* 1. **3 mục đích chính của hệ điều hành là gì?**

Trả lời :

* Để cung cấp môi trường cho người sd máy tính để thực hiện các chương trình trên phần cứng của máy tính 1 cách thuận tiện và hiệu quả.
* Để phân bổ các tài nguyên của máy tính khi cần thiết để giải các vấn đề đưa ra. Quá trình phân bổ phải công bằng và hiệu quả nhất có thể.
* Là một chương trình điều khiển, nó phục vụ 2 chức năng chính: 1 là giám sát việc thực hiện các chương trình của người dùng để ngăn ngừa lỗi và việc sử dụng máy tính không phù hợp, và (2) quản lí hoạt động và việc điều khiển các thiết bị vào ra.
  1. **Những điểm khác biệt giữa hệ điều hành của máy tính lớn và máy tính cá nhân ?**

Trả lời : Nói chung , hệ điều hành cho hệ thống hàng loạt ( batch systems) có yêu cầu đơn giản hơn máy tính cá nhân. Hệ thống hàng loạt không cần phải quan tâm đến việc tương tác với người dùng nhiều như máy tính cá nhân. Do đó, hệ điều hành cho PC cần phải được quan tâm với thời gian phản hồi cho người dùng tương tác. Hệ thống hàng loạt thì không có những yêu cầu như vậy. 1 hệ thống hàng loạt đơn thuần cũng có thể không cần phải xử lí việc chia sẻ thời gian, trong khi hệ điều hành phải nhanh chóng chuyển đồi giữa các công vc khác nhau.

* 1. **Liệt kê 4 bước cần thiết để chạy 1 chương trình trên máy tính chuyên dụng** .

Trả lời: a. Giữ gìn/ Dự trữ thời gian máy chạy

b. Tải thủ công chương trình vào bộ nhớ máy

c. Tải địa chỉ bắt đầu và bắt đầu thực hiện chương trình

d. Theo dõi và điều khiển việc thực hiện chương trình từ bàn điều khiển.

* 1. Chúng ta đã nhấn mạnh sự cần thiết của 1 hệ điều hành để sử dụng hiệu quả phần cứng máy tính. Khi nào là thích hợp để hệ điều hành loại bỏ nguyên tắc này và “lãng phí” tài nguyên. Tại sao hệ thống như vậy không thực sự lãng phí?

Trả lời: Các hệ thống một người dùng nên tối đa hoá việc sử dụng hệ thống cho người dùng. GUI (Graphics User Interface) (giao diện đồ hoạ người dùng) có thể “lãng phí” chu kì của CPU, nhưng nó tối ưu hoá hoạt động tương tác của người dùng với hệ thống.

* 1. **Cái khó khăn chính mà một lập trình viên phải vượt qua trong việc viết hệ điều hành cho môi trường thời gian thực.**

Trả lời: Khó khăn chính là giữ cho hệ điều hành bên trong ràng buộc thời gian cố định của một hệ thống thời gian thực. Nếu hệ thống không hoàn thành một nhiệm vụ trong một khung thời gian nhất định, nó có thể gây ra sự cố của toàn bộ hệ thống mà nó đang chạy. Do đó khi viết một hệ điều hành cho 1 hệ thống thời gian thực, người vt chương trình phải chắc chắn rằng việc lập kế hoạch của mình không cho phép thời gian phản hồi vượt quá giới hạn thời gian.

* 1. **Xem xét các định nghĩa khác nhau của hệ điều hành. Xem xét liệu hđh có nên bao gồm các ứng dụng như trình duyệt Web và các chương trình thư hay không? Lập luận cả 2 ý kiến trên liệu có hay không, và đưa ra ý kiến của bạn**.

Trả lời:

Quan điểm agree: Các ứng dụng như trình duyệt web và công cụ email đóng một vai trò ngày càng quan trọng trong hệ thống máy tính để bàn. Để hoàn thành vai trò này, chúng phải đc kết hợp như là 1 phần của hđh. Bằng cách đó, chúng cung cấp hiệu suất tốt hơn và tích hợp tốt hơn với phần còn lại của hệ thống. Thêm vào đó, các ứng dụng quan trọng này có thể có giao diện giống như phần mềm hđh.

Đối quan điểm- disagree: Vai trò thiết yếu của HĐH là quản lí tài nguyên hệ thống như là CPU, bộ nhớ, I/O devices, vv. Ngoài ra, vai trò của nó là chạy các ứng dụng phần mềm như là trình duyệt web và ưng dụng thư. Bằng cách kết hợp các ứng dụng như vậy vào hđh, cta tạo gánh nặng cho hđh với chức năng bổ sung. Gánh nặng như vậy có thể dẫn đến việc HĐH thực hiện công việc quản lí tài nguyên hệ thống không đạt yêu cầu. Ngoài ra, cta tăng kích thước của hđh do đó tăng khả năng sự cố hệ thống và vi phạm an ninh.

* 1. **Làm thế nào để phân biệt giữa chế độ kernel và chức năng chế độ người dùng như một hình thức cơ bản của hệ thống bảo vệ (an ninh)?**

Trả lời: Sự khác biệt giữa chế độ kernel và chế độ người dùng cung cấp một hình thức bảo vệ cơ bản theo cách sau. Các câu lệnh nhất định chỉ có thể được thực hiện khi CPU ở chế độ kernel. Tương tự, các thiết bị phần cứng chỉ có thể được truy cập khi chương trình đang chạy chế độ kernel. Việc kiểm soát khi các ngắt có thể bật hoặc tắt cũng chỉ có thể thực hiện khi CPU ở chế độ kernel. Do đó, CPU có khả năng rất giới hạn khi chạy chế độ người dụng, do đó phải thực hiện bảo vệ các tài nguyên quan trọng.

