đang được điều phối CPU.

Việc điều phối CPU xảy ra chỉ trong chế độ đa chương trình. Trong một số hệ thống máy tính, người ta còn phân biệt trạng thái của CPU: trạng thái bài toán (trạng thái người dùng) hay trạng thái SUPERVISOR (trạng thái kiểm tác) mà điều cốt lõi là trong trạng thái bài toán, không cho phép thực hiện một số lệnh đặc biệt của CPU. Việc phân biệt trạng thái của CPU cho phép phân loại các quá trình theo mức độ thâm nhập hệ thống và như vậy vấn đề an toàn và bảo vệ hệ thống được thuận lợi hơn. Chương trình người dùng chỉ thâm nhập sâu hệ thống chỉ thông qua các chương trình của hệ điều hành.

Tương tự như trong điều phối bộ nhớ người ta quan tâm đến khái niệm bộ nhớ ảo, trong đó đã mở rộng không gian bộ nhớ trong thành không gian bộ nhớ ảo: các chương trình "đang cập nhật" địa chỉ trên một miền bộ nhớ mở rộng và có sự chuyển đổi từ bộ nhớ mở rộng tới bộ nhớ trong để thực hiện; quan niệm không gian bộ nhớ ảo là đáp ứng cho mọi người sử dụng máy. Chế độ đa chương trình, người sử dụng quan niệm rằng các chương trình " đang được thực hiện" song thực tế CPU của máy tại mỗi thời điểm chỉ phục vụ một chương trình và như vậy ta chỉ có 1 bộ xử lý thực (cho chương trình đã nói); các chương trình đồng thời còn lại "hiện đang" sử dụng CPU "ảo". Tốc độ làm việc của CPU ảo là "nhỏ hơn" tốc độ làm việc của CPU thực sự.

Nếu quan niệm như trên, mỗi quá trình được coi là chiếm giữ CPU suốt trong quá trình thực hiện của mình, do vậy, cần có sự phân biệt khi nào chiếm giữ CPU thực sự và khi nào chiếm giữ CPU ảo.

1.2. Quan hệ giữa các quá trình

Các quá trình đồng hành thực thi trong hệ điều hành có thể là những quá trình độc lập hay những quá trình hợp tác. Một quá trình là độc lập (independent) nếu nó không thể ảnh hưởng hay bị ảnh hưởng bởi các quá trình khác thực thi trong hệ thống. Rõ ràng, bất kỳ một quá trình không chia sẻ bất cứ dữ liệu (tạm thời hay cố định) với quá trình khác là độc lập. Ngược lại, một quá trình là hợp tác (cooperating) nếu nó có thể ảnh hưởng hay bị ảnh hưởng bởi các quá trình khác trong hệ thống. Hay nói cách khác, bất cứ quá trình chia sẻ dữ liệu với quá trình khác là quá trình hợp tác.

Chúng ta có thể cung cấp một môi trường cho phép hợp tác quá trình với nhiều lý do:

- Chia sẻ thông tin: vì nhiều người dùng có thể quan tâm cùng phần thông tin (thí dụ, tập tin chia sẻ), chúng phải cung cấp một môi trường cho phép truy xuất đồng hành tới những loại tài nguyên này.
- Gia tăng tốc độ tính toán: nếu chúng ta muốn một tác vụ chạy nhanh hơn, chúng ta phải chia nó thành những tác vụ nhỏ hơn, mỗi tác vụ sẽ thực thi song song với các tác vụ khác. Việc tăng tốc như thế có thể đạt được

  chỉ nếu máy tính có nhiều thành phần đa xử lý (như các CPU hay các kênh

  I/O).
- Tính module hóa: chúng ta muốn xây dựng hệ thống trong một kiểu mẫu dạng module, chia các chức năng hệ thống thành những quá

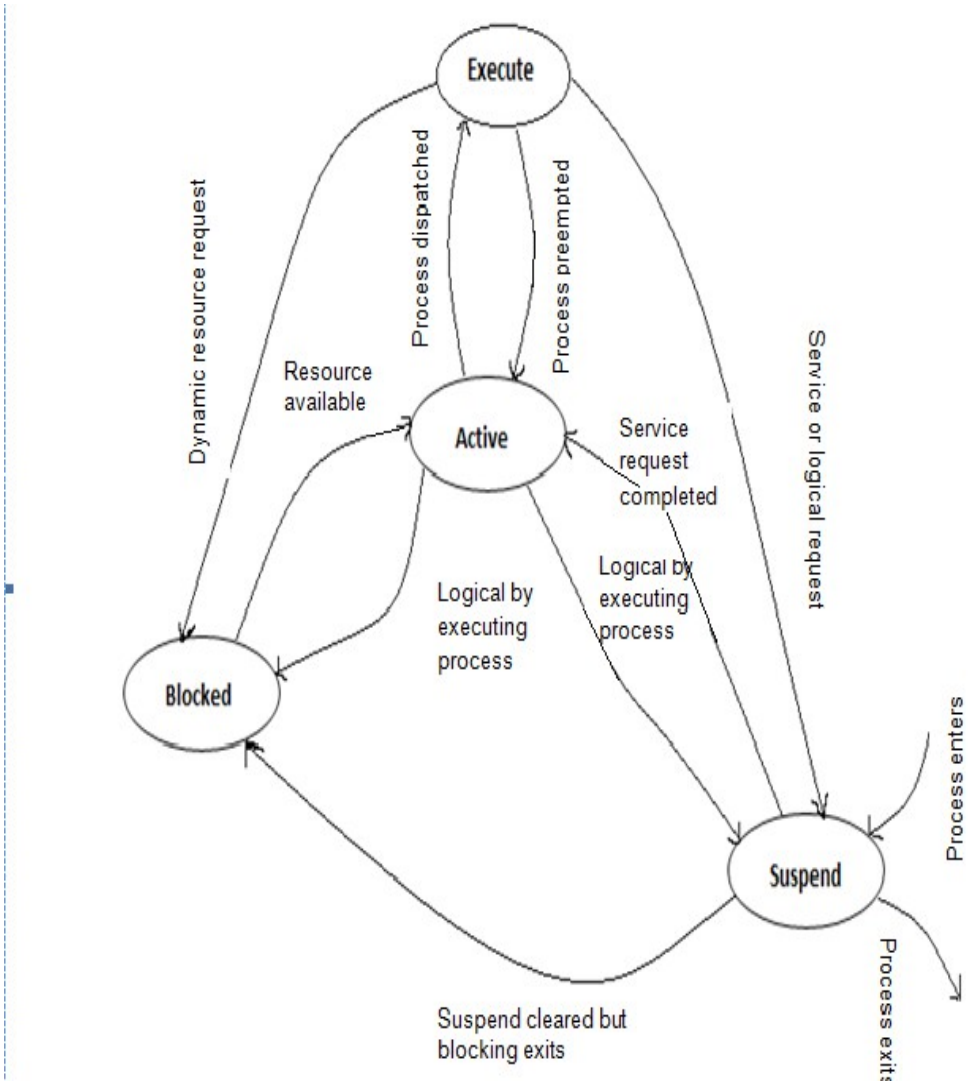
trình hay luồng như đã thảo luận ở chương trước.

- Tính tiện dụng: Thậm chí một người dùng đơn có thể có nhiều tác vụ thực hiện tại cùng thời điểm. Thí dụ, một người dùng có thể đang soạn thảo, in, và biên dịch cùng một lúc.