* 1. **Hướng dẫn nào sau đây nên được ưu tiên ?**

**a. Đặt giá trị của bộ đếm thời gian.**

**b. Đọc đồng hồ.**

**c. Dọn dẹp bộ nhớ.**

**d. Đưa ra một hướng dẫn bẫy.**

**đ. Tắt ngắt.**

**f. Sửa đổi các mục trong bảng trạng thái thiết bị.**

**g. Chuyển từ chế độ người dùng sang chế độ kernel.**

**h. Truy cập thiết bị I/O**

Trả lời: a > c > e > f > h > Phần còn lại có thể thực hiện trong user mode.

* 1. **Một số máy tính đời đầu đã bảo vệ hdh bằng cách đặt nó trong 1 phân vùng bộ nhớ không thể sửa đổi bởi cả công việc của người dùng và chính HĐH. Mô tả 2 khó khăn mà bạn nghĩ có thể phát sinh với 1 kế hoạch như vậy.**

Trả lời: Dữ liệu mà HĐH yêu cầu( mật khẩu, kiểm soát truy cập, thông tin tính toán, v.v) sẽ phải đươc lưu trữ trong hoặc đi qua bộ nhớ k đc bảo vệ và do đó có thể bị truy cập bởi ng dùng trái phép.

* 1. **Một số CPU cung cấp hơn 2 chế độ điều hành. 2 cách dùng khả thi của đa chế độ này là gì?**

Trả lời:

Mặc dù hầu hết các hệ thống chỉ phân biệt giữa người dùng và chế độ nhân, một số CPU đã hỗ trợ nhiều chế độ. Nhiều chế độ có thể được sử dụng để cung cấp một chính sách bảo mật chi tiết hơn. Vì ví dụ, thay vì phân biệt giữa chế độ chỉ người dùng và kernel, bạn có thể phân biệt giữa các loại chế độ người dùng khác nhau. Có lẽ người dùng thuộc cùng một nhóm có thể thực thi mã của nhau. Các máy sẽ chuyển sang chế độ được chỉ định khi một trong những người dùng này đang chạy mã. Khi máy ở chế độ này, một thành viên của nhóm có thể chạy mã thuộc về bất kỳ ai khác trong nhóm. Một khả năng khác là cung cấp sự phân biệt khác nhau trong mã hạt nhân. Ví dụ: một chế độ cụ thể có thể cho phép trình điều khiển thiết bị USB chạy. Điều này có nghĩa là các thiết bị USB có thể được bảo dưỡng mà không cần phải chuyển sang chế độ kernel, do đó về cơ bản cho phép thiết bị USB trình điều khiển để chạy ở chế độ gần như người dùng/hạt nhân

* 1. **Bộ hẹn giờ có thể đc sử dụng để tính toán thời gian hiện tại. Cung cấp 1 mô tả ngắn về cách điều này có thể đc thực hiện.**

Trả lời: 1 chương trình có thể sử dụng cách tiếp cận sau để tính toán thời gian hiện tại bằng cách sử dụng các ngắt hẹn giờ. Chương trình có thể đặt hẹn giờ trong một thời gian trong tương lai và đi ngủ. Khi nó được đánh thức bởi ngắt, nó có thể cập nhật trạng thái cục bộ mà nó đang sử dụng để theo dõi số lần ngắt mà nó đã nhận được cho đến nay. Sau đó nó có thể lặp lại quá trình này liên tục thiết lập các ngắt hẹn giờ và cập nhật trạng thái cục bộ của nó khi các ngắt thực sự được tăng lên.

* 1. **Internet là mạng LAN hay WAN?**

Trả lời: Internet là một mạng WAN vì các máy tính khác nhau được đặt ở những nơi địa lý khác nhau và được kết nối bởi mạng lưới đường dài.

CHUONG 2

2.1. Các cuộc gọi hệ thống cho phép các quy trình cấp người dùng yêu cầu các dịch vụ của hệ điều hành

2.2.

a. Việc tạo và xóa các quy trình của cả người dùng và hệ thống

b. Việc đình chỉ và nối lại các quy trình

c. Việc cung cấp các cơ chế để đồng bộ hóa quy trình

d. Việc cung cấp các cơ chế cho quá trình giao tiếp

e. Việc cung cấp các cơ chế xử lý bế tắc

2.3.

a. Theo dõi những phần bộ nhớ hiện đang được sử dụng và bởi ai.

b. Quyết định quy trình nào sẽ được tải vào bộ nhớ khi có dung lượng bộ nhớ.

c. Phân bổ và giải phóng không gian bộ nhớ khi cần thiết

2.4.

• Quản lý không gian trống. • Phân bổ lưu trữ. • Lập lịch đĩa

2.5.

Nó đọc các lệnh từ người dùng hoặc từ một tệp lệnh và thực thi chúng, thường bằng cách biến chúng thành một hoặc nhiều hệ thống các cuộc gọi. Nó thường không phải là một phần của kernel vì trình thông dịch lệnh có thể thay đổi

2.6.

Trong các hệ thống Unix, một cuộc gọi hệ thống ngã ba theo sau bởi một hệ thống exec cuộc gọi cần phải được thực hiện để bắt đầu một quá trình mới. Cuộc gọi ngã ba nhân bản quy trình hiện đang thực thi, trong khi lệnh gọi exec phủ lên một quy trình mới dựa trên một tệp thực thi khác trong quá trình gọi.

2.7.

Các chương trình hệ thống có thể được coi là các gói hữu ích cuộc gọi hệ thống. Chúng cung cấp chức năng cơ bản cho người dùng để người dùng làm không cần viết chương trình riêng để giải các bài toán thông thường

2.8.

Như trong tất cả các trường hợp thiết kế mô-đun, thiết kế một hệ điều hành hệ thống theo cách mô-đun có một số lợi thế. Hệ thống dễ dàng hơn để gỡ lỗi và sửa đổi vì các thay đổi chỉ ảnh hưởng đến các phần hạn chế của hệ thống hơn là chạm vào tất cả các phần của hệ điều hành. Thông tin chỉ được lưu giữ ở những nơi cần thiết và chỉ có thể truy cập được trong phạm vi một khu vực được xác định và hạn chế, do đó, bất kỳ lỗi nào ảnh hưởng đến dữ liệu đó phải được giới hạn trong một mô-đun hoặc lớp cụ thể.

2.9.

a. Thực hiện chương trình. Hệ điều hành tải nội dung (hoặc các phần) của tệp vào bộ nhớ và bắt đầu thực thi. Không thể tin cậy một chương trình cấp độ người dùng để phân bổ hợp lý thời gian CPU.

b. thao tác vào/ra. Đĩa, băng, đường nối tiếp và các thiết bị khác phải được giao tiếp ở mức rất thấp. Người dùng chỉ cần chỉ định thiết bị và thao tác để thực hiện trên thiết bị đó, trong khi hệ thống chuyển đổi yêu cầu đó thành thiết bị hoặc bộ điều khiển cụ thể mệnh lệnh. Các chương trình cấp người dùng không thể được tin cậy để chỉ truy cập các thiết bị mà họ nên có quyền truy cập và chỉ truy cập chúng khi mặt khác chúng không được sử dụng.

c. Thao tác với hệ thống tập tin. Có nhiều chi tiết trong việc tạo tập tin, xóa, phân bổ và đặt tên mà người dùng không cần phải thực hiện. Các khối không gian đĩa được sử dụng bởi các tệp và phải được theo dõi. Xóa một tệp yêu cầu xóa thông tin tên tệp và giải phóng các khối được phân bổ. Các biện pháp bảo vệ cũng phải được kiểm tra để đảm bảo truy cập tập tin thích hợp. Các chương trình người dùng không thể đảm bảo tuân thủ các phương pháp bảo vệ cũng như không được tin cậy để chỉ phân bổ miễn phí các khối và giải phóng các khối khi xóa tệp.

d. Truyền thông. Tin nhắn truyền giữa các hệ thống yêu cầu tin nhắn được chuyển thành các gói thông tin, được gửi đến bộ điều khiển mạng, được truyền qua phương tiện truyền thông, và được lắp ráp lại bởi hệ thống đích. Thứ tự gói và sửa dữ liệu phải diễn ra. Một lần nữa, chương trình người dùng có thể không phối hợp truy cập vào thiết bị mạng hoặc họ có thể nhận được các gói dành cho các quá trình khác.

e. Phát hiện lỗi. Phát hiện lỗi xảy ra ở cả phần cứng và các cấp độ phần mềm. Ở cấp độ phần cứng, tất cả các lần truyền dữ liệu phải được được kiểm tra để đảm bảo rằng dữ liệu không bị hỏng trong quá trình vận chuyển. Tất cả dữ liệu trên phương tiện phải được kiểm tra để chắc chắn rằng chúng không đã thay đổi kể từ khi chúng được viết cho giới truyền thông. Tại phần mềm cấp độ, phương tiện phải được kiểm tra tính nhất quán của dữ liệu; ví dụ, liệu số lượng khối lưu trữ được phân bổ và chưa phân bổ khớp với tổng số trên thiết bị. Ở đó, sai sót thường xuyên xảy ra không phụ thuộc vào quy trình (ví dụ: lỗi dữ liệu trên một đĩa), vì vậy phải có một chương trình toàn cầu (hệ điều hành) xử lý tất cả các loại lỗi. Ngoài ra, bằng cách xử lý lỗi bởi hệ điều hành, các quy trình không cần chứa mã để bắt và sửa tất cả các lỗi có thể xảy ra trên một hệ thống.

2.10.

Các cuộc gọi hệ thống cho phép các quy trình cấp người dùng yêu cầu các dịch vụ của hệ điều hành

2.11.

Lợi ích thường bao gồm những điều sau đây (a) thêm một cái mới dịch vụ không yêu cầu sửa đổi kernel, (b) nó an toàn hơn vì nhiều thao tác được thực hiện ở chế độ người dùng hơn ở chế độ nhân và (c) một thiết kế hạt nhân đơn giản hơn và chức năng thường dẫn đến nhiều hơn hệ điều hành đáng tin cậy

2.12.

Đối với một số thiết bị, chẳng hạn như PDA cầm tay và điện thoại di động, đĩa có hệ thống tệp có thể không có sẵn cho thiết bị. Trong tình huống này, hệ điều hành phải được lưu trữ trong phần sụn

2.13.

Hãy xem xét một hệ thống muốn chạy cả Windows XP và ba bản phân phối khác nhau của Linux (ví dụ: RedHat, Debian và nhân sâm). Mỗi hệ điều hành sẽ được lưu trữ trên đĩa. Trong thời gian hệ thống khởi động, một chương trình đặc biệt (mà chúng ta sẽ gọi là trình quản lý khởi động) sẽ xác định hệ điều hành nào sẽ khởi động vào. Điều này có nghĩa là ban đầu khởi động hệ điều hành, trình quản lý khởi động sẽ chạy trước trong quá trình khởi động hệ thống. Trình quản lý khởi động này chịu trách nhiệm cho xác định hệ thống nào sẽ khởi động vào. Thông thường trình quản lý khởi động phải được lưu trữ tại các vị trí nhất định của đĩa cứng để được nhận dạng trong quá trình khởi động hệ thống. Trình quản lý khởi động thường cung cấp cho người dùng một lựa chọn hệ thống để khởi động vào; trình quản lý khởi động cũng thường được thiết kế để khởi động thành hệ điều hành mặc định nếu người dùng không lựa chọn.