2. Trạng thái của quá trình

Mục tiêu : Nắm được từng trạng thái của quá trình.
Hiểu được sơ đồ không gian trạng thái

2.1.Sơ đồ không gian trạng thái (SNAIL)



Hình 4.1. Sơ đồ không gian trạng thái SNAIL

Tại thời điểm bắt đầu của một bộ xử lý, ít nhất 1 quá trình có thể thực hiện lệnh của mình (nó đang được phân phối CPU). Quá trình này nằm trong trạng thái sử dụng (hay trong trạng thái thực hiện-running), nó đang chiếm hữu CPU thực. Quá trình để có thể đi tới được trạng thái sử dụng chỉ khi nó đang ở trạng thái *chuẩn bị* (chuẩn bị được sử dụng, còn gọi là trạng thái sẵn sàng-ready). Các quá trình ở trạng thái chuẩn bị được coi là đã được cung cấp đầy đủ các nhu cầu khác: về bộ nhớ và các tài nguyên khác để có thể thực hiện được và nó chỉ chờ đợi một tài nguyên duy nhất đó là CPU. Khi quá trình trong trạng thái sử dụng đòi hỏi tài nguyên khác CPU, nó rơi vào trạng thái *chờ đợi* (còn được gọi là trạng thái kết khối) vì đang được kết khối (chờ đợi tài nguyên); nó chưa thể rơi vào trạng thái chuẩn bị vì tài nguyên cần thiết chưa có. Sự thiếu vắng như thế có thể kể đến: vắng segment hay trang, thao tác vào/ra, quá trình được phát sinh bao gồm cả nhận tín hiệu terminal khi truyền tin.

2.2. Một số khối điều khiển quá trình

Để chuyển trạng thái của một quá trình, hệ thống cần quản lý một số thông tin về nó: mô tả quá trình. Mô tả quá trình đối với các trạng thái khác nhau sẽ theo các phương pháp khác nhau. Thông thường người ta sử dụng dòng xếp hàng cho các mô tả đó, gọi tên là hàng đợi dù trong trường hợp chung không hoạt động theo đúng nguyên tắc của dòng xếp hàng (FIFO).

Trong một số hệ điều hành, đối với mỗi quá trình có khối điều khiển quá trình (Process control Block-viết tắt PCB) còn đối với một số hệ điều hành khác, khối tương ứng được gọi là khối điều khiển bài toán (Task Control Block-TCB) gắn với quá trình đó, là phần tử của dòng xếp hàng nói trên.

Nội dung của PCB (hay TCB) gồm toàn bộ hay bộ phận các thông tin được nêu dưới đây:

-Tên chỉ số quá trình;

-Độ ưu tiên của quá trình;

-Trạng thái của quá trình: chờ đợi (kết khối), sẵn sàng (chuẩn bị) hay sử dụng (thực hiện);

-Thông tin thời gian;

-Trạng thái phần cứng (các thanh ghi và cờ)

-Thông tin lập lịch và tình trạng sử dụng (ví dụ thời gian dự kiến quá trình thực hiện v.v…);

-Thông tin quản lý bộ nhớ (thanh ghi, bảng .v..v)

-Tình trạng vào – ra (thiết bị, thao tác.v.v…)

-Thông tin quản lý File;
-Thông tin thống kê (chẳng hạn thời gian quá trình đã thực hiện trong bộ nhớ trong…)

Các PCB (TCB) của các quá trình tồn tại trong máy tính liên kết trong một hay một số dòng xếp hàng để điều phối CPU sử dụng để chọn quá trình nào để phân phối CPU.

Chức năng của điều khiển CPU (điều phối CPU cho các quá trình):

-Phân phối và phân phối lại bộ xử lý thực;

-Tách ra bộ xử lý ảo (không có phân phối lại).

Chức năng của điều phối chính: một thành phần cơ bản của điều phối quá trình có tên là điều phối chính. Chức năng của điều phối chính là lên phương án (chọn công việc). Với mỗi công việc được chọn, điều phối chính sẽ tạo ra một quá trình được gói và đưa nó vào trạng thái *chuẩn bị*. Điều phối chính cũng thực hiện chức năng liên quan đến hoàn thiện quá trình (xem hình trên: giai đoạn từ dòng xếp hàng vào đi tới trạng thái chuẩn bị (được gọi là giai đoạn phát sinh khởi tạo quá trình) và giai đoạn từ trạng thái sử dụng đi ra dòng xếp hàng ra (được gọi là giai đoạn kết thúc – hoàn thiện) do điều phối chính đảm nhận). Như vậy, chức năng của điều phối chính: điều phối chính đảm bảo việc điều phối quá trình ở mức độ chung nhất còn chuyển trạng thái của một quá trình do một chương trình có tên là điều phối là supervisor hay *monitor*. Người ta cũng sử dụng một số thuật ngữ khác.

-Nếu sử dụng khởi tạo thì kết thúc sẽ giải phóng bộ xử lý ảo

-Điều phối chính mức trên cho việc giải phóng bộ xử lý ảo thì điều phối mức dưới cho giải phóng bộ xử lý thực.

Một đặc điểm phân biệt hai điều phối: với mọi công việc, điều phối chính chỉ thực hiện một lần trong khi đó điều phối có thể thực hiện nhiều lần.

3. 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.

3.1. Nguyên tắc chung

Điều phối chọn trong quá trình đang có mặt trong hàng đợi, ở trạng thái sẵn sàng và có độ ưu tiên cao nhất. Tồn tại rất nhiều quan điểm liên quan đến việc xác định độ ưu tiên, chẳng hạn: thời điểm tạo ra quá trình, thời điểm xuất hiện công việc, thời gian phục vụ, thời gian đã dành cho phục vụ, thời gian trung bình quá trình chưa được phục vụ v.v… Các yếu tố này được tính toán, đánh giá theo các phương pháp khác nhau và do đó tồn tại nhiều nguyên tắc điều phối khác nhau.

Tiêu chuẩn chọn một cách thức điều phối CPU là: cần chú ý tới việc nó ảnh hưởng như thế nào tới *thời gian chờ đợi xử lý*, tức là thời gian chi phí của quá trình đó trong trạng thái chuẩn bị tới trạng thái sử dụng. Đối với người dùng, các kiểu chờ đợi sau đây của quá trình trong hệ thống là không phân biệt:

-Thời gian trong trạng thái chuẩn bị;

-Thời gian trong trạng thái kết khối;

-Thời gian quá trình trong đầu vào chờ đợi tài nguyên.

Như đã nói, có nhiều nguyên lý để điều phối; ở đây chỉ xem xét những nguyên lý chung và phổ biến nhất cũng như khảo sát những chiến lược để cài đặt các nguyên lý trên.

3.2. Các trình lập lịch (long term, short term)

Định thời biểu dài (long-term scheduling) (hay định thời biểu công việc) là chọn các quá trình được phép cạnh tranh CPU. Thông thường, định thời biểu dài bị ảnh hưởng nặng nề bởi việc xem xét cấp phát tài nguyên, đặc biệt quản lý bộ nhớ.

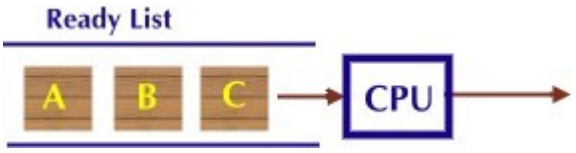
Định thời ngắn (short-term scheduling) là sự chọn lựa một quá trình từ các hàng đợi sẵn sàng.