CHƯƠNG 3:

* 1. **Palm OS không cung cấp phương tiện xử lý đồng thời. thảo luận ba các biến chứng chính mà xử lý đồng thời thêm vào một hoạt động hệ thống**.

Trả lời:

a. Một phương pháp chia sẻ thời gian phải được thực hiện để cho phép mỗi của một số quá trình để có quyền truy cập vào hệ thống. Phương pháp này liên quan đến việc ưu tiên các quy trình không tự nguyện cung cấp lên CPU (ví dụ bằng cách sử dụng lệnh gọi hệ thống) và nhân đang được đăng nhập lại (vì vậy có thể có nhiều hơn một tiến trình đang thực thi kernel mã đồng thời).

b. Các quy trình và tài nguyên hệ thống phải được bảo vệ và phải được bảo vệ khỏi nhau. Bất kỳ quá trình nhất định phải được giới hạn trong dung lượng bộ nhớ mà nó có thể sử dụng và các hoạt động mà nó có thể thực hiện trên các thiết bị như đĩa.

c. Cần phải cẩn thận trong kernel để ngăn ngừa bế tắc giữa các quy trình, vì vậy các quy trình không chờ phân bổ lẫn nhau tài nguyên.

* 1. **Bộ xử lý Sun UltraSPARC có nhiều bộ thanh ghi. Mô tả hành động của chuyển đổi ngữ cảnh nếu ngữ cảnh mới đã được tải vào một trong các bộ thanh ghi. Điều gì khác phải xảy ra nếu bối cảnh mới ở bộ nhớ chứ không phải trong một bộ thanh ghi và tất cả các bộ thanh ghi đang được sử dụng?**

Trả lời: Con trỏ thiết lập thanh ghi hiện tại của CPU được thay đổi để trỏ đến set chứa bối cảnh mới, mất rất ít thời gian. Nếu bối cảnh trong bộ nhớ, một trong các ngữ cảnh trong bộ thanh ghi phải được chọn và được được chuyển vào bộ nhớ và bối cảnh mới phải được tải từ bộ nhớ. Quy trình vào bộ. Quá trình này mất nhiều thời gian hơn một chút so với trên các hệ thống có một bộ thanh ghi, tùy thuộc vào cách nạn nhân thay thế được chọn. Quá trình này mất nhiều thời gian hơn một chút so với trên các hệ thống có một bộ thanh ghi, tùy thuộc vào cách nạn nhân thay thế được chọn.

* 1. **Khi một tiến trình tạo một tiến trình mới sử dụng thao tác fork(), trạng thái sau đây được chia sẻ giữa tiến trình cha và tiến trình con quá trình? Một. Cây rơm b. đống c. Phân đoạn bộ nhớ dùng chung**

Trả lời: Chỉ các phân đoạn bộ nhớ dùng chung được chia sẻ giữa tiến trình cha và tiến trình con mới rẽ nhánh. Bản sao của ngăn xếp và đống được tạo cho quy trình mới được tạo.

* 1. Một lần nữa xem xét cơ chế RPC, hãy xem xét ngữ nghĩa “chính xác một lần”. Thuật toán triển khai ngữ nghĩa này có thực thi chính xác ngay cả khi thông báo “ACK” trở lại máy khách bị mất do mạng không vấn đề? Mô tả trình tự các tin nhắn và liệu "chính xác một lần" vẫn được bảo tồn.

Trả lời: Ngữ nghĩa “chính xác một lần” đảm bảo rằng một thủ tục khác sẽ được thực hiện chính xác một lần và chỉ một lần. Thuật toán chung cho đảm bảo điều này kết hợp lược đồ xác nhận (ACK) kết hợp với dấu thời gian (hoặc một số bộ đếm gia tăng khác cho phép máy chủ để phân biệt giữa các tin nhắn trùng lặp). Chiến lược chung là để máy khách gửi RPC đến máy chủ cùng với với một dấu thời gian. Máy khách cũng sẽ bắt đầu đồng hồ hết thời gian chờ. Khách hàng sau đó sẽ đợi một trong hai lần xuất hiện: (1) nó sẽ nhận được ACK từ máy chủ chỉ ra rằng quy trình từ xa đã được thực hiện hoặc (2) nó sẽ hết thời gian. Nếu máy khách hết thời gian, nó cho rằng máy chủ không thể để thực hiện quy trình từ xa để máy khách gọi RPC một giây thời gian, gửi dấu thời gian sau. Khách hàng có thể không nhận được ACK cho một trong hai lý do: (1) máy chủ chưa bao giờ nhận được RPC ban đầu, hoặc (2) RPC đã được nhận chính xác—và được thực hiện—bởi máy chủ nhưng ACK đã bị mất. Trong tình huống (1), việc sử dụng ACK cho phép máy chủ cuối cùng để nhận và thực hiện RPC. Trong trường hợp (2), máy chủ sẽ nhận được một RPC trùng lặp và nó sẽ sử dụng dấu thời gian để xác định đó là một sao chép để không thực hiện RPC lần thứ hai. Điều quan trọng là lưu ý rằng máy chủ phải gửi lại ACK thứ hai cho máy khách để thông báo khách hàng mà RPC đã được thực hiện.

* 1. Giả sử rằng một hệ thống phân tán dễ bị lỗi máy chủ. Cái gì các cơ chế sẽ được yêu cầu để đảm bảo ngữ nghĩa “chính xác một lần” để thực thi các RPC?

Trả lời: Máy chủ sẽ theo dõi trong bộ lưu trữ ổn định (chẳng hạn như disk log) thông tin liên quan đến hoạt động RPC nào đã được nhận, liệu chúng có được thực hiện thành công hay không và các kết quả liên quan với các thao tác. Khi xảy ra sự cố máy chủ và thông báo RPC được nhận, máy chủ có thể kiểm tra xem RPC đã được nhận trước đó chưa được thực hiện và do đó đảm bảo ngữ nghĩa “chính xác một lần” cho thực thi RPC( remote procedure call)

CHUONG 4:

**4.1. Cung cấp hai ví dụ lập trình trong đó đa luồng cung cấp hiệu suất tốt hơn so với một giải pháp đơn luồng.**

Trả lời: (1) Một máy chủ Web phục vụ từng yêu cầu trong một chuỗi riêng biệt. 2) (Một ứng dụng song song chẳng hạn như phép nhân ma trận trong đó (các phần khác nhau của ma trận có thể được thực hiện song song. (3) Một (chương trình GUI tương tác chẳng hạn như trình gỡ lỗi trong đó một luồng được sử dụng (để giám sát đầu vào của người dùng, một luồng khác đại diện cho hoạt động (ứng dụng và chủ đề thứ ba theo dõi hiệu suất.

4.2 **Hai sự khác biệt giữa luồng cấp người dùng và cấp nhân là gì chủ đề? Trong hoàn cảnh nào thì một loại tốt hơn loại kia?**

Trả lời: (1) Các luồng cấp độ người dùng không được biết bởi nhân, trong khi kernel nhận thức được các luồng kernel. (2) Trên hệ thống sử dụng M:1 hoặc M:N lập bản đồ, luồng người dùng được lên lịch bởi thư viện luồng và hạt nhân lịch trình chủ đề hạt nhân. (3) Chuỗi hạt nhân không cần được liên kết với một quy trình trong khi mọi chuỗi người dùng thuộc về một quy trình. Chủ đề hạt nhân thường tốn kém hơn để duy trì so với chủ đề người dùng vì họ phải được biểu diễn bằng cấu trúc dữ liệu hạt nhân.

4.3. Mô tả các hành động được thực hiện bởi một nhân để chuyển ngữ cảnh giữa các luồng cấp nhân.

Trả lời: Chuyển đổi ngữ cảnh giữa các luồng nhân thường yêu cầu lưu giá trị của các thanh ghi CPU từ luồng bị tắt và khôi phục các thanh ghi CPU của luồng mới đang được lên lịch. 4.4 Những tài nguyên nào được sử dụng khi một luồng được tạo? Chúng khác nhau như thế nào từ những cái được sử dụng khi một quá trình được tạo ra? Trả lời: Bởi vì một luồng nhỏ hơn một tiến trình, nên việc tạo luồng thường sử dụng ít tài nguyên hơn so với quá trình tạo. Tạo một quy trình yêu cầu cấp phát một khối điều khiển quá trình (PCB), một cấu trúc dữ liệu khá lớn. PCB bao gồm bản đồ bộ nhớ, danh sách các tệp đang mở và các biến môi trường. Cấp phát và quản lý bản đồ bộ nhớ thường là hoạt động tốn nhiều thời gian nhất. Tạo một chủ đề người dùng hoặc hạt nhân liên quan đến việc phân bổ một cấu trúc dữ liệu nhỏ để giữ một bộ thanh ghi, ngăn xếp và sự ưu tiên.

4.5. Giả sử một hệ điều hành ánh xạ các luồng cấp người dùng tới nhân bằng cách sử dụng mô hình nhiều-nhiều và việc ánh xạ được thực hiện thông qua LWP. Hơn nữa, hệ thống cho phép các nhà phát triển tạo các luồng thời gian thực. Có nhất thiết phải liên kết chuỗi thời gian thực với LWP không? Giải thích.

Trả lời có. Thời gian là rất quan trọng đối với các ứng dụng thời gian thực. Nếu một chủ đề là được đánh dấu là thời gian thực nhưng không bị ràng buộc với LWP, chuỗi có thể có để đợi được gắn vào LWP trước khi chạy. Xem xét nếu một thời gian thực luồng đang chạy (được gắn vào LWP) và sau đó tiến hành chặn (tức là phải thực hiện I/O, đã được ưu tiên trước bởi thời gian thực có mức độ ưu tiên cao hơn luồng, đang chờ khóa loại trừ lẫn nhau, v.v.) Trong khi thời gian thực luồng bị chặn, LWP mà nó được gắn vào đã được gán cho luồng khác chủ đề. Khi chuỗi thời gian thực đã được lên lịch để chạy lại, nó trước tiên phải đợi để được gắn vào LWP. Bằng cách ràng buộc một LWP với một luồng thời gian thực, bạn đảm bảo rằng luồng đó sẽ có thể chạy với tốc độ tối thiểu trì hoãn một khi nó được lên lịch.

4.6 Một chương trình Pthread thực hiện chức năng tính tổng đã được cung cấp trong Mục 4.3.1. Viết lại chương trình này bằng Java.

Trả lời: Vui lòng tham khảo trang web hỗ trợ để biết giải pháp mã nguồn.

CHUONG 5

5.1.

n! (n factorial = n × n – 1 × n – 2 × ... × 2 × 1).

5.2.