4. Các thuật toán lập lịch

Mục tiêu:Nắm được các giải pháp lập lịch mà hệ điều hành thực hiện nhằm điều phối các quá trình được thực hiện trên CPU.

4.1. First Come First Served (FCFS)

Tiến trình nào có yêu cầu sử dụng CPU trước thì sẽ được thực hiện trước.
Ưu điểm là thuật thoán đơn giản nhất
Nhược điểm la hiệu quả thuật toán phụ thuộc vào thứ tự của các tiến trình trong hàng đợi.



Hình 4.2 .hàng đợi FCFS

sử có 3 tiến trình P1 , P2 , P3 với thời gian thực hiện tương ứng là 24ms, 3ms, 6ms
Giả sử ba tiến trình xếp hàng theo thứ tự P1, P2, P3
Thời gian chờ các tiến trình là:
P1chờ 0ms, P2 chờ 24ms, P3 chờ 27ms
Thời gian chờ trung bình: (0+24+27)/3=17ms

Thời gian chờ trung bình không đạt cực tiểu, và biến đổi đáng kể đối với các giá trị về thời gian yêu cầu xử lý và thứ tự khác nhau của các tiến trình trong danh sách sẵn sàng. Có thể xảy ra hiện tượng tích lũy thời gian chờ, khi các tất cả các tiến trình (có thể có yêu cầu thời gian ngắn) phải chờ đợi một tiến trình có yêu cầu thời gian dài kết thúc xử lý.

Giải thuật này đặc biệt không phù hợp với các hệ phân chia thời gian, trong các hệ này, cần cho phép mỗi tiến trình được cấp phát CPU đều đặn trong từng khoảng thời gian.

4.2. Shortest Job First (SJF)

Nguyên tắc : Đây là một trường hợp đặc biệt của giải thuật điều phối với độ ưu tiên. Trong giải thuật này, độ ưu tiên *p* được gán cho mỗi tiến trình là nghịch đảo của thời gian xử lý *t* mà tiến trình yêu cầu : $p = 1/t$. Khi CPU được tự do, nó sẽ được cấp phát cho tiến trình yêu cầu ít thời gian nhất để kết thúc- tiến trình ngắn nhất. Giải thuật này cũng có thể độc quyền hay không độc quyền. Sự chọn lựa xảy ra khi có một tiến trình mới được đưa vào danh sách sẵn sàng trong khi một tiến trình khác đang xử lý. Tiến trình mới có thể sỡ hữu một yêu cầu thời gian sử dụng CPU cho lần tiếp theo (CPU-burst) ngắn hơn thời gian còn lại mà tiến trình hiện hành cần xử lý. Giải thuật SJF không độc quyền sẽ dừng hoạt động của tiến trình hiện hành, trong khi giải thuật độc quyền sẽ cho phép tiến trình hiện hành tiếp tục xử lý. Nếu hai tiến trình có cùng thời gian sử dụng CPU, tiến trình đến trước sẽ đươc yêu cầu CPU trước.

Ví dụ :

Tiến trình	Thời điểm vào RL	Thời gian xử lý
P1	0	6
P2	1	8
P3	2	4
P4	3	2

Sử dụng thuật giải SJF độc quyền, thứ tự cấp phát CPU như sau:

P1	P4	P3	P2
0	6	8	12 20

Sử dụng thuật giải SJF không độc quyền, thứ tự cấp phát CPU như sau:

P1	P4	P1	P3	P2
0	3	5	8	12 20

Thảo luận : Giải thuật này cho phép đạt được thời gian chờ trung bình cực tiểu. Khó khăn thực sự của giải thuật SJF là không thể biết được thời gian yêu cầu chu kỳ CPU tiếp theo? Chỉ có thể dự đoán giá trị này theo cách tiếp cận sau : gọi *tn* là độ dài của thời gian xử lý lần thứ *n,* t_{n+1} là giá trị dự đoán cho lần xử lý tiếp theo. Với hy vọng giá trị dự đoán sẽ gần giống với các giá trị trước đó, có thể sử dụng công thức:

$$t_{n+1} = a\, tn + (1-a)\, t_n$$

Trong công thức này,tn chứa đựng thông tin gần nhất ; t_n chứa đựng các thông tin quá khứ được tích lũy. Tham số a (0 <= a <= 1) kiểm soát trọng số của hiện tại gần hay quá khứ ảnh hưởng đến công thức dự đoán.

4.3. Shortest Remain Time (SRT)

Nguyên tắc : Mỗi tiến trình được gán cho một độ ưu tiên tương ứng, tiến trình có độ ưu tiên cao nhất sẽ được chọn để cấp phát CPU đầu tiên. Các tiến trình có độ ưu tiên bằng nhau thì tiến trình nào đến trước thì sẽ được cấp trước.Độ ưu tiên có thể được định nghĩa nội tại hay nhờ vào các yếu tố bên ngoài. Độ ưu tiên nội tại sử dụng các đại lượng có thể đo lường để tính toán độ ưu tiên của tiến trình, ví dụ các giới hạn thời gian, nhu cầu bộ nhớ…Độ ưu tiên cũng có thể được gán từ bên ngoài dựa vào các tiêu chuẩn do hệ điều hành như tầm quan trọng của tiến trình, loại người sử dụng sỡ hữu tiến trình…

Giải thuật điều phối với độ ưu tiên có thể theo nguyên tắc độc quyền hay không độc quyền. Khi một tiến trình được đưa vào danh sách các tiến trình sẵn sàng, độ ưu tiên của nó được so sánh với độ ưu tiên của tiến trình hiện hành đang xử lý. Giải thuật điều phối với độ ưu tiên và không độc quyền sẽ thu hồi CPU từ tiến trình hiện hành để cấp phát cho tiến trình mới nếu độ ưu tiên của tiến trình này cao hơn tiến trình hiện hành. Một giải thuật độc quyền sẽ chỉ đơn giản chèn tiến trình mới vào danh sách sẵn sàng, và tiến trình hiện hành vẫn tiếp tục xử lý hết thời gian dành cho nó.

Ví dụ : (độ ưu tiên 1 > độ ưu tiên 2> độ ưu tiên 3)

Tiến trình	**Thời điểm vào RL**	**Độ ưu tiên**	**Thời gian xử lý**
P1	**0**	**3**	**24**
P2	**1**	**1**	**3**
P3	**2**	**2**	**3**

Sử dụng thuật giải độc quyền, thứ tự cấp phát CPU như sau :

P1	**P2**	**P3**
0	**‘24**	**27 30**

Sử dụng thuật giải không độc quyền, thứ tự cấp phát CPU như sau :

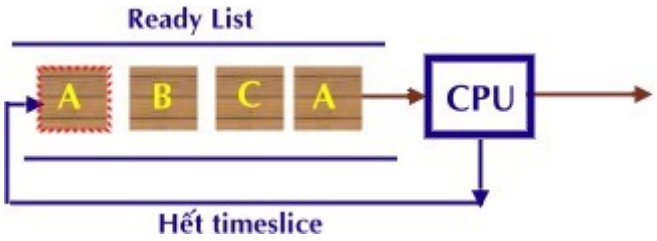
P1	**P2**	**P3**	**P1**
0	**‘1**	**4**	**7 30**

Thảo luận : Tình trạng ‘đói CPU’ (starvation) là một vấn đề chính yếu của các giải thuật sử dụng độ ưu tiên. Các giải thuật này có thể để các tiến trình có độ ưu tiên thấp chờ đợi CPU vô hạn ! Để ngăn cản các tiến trình có độ ưu tiên cao chiếm dụng CPU vô thời hạn, bộ điều phối sẽ giảm dần độ ưu tiên của các tiến trình này sau mỗi ngắt đồng hồ. Nếu độ ưu tiên của tiến trình này giảm xuống thấp hơn tiến trình có độ ưu tiên cao thứ nhì, sẽ xảy ra sự chuyển đổi quyền sử dụng CPU.Quá trình này gọi là sự ‘lão hóa’ *(aging)* tiến trình.