Lập lịch ưu tiên cho phép một quá trình bị gián đoạn ở giữa quá trình thực thi, lấy CPU đi và phân bổ nó sang một tiến trình khác. Lập lịch không ưu tiên đảm bảo rằng một quá trình chỉ từ bỏ quyền kiểm soát CPU khi nó hoàn thành công việc hiện tại nổ CPU.

5.3.

Hãy nhớ rằng thời gian quay vòng là thời gian kết thúc trừ đi thời gian đến, vì vậy bạn phải trừ thời gian đến để tính thời gian quay vòng. FCFS là 11 nếu bạn quên trừ thời gian đến.

5.4.

Ví dụ, các quy trình cần bảo dưỡng thường xuyên hơn, các quy trình tương tác như trình chỉnh sửa, có thể xếp hàng trong thời gian ngắn lượng tử. Các quy trình không cần bảo trì thường xuyên có thể nằm trong hàng đợi với lượng tử lớn hơn, yêu cầu ít chuyển ngữ cảnh hơn để hoàn thành xử lý, và do đó làm cho việc sử dụng máy tính hiệu quả hơn.

5.5.

a. Công việc ngắn nhất có mức độ ưu tiên cao nhất.

b. Mức thấp nhất của MLFQ là FCFS.

c. FCFS dành ưu tiên cao nhất cho công việc đã tồn tại dài nhất.

d. Không có

5.6.

Nó sẽ ưu tiên các chương trình liên kết I/O vì tương đối yêu cầu bùng nổ CPU ngắn của họ; tuy nhiên, các chương trình liên kết với CPU sẽ không chết đói vì các chương trình liên kết với I/O sẽ từ bỏ CPU tương đối thường xuyên để thực hiện I/O của họ.

5.7.

Lập lịch trình PCS được thực hiện cục bộ cho quy trình. Đó là cách thư viện luồng lên lịch các luồng vào các LWP có sẵn. lập lịch trình SCS là tình huống mà hệ điều hành lên lịch cho các luồng hạt nhân. TRÊN các hệ thống sử dụng nhiều-một hoặc nhiều-nhiều, lập lịch hai các mô hình về cơ bản là khác nhau. Trên các hệ thống sử dụng một đối một, PCS và SCS là như nhau.

5.8.

Có, nếu không thì chuỗi người dùng có thể phải cạnh tranh để giành được một LWP có sẵn trước khi được lên lịch thực sự. Bằng cách ràng buộc người dùng luồng tới LWP, không có độ trễ trong khi chờ LWP khả dụng; chủ đề người dùng thời gian thực có thể được lên lịch ngay lập tức

CHUONG 6

**6.1** Trong Phần 6.4, chúng tôi đã đề cập rằng việc vô hiệu hóa ngắt thường xuyên có thể

ảnh hưởng đến đồng hồ của hệ thống. Giải thích tại sao nó có thể và những ảnh hưởng như thế nào có thể được giảm thiểu.

Trả lời: Đồng hồ hệ thống được cập nhật tại mỗi lần ngắt đồng hồ. Nếu các ngắt bị vô hiệu hóa—đặc biệt là trong một thời gian dài—thì đó là có thể đồng hồ hệ thống có thể dễ dàng mất thời gian chính xác. Đồng hồ hệ thống cũng được sử dụng cho mục đích lập lịch trình. Ví dụ, thời gian lượng tử cho một quá trình được biểu thị bằng một số tích tắc đồng hồ. Ở mọi đồng hồ bị gián đoạn, bộ lập lịch xác định xem định lượng thời gian cho quá trình đang chạy đã hết hạn. Nếu ngắt đồng hồ bị vô hiệu hóa, bộ lập lịch trình không thể chỉ định chính xác các định mức thời gian. Hiệu ứng này có thể được giảm thiểu bằng cách vô hiệu hóa các ngắt đồng hồ chỉ trong khoảng thời gian rất ngắn.

**6.2** Vấn Đề Người Hút Thuốc Lá. Hãy xem xét một hệ thống có ba quy trình hút thuốc và một quy trình tác nhân. Mỗi người hút cuộn liên tục một điếu thuốc và sau đó hút nó. Nhưng để cuộn và hút một điếu thuốc, người hút cần ba thành phần: thuốc lá, giấy và diêm. Một trong những quy trình hút thuốc có giấy, một quy trình khác có thuốc lá và quy trình thứ ba có diêm. Các đại lý có nguồn cung cấp vô hạn cả ba nguyên liệu. Các đại lý nơi hai trong số các thành phần trên bàn. Người hút thuốc còn lại thành phần sau đó tạo ra và hút một điếu thuốc, báo hiệu tác nhân trên hoàn thành. Sau đó, tác nhân đưa ra hai trong số ba thành phần khác và chu kỳ lặp lại. Viết chương trình đồng bộ hóa agent và những người hút thuốc sử dụng đồng bộ hóa Java.

Trả lời: Vui lòng tham khảo trang web hỗ trợ để biết giải pháp mã nguồn.

**6.3** Đưa ra lý do tại sao Solaris, Windows XP và Linux triển khai nhiều cơ chế khóa. Mô tả hoàn cảnh mà họ họ sử dụng spinlocks, mutexes, semaphores, Adaptive mutexes, và điều kiện

biến. Trong mỗi trường hợp, hãy giải thích tại sao cơ chế này là cần thiết.

Trả lời: Các hệ điều hành này cung cấp các cơ chế khóa khác nhau tùy thuộc vào nhu cầu của nhà phát triển ứng dụng. Spinlocks là

hữu ích cho các hệ thống đa bộ xử lý nơi một luồng có thể chạy trong vòng lặp bận

(trong một khoảng thời gian ngắn) thay vì chịu chi phí

đặt trong một hàng đợi ngủ. Mutexes rất hữu ích để khóa tài nguyên. năng lượng mặt trời 2

sử dụng các bộ chuyển đổi thích ứng, có nghĩa là bộ chuyển đổi đó được triển khai với một

khóa quay trên các máy đa bộ xử lý. Semaphores và các biến điều kiện là những công cụ thích hợp hơn để đồng bộ hóa khi một tài nguyên

phải được giữ trong một thời gian dài, vì kéo sợi không hiệu quả trong một thời gian dài

thời lượng dài.