4.4. Round Robin (RR)

Nguyên tắc : Danh sách sẵn sàng được xử lý như một danh sách vòng, bộ điều phối lần lượt cấp phát cho từng tiến trình trong danh sách một khoảng thời gian tối đa sử dụng CPU cho trước gọi là *quantum*. Tiến trình đến trước thì được cấp phát CPU trước. Đây là một giải thuật điều phối không độc quyền : khi một tiến trình sử dụng CPU đến hết thời gian quantum dành cho nó, hệ điều hành thu hồi CPU và cấp cho tiến trình kế tiếp trong danh sách. Nếu tiến trình bị khóa hay kết thúc trước khi sử dụng hết thời gian quantum, hệ điều hành cũng lập tức cấp phát CPU cho tiến trình khác. Khi tiến trình tiêu thụ hết thời gian CPU dành cho nó mà chưa hoàn tất, tiến trình được đưa trở lại vào cuối danh sách sẵn sàng để đợi được cấp CPU trong lượt kế tiếp.

Ví dụ :



Hình 4.3 Điều phối Round Robin

Tiến trình	Thời điểm vào RL	Thời gian xử lý
P1	0	24
P2	1	3
P3	2	3

Nếu sử dụng quantum là 4 milisecondes, thứ tự cấp phát CPU sẽ là

P1	P2	P3	P1	P1	P1	P1	P1
0	‘4	7	10	14	18	22	26 30

Thời gian chờ đợi trung bình sẽ là (0+6+3+5)/3 = 4.66 milisecondes.

Nếu có *n* tiến trìh trong danh sách sẵn sàng và sử dụng quantum *q*, thì mỗi tiến trình sẽ được cấp phát CPU *1/n* trong từng khoảng thời gian *q*.

Mỗi tiến trình sẽ không phải đợi quá *(n-1)q* đơn vị thời gian trước khi nhận được CPU cho lượt kế tiếp.

Vấn đề đáng quan tâm đối với giải thuật RR là độ dài của quantum. Nếu thời lượng quantum quá bé sẽ phát sinh quá nhiều sự chuyển đổi giữa các tiến trình và khiến cho việc sử dụng CPU kém hiệu qủa. Nhưng nếu sử dụng quantum quá lớn sẽ làm tăng thời gian hồi đáp và giảm khả năng tương tác của hệ thống.

4.5. Multi Level Queue (MLQ)

Để phân lớp các quá trình đang trong trạng thái chuẩn bị và chọn lựa quá trình chuyển sang trạng thái sử dụng có thể sử dụng các thông tin được cho bằng người tạo ra quá trình đó và các thông tin nhận được trong việc điều phối các quá trình. Các thông tin này có thể là:

- Thông tin có sẵn, đã cho trước;
- Thời gian sử dụng thực tế;
- Số nhu cầu vào-ra đã tiến hành…

Với hệ thống tổ chức trang bộ nhớ, tiện lợi nhất là sử dụng một số dòng xếp hàng khác nhau để phân biệt các quá trình ở trạng thái đặt/tách trang với các quá trình chờ đợi sự kết thúc vào/ra.

- Đầu tiên, CPU có quá trình của dòng đợi có độ ưu tiên cao nhất. Quá trình trong mỗi hàng đợi có một lượng tử thời gian: nếu trong thời đoạn của lượng tử thời gian đó nó không hoàn thiện thì nó được xếp vào cuối cùng trong hàng đợi với độ ưu tiên ngay sát nó (ngay cả khi nó đòi hỏi một thời gian nào đó trong trạng thái kết khối). Chỉ có quá trình rơi vào dòng đợi với độ ưu tiên thấp nhất là hoạt động theo chế độ vòng còn các hàng đợi khác hoạt động theo kiểu FCFS.
- Ý nghĩa lôgic của điều phối kiểu này là ở chỗ quá trình đòi hỏi thời gian lâu hơn sẽ kết thúc muộn hơn theo xác xuất. Sự điều phối đa mức đã xem xét với sự liên kết ngược sẽ hiệu quả trong điều kiện tốc độ hoàn thiện của quá trình giảm đi theo lượng thời gian nó đã được phục vụ.

4.6. Multi Level Feedback Queues (MLFQ)

Nguyên tắc : Ý tưởng chính của giải thuật là phân lớp các tiến trình tùy theo độ ưu tiên của chúng để có cách thức điều phối thích hợp cho từng nhóm. Danh sách sẵn sàng được phân tách thành các danh sách riêng biệt theo cấp độ ưu tiên, mỗi danh sách bao gồm các tiến trình có cùng độ ưu tiên và được áp dụng một giải thuật điều phối thích hợp để điều phối.

Ngoài ra, còn có một giải thuật điều phối giữa các nhóm, thường giải thuật này là giải thuật không độc quyền và sử dụng độ ưu tiên cố định.Một tiến trình thuộc về danh sách ở cấp ưu tiên *i* sẽ chỉ được cấp

phát CPU khi các danh sách ở cấp ưu tiên lớn hơn *i* đã trống.

Hình 4.4 Điều phối nhiều cấp ưu tiên

Thông thường, một tiến trình sẽ được gán vĩnh viễn với một danh sách ở cấp ưu tiên *i* khi nó được đưa vào hệ thống. Các tiến trình không di chuyển giữa các danh sách. Cách tổ chức này sẽ làm giảm chi phí điều phối, nhưng lại thiếu linh động và có thể dẫn đến tình trạng 'đói CPU' cho các tiến trình thuộc về những danh sách có độ ưu tiên thấp. Do vậy có thể xây dựng giải thuật điều phối nhiều cấp ưu tiên và xoay vòng. Giải thuật này sẽ chuyển dần một tiến trình từ danh sách có độ ưu tiên cao xuống danh sách có độ ưu tiên thấp hơn sau mỗi lần sử dụng CPU. Cũng vậy, một tiến trình chờ quá lâu trong các danh sách có độ ưu tiên thấp cũng có thể được chuyển dần lên các danh sách có độ ưu tiên cao hơn. Khi xây dựng một giải thuật điều phối nhiều cấp ưu tiên và xoay vòng cần quyếtđịnh các tham số :

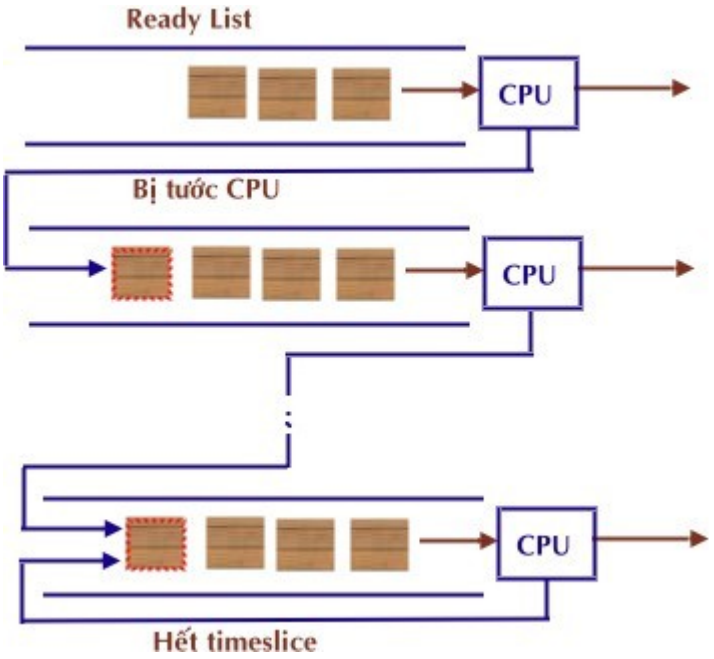
Số lượng các cấp ưu tiên

Giải thuật điều phối cho từng danh sách ứng với một cấp ưu tiên.