**6.4** Giải thích sự khác biệt về mặt chi phí giữa ba loại lưu trữ

dễ bay hơi, không bay hơi và ổn định.

Trả lời: Bộ lưu trữ dễ bay hơi đề cập đến bộ nhớ chính và bộ nhớ cache và rất

nhanh. Tuy nhiên, lưu trữ dễ bay hơi không thể tồn tại khi hệ thống gặp sự cố hoặc tắt nguồn hệ thống. Lưu trữ cố định vẫn tồn tại sau sự cố hệ thống và

hệ thống tắt nguồn. Đĩa và băng là những ví dụ về tính không bay hơi

kho. Gần đây, các thiết bị USB sử dụng bộ nhớ chỉ đọc chương trình có thể xóa được (EPROM) đã xuất hiện cung cấp khả năng lưu trữ cố định. Lưu trữ ổn định đề cập đến lưu trữ mà về mặt kỹ thuật không bao giờ có thể bị mất vì có các bản sao dự phòng dự phòng của dữ liệu (thường là trên đĩa).

**6.5** Giải thích mục đích của cơ chế điểm kiểm tra. Bao lâu nên điểm kiểm tra được thực hiện? Mô tả tần suất của các trạm kiểm soát ảnh hưởng đến:

• Hiệu suất hệ thống khi không xảy ra lỗi

• Thời gian cần thiết để phục hồi sau sự cố hệ thống

• Thời gian cần thiết để phục hồi sau sự cố đĩa

**Trả lời:** Một bản ghi nhật ký điểm kiểm tra cho biết rằng một bản ghi nhật ký và dữ liệu sửa đổi đã được ghi vào bộ lưu trữ ổn định và giao dịch không cần phải thực hiện lại trong trường hợp hệ thống gặp sự cố. Rõ ràng, càng thường xuyên điểm kiểm tra được thực hiện, thì khả năng cập nhật dư thừa càng ít sẽ phải được thực hiện trong quá trình phục hồi.

• Hiệu năng hệ thống khi không xảy ra lỗi— Nếu không xảy ra lỗi, hệ thống phải chịu chi phí thực hiện các điểm kiểm tra về cơ bản là không cần thiết. Trong tình huống này, việc thực hiện các điểm kiểm tra ít thường xuyên hơn sẽ dẫn đến hiệu năng hệ thống tốt hơn.

• Thời gian cần thiết để phục hồi sau sự cố hệ thống—Sự tồn tại của một bản ghi điểm kiểm tra có nghĩa là một hoạt động sẽ không phải làm lại trong quá trình phục hồi hệ thống. Trong tình huống này, càng thường xuyên điểm kiểm tra đã được thực hiện, thời gian phục hồi từ một sự cố hệ thống.

• Thời gian cần thiết để phục hồi sau sự cố đĩa—Sự tồn tại của một bản ghi điểm kiểm tra có nghĩa là một hoạt động sẽ không phải làm lại trong quá trình phục hồi hệ thống. Trong tình huống này, càng thường xuyên điểm kiểm tra đã được thực hiện, thời gian phục hồi từ một sự cố đĩa.

**6.6**

Một giao dịch là một loạt các thao tác đọc và ghi trên một số dữ liệu theo sau là một hoạt động cam kết. Nếu chuỗi hoạt động trong một giao dịch không thể hoàn thành, giao dịch đó phải bị hủy bỏ và các hoạt động đã diễn ra phải được khôi phục. Nó quan trọng chuỗi hoạt động trong một giao dịch xuất hiện như một khối không thể chia cắt hoạt động để đảm bảo tính toàn vẹn của dữ liệu được cập nhật. Nếu không thì, dữ liệu có thể bị xâm phạm nếu hoạt động từ hai (hoặc nhiều) khác nhau các giao dịch đã được trộn lẫn.

**6.7**

Một lịch trình được cho phép trong giao thức khóa hai pha

nhưng không phải trong giao thức dấu thời gian là:

Lịch trình này không được phép trong giao thức dấu thời gian vì ở bước

7, dấu thời gian W của B là 1.

Một lịch trình được cho phép trong giao thức dấu thời gian nhưng không có trong

giao thức khóa hai pha là:

Lịch trình này không được thêm hướng dẫn khóa để làm cho nó hợp pháp theo

giao thức khóa hai pha vì T1 phải mở khóa (A) giữa các bước 2

và 3, và phải khóa (B) giữa các bước 4 và 5.

**6.8**

Đây là vấn đề quan trọng cần nhấn mạnh! Java chỉ cung cấp thông báo ẩn danh—bạn không thể thông báo cho một luồng nhất định rằng một điều kiện nhất định là đúng. Khi một chủ đề được thông báo, đó là trách nhiệm của nó để kiểm tra lại điều kiện mà nó đang chờ đợi. Nếu một luồng không kiểm tra lại điều kiện, nó có thể đã nhận được thông báo mà không cần điều kiện đã được đáp ứng.

CHUONG 7

**7.1**

Trả lời:

• Hai ô tô đi ngược chiều nhau qua cầu một làn.

• Một người xuống thang trong khi một người khác đang leo lên

thang.

• Hai đoàn tàu chạy ngược chiều nhau trên cùng một đường ray.