Phương pháp xác định thời điểm di chuyển một tiến trình lên danh sách có độ ưu tiên cao hơn.

Phương pháp xác định thời điểm di chuyển một tiến trình lên danh sách có độ ưu tiên thấp hơn.

Phương pháp sử dụng để xác định một tiến trình mới được đưa vào hệ thống sẽ thuộc danh sách ứng với độ tiên nào.



Hình 4.5 Điều phối Multilevel Feedback

5. Hệ thống ngắt

Mục tiêu : Nắm được khái niệm ngắt, phân loại ngắt
Nắm được quy trình xử lý ngắt.

5.1. Khái niệm ngắt

Tồn tại mối quan hệ giữa các bộ phận trong hệ điều hành, ví dụ: *điều phối,* thực hiện quá trình và hệ thống con vào – ra. Thông thường khi hết hạn lượng tử thời gian hay hoàn thiện vào/ra nảy sinh ngắt. Ngắt sinh ra những sự kiện khác và xử lý ngắt là những phương tiện quan trọng của điều khiển CPU. Xem xét chương trình thực hiện các lệnh một cách tuần tự, trong đó có lệnh chuyển điều khiển vô điều kiện và có điều kiện. Ngắt có thể được xác định như là một chương trình gắn vào truyền điều khiển cho một chương trình khác thực hiện tại thời điểm ngắt. Ngắt được coi như cách thức truyền điều khiển cho quá trình xử lý ngắt chưa được biết từ quá trình bị ngắt.

Ngắt được phân chia ra hai lớp cơ bản: ngắt trong và ngắt ngoài.

-Ngắt trong liên quan đến các sự kiện liên kết tới công việc của CPU và để đồng bộ hoạt động của nó. Ví dụ: tràn ô khi cộng hay trừ dấu phẩy động, xuất hiện phép chia cho 0; thực hiện phép toán dấu phẩy động truyền hoặc xóa phần bậc; vi phạm địa chỉ bộ nhớ, thiếu vắng segment hoặc trang, mã lệnh sai…

-Ngắt ngoài: được xảy ra theo các hiện tượng liên quan ngoài sử thực hiện của CPU: ngắt vào-ra, ngắt do sơ đồ kiểm tra, ngắt từ CPU khác, ngắt do hết lượng tử thời gian.v.v…

5.2. Xử lý ngắt

Như vậy, ngắt là một hiện tượng xảy ra có thể độc lập với sự làm việc của CPU. Một vấn đề được đặt ra là thời điểm xử lý ngắt: xử lý ngắt lúc nào là thích hợp nhất khi quan hệ với lệnh máy đang thực hiện. Ngắt xảy ra có thể hoặc do sự thực hiện lệnh, hoặc do tác động từ chính bản thân lệnh. Nếu cơ chế xử lý ngắt không thích hợp sẽ loại bỏ chính lệnh máy đang thực hiện. Thuận lợi hơn cả là xử lý ngắt sau khi thực hiện lệnh và việc ghi nhận ngắt là độc lập với sự thực hiện lệnh. Cơ chế ghi nhận ngắt là nằm ngoài các chương trình xử lý ngắt.

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

1. Tại những ô nhớ quy định, ghi nhận các đặc trưng của số hiệu ngắt vừa phát sinh (tùy thuộc vào số liệu được đưa vào ô nhớ tương ứng). Ví dụ với máy IBM 360-370 có các số hiệu để phân biệt các kiểu ngắt như sau:
   -Ngắt vào-ra
   -Ngắt theo chương trình: vi phạm cách thức phương tiện máy: lệnh không chính quy; dữ liệu không chính quy;
   -Ngắt hướng tới supervisor (gọi chương trình supervisor và thay chế độ làm việc của CPU);
   -Ngắt ngoài: có tín hiệu hướng tới CPU, ngắt theo thời gian, ngắt khi có tín hiệu của các bộ xử lý khác…;
   -Ngắt theo sơ đồ kiểm tra.
2. Ghi nhớ trạng thái của quá trình bị ngắt: giá trị bộ đếm lệnh (chú ý từ trạng thái chương trình PSW: Program Status Word, trên bàn điều khiển có một hàng đèn tương ứng với từ máy…)
3. Thanh ghi địa chỉ lệnh hướng tới địa chỉ để xử lý ngắt.
4. Ngắt được xử lý.
5. Quay lại quá trình đã bị ngắt (nếu được)

Các bước 1-3 do các thành phần chức năng của máy tính đảm nhận, bước 4-5 do chương trình xử lý ngắt đảm nhận.

Bước 4. chương trình xử lý ngắt tiến hành các công việc:

Ghi nhớ bổ sung một số thông tin mà do cách thức phương tiện (bước 2) chưa ghi hết, ví dụ, bước 2 ghi PSW còn chương trình xử lý ngắt phải bảo vệ trạng thái của quá trình bị ngắt bằng việc lưu trữ hệ thống các thanh ghi chung và công việc nói trên

đòi hỏi một vùng bộ nhớ nhất định (chẳng hạn, với IBM, EC đòi hỏi vùng 72 bytes cho 16 thanh và 2 địa chỉ chuyển đổi).

Định danh chương trình xử lý ngắt.

Thông tin bước 3 là bộ phận đối với chương trình xử lý ngắt: mỗi loại ngắt có thể do một chương trình ngắt riêng, ví dụ ngắt do vào ra (thiết lập cách thức phương tiện ở bước 1) khác biệt hoàn toàn với ngắt hướng tới supervisor (phân tích tác động tiếp theo supervisor).

- Thực hiện tác động tương ứng với ngắt đã được định danh.

Các tác động này hết sức đơn giản. Ví dụ, chỉ thiết lập dấu hiệu nào đó như trạng thái tràn ô, hoặc quay lại băng từ chuyển sang việc chuẩn bị đọc nếu đã đọc sai.v.v…

Nếu không quá gấp, chương trình xử lý ngắt tương ứng sẽ được ghi vào dòng xếp hàng quá trình ở trạng thái chuẩn bị.

Chương trình xử lý ngắt đảm bảo việc quay về trạng thái bình thường của CPU (chọn quá trình người dùng để thực hiện) tùy thuộc vào:

-Kiểu ngắt;

-Kiểu của chương trình điều phối CPU được sử dụng.

Từ các yếu tố trên sẽ xác định công việc kết khối, về trạng thái chuẩn bị và các công việc được chọn tiếp theo…

Chú ý:

Một số tác động của chương trình xử lý ngắt được thực hiện chậm nếu để ở bộ nhớ ngoài cho nên đưa ra giải pháp một số bộ phận của chương trình xử lý ngắt được đặt thường trực trong bộ nhớ trong như là một phần trong nhân hệ thống. Nếu chương trình xử lý ngắt quá lớn, nó được chia làm hai phần: phần thường trực và phần không thường trực.

Nhiều ngắt có quan hệ đến điều khiển CPU (ngắt theo thời gian, ngắt theo hoạt động thiết bị, ngắt hoàn thiện vào/ra). Quá trình do điều phối làm không chỉ là quá trình người dùng mà còn là những bộ phận khác nhau của hệ điều hành (bao hàm chương trình xử lý ngắt mức 2; chương trình con thống kê; điều phối chính; tải và thậm chí chính cả điều phối).

Ngắt đa mức

Ngắt xảy ra có thể đối với chương trình người dùng, có thể xảy ra chính trong quá trình đang xử lý ngắt. Đây là tình huống được gọi là ngắt đa mức. Xử lý ngắt đa mức ra sao?

-Phân cấp các loại ngắt theo độ ưu tiên, thông thường ngắt liên quan tới cách thức kĩ thuật có độ ưu tiên thấp hơn so với các ngắt có liên quan đến hệ điều hành. Ví dụ: ngắt gọi supervisor có độ ưu tiên cao hơn so với ngắt vào/ra.

-Chọn ngắt nào được xử lý trước tiên: ngắt cũ và ngắt mới, việc đó tùy thuộc vào kiểu của hai ngắt. Ngắt mới hoặc được giải quyết ngay (ngắt trội hơn), hoặc bị hủy bỏ, hoặc chờ để giải quyết tiếp theo.

Xử lý ngắt đa mức theo các độ ưu tiên khác nhau được đảm bảo theo các cách thức phương tiện khác nhau ghi nhận mỗi kiểu ngắt khác nhau trên các ô nhớ khác nhau.

6. Hiện tượng bế tắc

Mục tiêu : Nắm được khái niệm bế tắc, các biện pháp phòng tránh, xử lý bế tắc.

6.1. Khái niệm bế tắc

Bế tắc là hiện tượng khi một nhóm các quá trình bị kết khối một cách lâu dài do mỗi quá trình trong nhóm đang chiếm một tập con các tài nguyên để hoàn thiện quá trình đó và chờ đợi việc giải phóng một số tài nguyên còn lại đang bị các quá trình thuộc cùng nhóm đang chiếm giữ.

Một trong các ví dụ dễ thấy là hiện tượng yêu cầu chu trình các thiết bị: có 3 quá trình, A đang chiếm giữ thiết bị và đòi hỏi thiết bị 2, B đang chiếm giữ thiết bị 2 và đòi hỏi thiết bị 3, C đang chiếm giữ thiết bị 3 và đòi hỏi thiết bị 1.

Chúng ta có hai quá trình Pr1 và Pr2. Chúng chia xẻ hai tài nguyên p1 và p2 và để loại trừ ràng buộc, trong hai quá trình trên sử dụng semaphore s1cho tài nguyên p1 và semaphore s2 cho tài nguyên p2. Việc sử dụng các semaphore nói trên trong thân các thủ tục được trình bày như dưới đây.

Pr1:…	Pr2:…
1.P(s1)	5.P(s2)
…	….
2.P(s2)	6.P(s1)
…	….
3.V(s1)	7.V(s1)
…	…
4.V(s2)	8.V(s2)
….	…

Ban đầu, cả hai semaphore có giá trị 1. Xem xét trường hợp theo thời gian, dãy trạng thái đòi hỏi nhu cầu và giải phóng biến chung là dãy 1,2,5,3,4,6,7. Khi thủ tục Pr2 có nhu cầu p2 (lệnh 5), nó bị kết khối do p2 đang được Pr1 dùng. Chỉ sau khi Pr1 thực hiện giải phóng p2 (lệnh 4) thì Pr2 mới được tách khối. Đến bây giờ dãy thực hiện các câu lệnh trở thành 5,1,6,2 thế thì Pr2 bị kết khối khi đòi hỏi p1 (lệnh 6) còn Pr1 bị kết khối khi đòi hỏi p2 (lệnh 2): Pr1 chờ đợi cho đến khi Pr2 đi tới

lệnh 8 còn Pr2 chờ đợi cho đến khi Pr1 đi tới lệnh 3. Hiện tượng bế tắc xuất hiện do hai quá trình con này chờ đợi lẫn nhau.

6.2. Các biện pháp phòng tránh bế tắc

Chủ yếu có ba hương tiếp cận để xử lý tắc nghẽn :

- Sử dụng một vài giao thức (protocol) để bảo đảm rằng hệ thống không bao giờ xảy ra tắc nghẽn.

HĐH không có khả năng chống Deadlock

Lý do dung phương pháp này:

Do xác suất xảy ra đealock nhỏ

Giải quyết deadlock đòi hỏi chi phí cao

Xử lý bằng tay do người quản trị hệ thống làm.

Đây là giải pháp của hầu hết các hệ điều hành hiện nay.

- Cho phép xảy ra tắc nghẽn và tìm cách sữa chữa tắc nghẽn.

- Hoàn toàn bỏ qua việc xử lý tắc nghẽn, xem như hệ thống không bao giờ xảy ra tắc nghẽn.

6.3. Phát hiện bế tắc

Một câu hỏi đặt ra có thể tính toán để khẳng định được hay không khẳng định rằng một quá trình có thể rơi vào tình trạng bế tắc. Để tính toán được điều này, hệ điều hành cần đưa ra danh sách các tài nguyên mà các quá trình đang chờ đợi và danh sách các quá trình đang chờ đợi tài nguyên mà không được thỏa mãn.

Để đoán nhận việc bế tắc có thể xảy ra hay không cần có thông tin để kiểm soát nhu cầu tài nguyên của các quá trình. Rõ ràng là không phải tình trạng cần tài nguyên là sẽ xảy ra bế tắc. Có một số thuật toán dựa vào các danh sách đã liệt kê ở trên để đoán nhận được bế tắc có thể xảy ra để loại bỏ .

6.4. Xử lý bế tắc

Đình chỉ hoạt động của các tiến trình liên quan

Cách tiếp cận này dựa trên việc thu hồi lại các tài nguyên của những tiến trình bị kết thúc. Có thể sử dụng một trong hai phương pháp sau :

Đình chỉ tất cả các tiến trình trong tình trạng tắc nghẽn

Đình chỉ từng tiến trình liên quan cho đến khi không còn chu trình gây tắc nghẽn : để chọn được tiến trình thích hợp bị đình chỉ, phải dựa vào các yếu tố như độ ưu tiên, thời gian đã xử lý, số lượng tài nguyên đang chiếm giữ , số lượng tài nguyên yêu cầu...

Thu hồi tài nguyên

Có thể hiệu chỉnh tắc nghẽn bằng cách thu hồi một số tài nguyên từ các tiến trình và cấp phát các tài nguyên này cho những tiến trình khác cho đến khi loại bỏ được chu trình tắc nghẽn. Cần giải quyết 3 vấn đề sau:

Chọn lựa một nạn nhân: tiến trình nào sẽ bị thu hồi tài nguyên ? và thu hồi những tài nguyên nào ?

Trở lại trạng thái trước tắc nghẽn: khi thu hồi tài nguyên của một tiến trình, cần phải phục hồi trạng thái của tiến trình trở lại trạng thái gần nhất trước đó mà không xảy ra tắc nghẽn.

Tình trạng « đói tài nguyên »: làm sao bảo đảm rằng không có một tiến trình luôn luôn bị thu hồi tài nguyên ?

6.5. Kết luận chung về phòng tránh bế tắc

Trạng thái deadlock xảy ra khi hai hay nhiều quá trình đang chờ không xác định một sự kiện mà có thể được gây ra chỉ bởi một trong những quá trình đang chờ. Về nguyên tắc, có ba phương pháp giải quyết deadlock:

- Sử dụng một số giao thức để ngăn chặn hay tránh deadlock, đảm bảo rằng hệ thống sẽ không bao giờ đi vào trạng thái deadlock.
- Cho phép hệ thống đi vào trạng thái deadlock, phát hiện và sau đó

phục hồi.

- Bỏ qua vấn đề deadlock và giả vờ deadlock chưa bao giờ xảy ra trong hệ thống. Giải pháp này là một giải pháp được dùng bởi hầu hết các hệ điều hành bao gồm UNIX.

Trường hợp deadlock có thể xảy ra nếu và chỉ nếu bốn điều kiện cần xảy ra cùng một lúc trong hệ thống: loại trừ hỗ tương, giữ và chờ cấp thêm tài nguyên,
không đòi lại tài nguyên, và tồn tại chu trình trong đồ thị cấp phát tài nguyên. Để ngăn chặn deadlock, chúng ta đảm bảo rằng ít nhất một điều kiện cần không bao giờ xảy ra.

Một phương pháp để tránh deadlock mà ít nghiêm ngặt hơn giải thuật ngăn chặn deadlock là có thông tin trước về mỗi quá trình sẽ đang dùng tài nguyên như thế nào. Thí dụ, giải thuật Banker cần biết số lượng tối đa của mỗi lớp tài nguyên có thể được
yêu cầu bởi mỗi quá trình. Sử dụng thông tin này chúng ta có thể định nghĩa giải thuật tránh deadlock.

Nếu hệ thống không thực hiện một giao thức để đảm bảo rằng deadlock sẽ không bao giờ xảy ra thì lược đồ phát hiện và phục hồi phải được thực hiện. Giải thuật phát hiện deadlock phải được nạp lên để xác định deadlock có thể xảy ra hay không. Nếu deadlock được phát hiện hệ thống phải phục hồi bằng cách kết thúc một số quá trình bị deadlock hay đòi lại tài nguyên từ một số quá trình bị deadlock.

Trong một hệ thống mà nó chọn các nạn nhân để phụv hồi về trạng thái trước đó chủ yếu dựa trên cơ sở yếu tố chi phí, việc đói tài nguyên có thể xảy ra. Kết quả là quá trình được chọn không bao giờ hoàn thành tác vụ được chỉ định của nó.

CÂU HỎI VÀ BÀI TẬP

1. Nêu khái niệm quá trình (tiến trình). Phân biệt quá trình với chương

trình.

2. Vẽ sơ đồ không gian trạng thái. Nêu ý nghĩa các trạng thái của một quá trình.

3. Thế nào là lập lịch dài kỳ và lập lịch ngắn kỳ.

4. Khái niệm ngắt và qui trình xử lý ngắt.

5. Nêu các tiêu chuẩn lập lịch cho CPU.

6. Cho các quá trình với thời gian thực hiện tương ứng như sau:

Quá trình (process)	$t_{thực\ hiện}$
P1	10
P2	2
P3	7
P4	1
P5	5

Tính thời gian chờ đợi trung bình của các quá trình trong các chiến lược FCFS, SJN, RR (với lượng tử thời gian là 2).

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

HƯỚNG DẪN TRẢ LỜI

1. Tiến trình là một đoạn chương trình hay đoạn dữ liệu chương trình được đưa vào CPU để xử lý. Dựa vào khái niệm chương trình và tiến trình để phân biệt.
2. Vẽ sơ đồ. Nêu lên khi nào thì quá trình ở các trạng thái trong sơ đồ.
3. Nêu ở phần lập lịch dài kỳ và lập lịch ngắn kỳ.
4. Ở phần ngắt và các bước xử lý ngắt trong giáo trình
5. Xem xét số tiến trình vào xử lý trong CPU, thời gian chờ của các tiến trình.
6. Áp dụng và xem ví dụ các chiến lược FCFS, SJN,RR nói ở trên để giải.
7. Ở phần bế tắc và các điều khiện xảy ra bế tắc.

CHƯƠNG 5: HỆ ĐIỀU HÀNH ĐA XỬ LÝ

Chương 5: **Hệ điều hành đa xử lý** **Mã chương**:..**M5**

Mục tiêu:
Sau khi học xong bài học này, sinh viên có khả năng:
- 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. Hệ điều hành đa xử lý tập trung

Mục tiêu: Hiểu khái quát được xu thế sử dụng hệ thống đa xử lý

1.1 Hệ thống đa xử lý

a. Hệ thống nhiều CPU

Hiện nay, từ sự phát triển với tốc độ 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. Về vấn đề này, nảy sinh giải pháp theo hai hướng:

Giải pháp *tăng cường năng lực của một CPU riêng* cho từng máy máy tính: công nghệ vi mạch ngày càng phát triển vì vậy năng lực của từng CPU cũng ngày nâng cao, các dự án các vi mạch VLSI với hàng triệu, hàng chục triệu transitor. Tuy nhiên giải pháp này cũng nảy sinh những hạn chế về kĩ 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 của CPU là giải pháp *liên kết nhiều CPU để* tạo ra một hệ thống chung có năng lực đáng kể: việc đưa 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.

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.

Trong chương này, xem xét việc 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 hoặc nhiều máy tính trong một hệ thống thống nhất. Gọi chung các hệ có nhiều CPU như vậy là *hệ đa xử lý.*

b.Phân loại các hệ đa xử lý

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

Ví dụ về hệ đa xử lý tập trung là tập 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;

Ví dụ về hệ đa xử lý phân tán là các mạng máy tính: mạng gồm nhiều máy tính liên kết và được đặt ở những vị trí khác nhau, với một khoảng cách có thể coi là xa tùy ý.

Phân loại theo đặc tính của các CPU thành phần: hệ đã xử lý thuần nhất hoặc hệ đa xử lý không thuần nhất v.v…

Một ví dụ dễ quen thuộc là trong các máy vi tính từ 80486 trở đi trong đó có hai CPU (80x86 và 80x87) là hai CPU không thuần nhất.

Siêu máy tính ILLIAC-IV gồm nhiều CPU có đặc trưng giống nhau là một ví dụ về thuần nhất.

Phân loại theo cách các CPU thành phần tiếp nhận và xử lý dữ liệu. Trong cách phân loại này bao gồm cả những máy tính đơn xử lý thông thường:

- *Đơn câu lệnh, 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 câu lệnh được thực hiện.
- *Đơn câu lệnh, đa dữ liệu (SIMD: single instruction multiple data):*

Các bộ xử lý trong cùng một nhịp thời gian chỉ thực hiện cùng một câu lệnh. Có thể lấy ví dụ từ việc cộng hai vector cho trước: Các CPU thành phần đều thực hiện các phép cộng; đổi số tương ứng đã có từng CPU; sau đó, chọn tiếp lệnh (chỉ thị) mới để điều khiển công việc này. Thông thường có một hệ chọn câu lệnh chung và mọi CPU thành phần cùng thực hiện: siêu máy tính ILLIAC-IV sử dụng cách thức này, có một máy tính con có tác dụng lưu giữ hệ điều hành để điều khiển ILLIAC.IV (bộ xử lý ma trận).

- Đa câu lệnh, 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 một cách tuần tự: output của bộ xử lý này là input của bộ xử lý tiếp theo (ví dụ CRAY-1: Bộ xử lý vector). Các CPU kết nối theo kiểu này được gọi là kết nối "dây chuyền".

- Đa dữ liệu, đa câu lệnh (MIMD)

Mỗi bộ xử lý có bộ phân tích chương trình riêng; câu lệnh và dữ liệu do chính mỗi CPU phải đảm nhận; có thể hình dung các CPU này hoạt động hoàn toàn "độc lập nhau". Các hệ điều hành mạng, hệ điều hành phân tán là những ví dụ về đa dữ liệu, đa câu lệnh.

Trong nội dung ở chương này, xem xét cách phân loại dạng tập trung/phân tán song thực chất chỉ quan tâm đến hệ đa xử lý tập trung còn với hệ đa xử lý phân tán, sẽ có những chuyên đề riêng đáp ứng.

Chú ý, một xu thế nghiên cứu và triển khai các hệ thống tính toán đa xử lý thời sự là nghiên cứu về tính toán cụm trong đó các mô hình SIMD, MISD và MIMD tương ứng được phát triển.

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, Hitachi và các máy tính nhiều xử lý hiện nay (máy tính của khoa CNTT, trường ĐHKHTN-ĐHQGHN có hai bộ xử lý). 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.

a.Phân phối bộ nhớ

Các quá trình xuất hiện trong bộ nhớ chung. Việc phân phối bộ nhớ được tiến hành cho 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 việc với bộ nhớ (bài toán xử lý con trỏ ngoài v.v.) có thể gắn với mỗi CPU một cache nhớ. Phân ra hai loại thâm nhập cache: tĩnh và động. Thâm nhập tĩnh: mỗi CPU chỉ thâm nhập cache tương ứng, không thâm nhập dữ liệu tại vùng cache của các CPU khác. Thâm nhập động cho phép CPU của máy này có thể thâm nhập các cache của CPU khác.

b.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 được 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.

Về dòng xếp hàng có thể xem xét theo hai mô hình dưới 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 semaphore nhiều giá trị để phân phối CPU cho các quá trình này).

2. Hệ điều hành đa xử lý phân tán

Mục tiêu: hiểu được những nét cơ bản về hệ điều hành đa xử lý phân tán nhằm trang bị khả năng tự nghiên cứu trong tương lai.

2.1. Giới thiệu hệ phân tán

Trong phần phân loại hệ thống đa xử lý, chú ý cách phân loại theo vị trí đặt các CPU (tập trung và phân tán) thì hệ phân tán được xây dựng từ các " máy tính" rời rạc nhau: mỗi vị trí là một máy tính nguyên vẹn, có đầy đủ chức năng xử lý, lưu trữ và truyền dữ liệu.

Hệ tập trung cho phép xử lý song song theo thao tác hoặc theo quá trình, trong khi đó, hệ phân tán chỉ có thể xử lý song song theo quá trình: các quá trình con được xử lý trên các máy tính khác nhau. Việc phân chia quá trình cho các CPU thành phần hoặc theo chức năng của CPU đó (server/client) hoặc theoo một lịch được phân công của một hệ thống chung.

Do phân tán nên vấn đề truyền dẫn dữ liệu đóng vai trò quan trọng trong các hệ phân tán. Đây cũng là một trong những lí do điển hình nhất để cách thức xử lý song song trên các hệ phân tán là theo quá trình mà không phải theo phép toán.

2.2. Đặc điểm hệ phân tán

Hệ thống phân tán (kéo theo sự hình thành các hệ điều hành phân tán) được phát sinh do các nhu cầu hết sức tự nhiên về việc nâng cao năng lực tài nguyên hệ thống (sức mạnh của hệ thống tính toán và cơ sở dữ liệu chung v.v..). Giải pháp phân tán có tác dụng phát huy năng lực chung của toàn bộ hệ thống khi giải quyết bài toán với kích thước bài toán tăng lên và vẫn đảm bảo hoạt động bình thường của các máy tính thành viên. Hệ thống phân tán được thiết lập hoặc là một hệ thống mới hoàn toàn được thiết kế theo mô hình phân tán hoặc xây dựng một hệ phân tán dựa trên các tài nguyên địa phương (máy tính, cơ sở dữ liệu) sẵn có.

Một trong các trường hợp điển hình, các hệ phân tán được dùng để quản trị các hệ thống cơ sở dữ liệu lớn. Trong các hệ cơ sở dữ liệu phân tán, tính dư thừa thông tin lại được quan tâm chú ý không chỉ tới khía cạnh gây khó khăn khi tính đến tính nhất quán dữ liệu mà còn tới khía cạnh thuận lợi về vần đề an toàn: lưu trữ kép (ngoài bản chính còn một số bản sao) để có thể phục hồi khi xảy ra sự cố đối với hệ thống. Để đảm bảo tính nhất quán của hệ thống định kỳ "làm tươi" các thông tin do hệ thống quản lý.

Như đã biết, bài toán lập lịch cho hệ thống chung là phức tạp ngay cả đối với máy tính với một CPU, vì vậy trong các hệ phân tán, bài toán nói trên là hết sức phức tạp (ngay cả các hệ đồng nhất) cho nên người ta thường chọn các phương án đơn giản nhất.

Các nội dung kiến thức về hệ thống phân tán được trình bày chi tiết hơn trong giáo trình chuyên đề" mạng và các hệ phân tán". Bản chất của hệ điều hành trong các mô hình phân tán là một hệ điều hành đa chương trình. Do tính chất không thuần nhất của các máy tính địa phương và có liên quan chặt chẽ đến đường truyền thông, bài toán lập lịch và các hệ thống chương trình điều khiển là phức tạp. Các thuật toán điều khiển được chọn lựa là đủ đơn giản và vẫn luôn là bài toán thời sự đang được nghiên cứu.

CÂU HỎI VÀ BÀI TẬP

1. Trình bày mục đích của hệ nhiều CPU.
2. Hệ phân tán là gì?
3. Đặc điểm hệ phân tán.

HƯỚNG DẪN TRẢ LỜI

1. Tăng cường năng lực của CPU là giải pháp *liên kết nhiều CPU để* tạo ra một hệ thống chung có năng lực đáng kể: việc đưa xử lý song song tạo ra nhiều lợi điểm.

2. Hệ phân tán tập hợp các máy tính ghép nối với nhau bằng đường truyền theo một tiêu chuẩn qui định trước.

3. Đặc điểm hệ phân tán: tạo khả năng làm việc phân tán, nâng cao việc khai thác và xử lý dữ liệu, tăng độ tin cậy của hệ thống, chia sẻ tài nguyên.

TÀI LIỆU THAM KHẢO

[1]. TS Hà Quang Thụy, ***Giáo trình Nguyên lý các hệ điều hành*** **, NXB KH & KT, 2005.**

[2]. Trần Hồ Thủy Tiên, ***Nguyên lý hệ điều hành,*** **Đại học Đà Nẵng, Năm 2007.**

[3]. Đặng Vũ Tùng, ***Giáo trình Nguyên lý hệ điều hành,*** **Nhà xuất bản Hà Nội, 2005.**

[4]. James R.Pinkert, ***Operating systems,*** **California State University.**