• Hai người thợ mộc phải đóng đinh. Có một cái búa duy nhất

và một thùng đinh. Bế tắc xảy ra nếu một người thợ mộc có

cái búa và người thợ mộc khác có những cái đinh

**7.3**

Hình 7.1 cung cấp mã Java triển khai thuật toán an toàn của

thuật toán của chủ ngân hàng (việc triển khai đầy đủ thuật toán của chủ ngân hàng

thuật toán có sẵn khi tải xuống mã nguồn).

Có thể thấy, các vòng lặp bên ngoài lồng nhau—cả hai đều lặp qua n

lần—cung cấp hiệu suất n2. Trong các vòng lặp bên ngoài này là hai

tuần tự các vòng lặp bên trong lặp lại m lần. Big-oh của thuật toán này

do đó là O(m × n2).

**7.4**

Một đối số để cài đặt tránh bế tắc trong hệ thống là chúng ta có thể đảm bảo bế tắc sẽ không bao giờ xảy ra. Ngoài ra, mặc dù tăng thời gian quay vòng, tất cả 5.000 công việc vẫn có thể chạy.Một lập luận chống lại việc cài đặt phần mềm tránh bế tắc là bế tắc xảy ra không thường xuyên và chúng tốn ít chi phí khi chúng xảy ra.

**7.5**

Chết đói là một chủ đề khó xác định vì nó có thể có nghĩa khác thứ cho các hệ thống khác nhau. Đối với mục đích của câu hỏi này, chúng tôi sẽ định nghĩa chết đói là tình huống theo đó một quá trình phải đợi quá một khoảng thời gian hợp lý—có thể là vô thời hạn—trước khi nhận được tài nguyên được yêu cầu. Một cách để phát hiện nạn đói trước tiên là xác định một khoảng thời gian—T—được coi là không hợp lý. Khi một quá trình yêu cầu một tài nguyên, một bộ đếm thời gian được bắt đầu. Nếu thời gian trôi qua vượt quá T, sau đó quá trình được coi là bị bỏ đói.Một chiến lược để đối phó với nạn đói là áp dụng chính sách trong đó tài nguyên chỉ được gán cho quá trình đang chờ dài nhất. Ví dụ: nếu quy trình Pa đợi tài nguyên X lâu hơn quy trình Pb, thì yêu cầu từ quy trình Pb sẽ bị hoãn lại cho đến khi yêu cầu của tiến trình Pa được thỏa mãn.

Một chiến lược khác sẽ ít nghiêm ngặt hơn những gì vừa được đề cập. TRONG trong trường hợp này, một tài nguyên có thể được cấp cho một quá trình đã chờ đợi ít hơn tiến trình khác, với điều kiện là tiến trình kia không bị chết đói. Tuy nhiên, nếu một quá trình khác được coi là đang chết đói, yêu cầu của nó sẽ được hài lòng đầu tiên.

**CHƯƠNG 8**

**8.1**

Một địa chỉ logic không đề cập đến một địa chỉ thực tế hiện có; đúng hơn, nó đề cập đến một địa chỉ trừu tượng trong một không gian địa chỉ trừu tượng. Tương phản điều này với một địa chỉ vật lý đề cập đến một địa chỉ vật lý thực tế trong trí nhớ. Một địa chỉ logic được tạo bởi CPU và được dịch thành một địa chỉ vật lý bởi đơn vị quản lý bộ nhớ (MMU). Do đó, các địa chỉ vật lý được tạo bởi MMU

**8.2**

Ưu điểm chính của chương trình này là nó có hiệu quả cơ chế chia sẻ mã và dữ liệu. Ví dụ, chỉ một bản sao của một trình soạn thảo hoặc trình biên dịch cần được lưu trong bộ nhớ và mã này có thể được được chia sẻ bởi tất cả các quy trình cần truy cập vào trình soạn thảo hoặc mã trình biên dịch.

Một ưu điểm khác là bảo vệ mã chống sửa đổi sai. Nhược điểm duy nhất là mã và dữ liệu phải được tách ra, thường được tuân thủ trong mã do trình biên dịch tạo.

**8.3**

Nhớ lại rằng phân trang được thực hiện bằng cách chia địa chỉ thành một trang và số bù. Cách hiệu quả nhất là chia địa chỉ thành các bit trang X và bit bù Y, thay vì thực hiện phép tính số học trên địa chỉ để tính số trang và độ lệch. Bởi vì mỗi vị trí bit đại diện cho lũy thừa 2, nên việc chia địa chỉ giữa các bit dẫn đến kích thước trang là lũy thừa 2.

**8.5**

Bằng cách cho phép hai mục trong bảng trang trỏ đến cùng một khung trang trong bộ nhớ, người dùng có thể chia sẻ mã và dữ liệu. Nếu mã được nhập lại, nhiều không gian bộ nhớ có thể được tiết kiệm thông qua việc sử dụng chung các chương trình lớn như trình soạn thảo văn bản, trình biên dịch và hệ thống cơ sở dữ liệu. Việc “sao chép” một lượng lớn bộ nhớ có thể được thực hiện bằng cách để các bảng trang khác nhau trỏ đến cùng một vị trí bộ nhớ. Tuy nhiên, việc chia sẻ mã hoặc dữ liệu không liên quan có nghĩa là bất kỳ người dùng nào có quyền truy cập vào mã có thể sửa đổi nó và những sửa đổi này sẽ được phản ánh trong “bản sao” của người dùng khác.

**8.6**

Vì các bảng phân đoạn là một tập hợp các thanh ghi giới hạn cơ sở, các phân đoạn có thể được chia sẻ khi các mục trong bảng phân đoạn của hai công việc khác nhau trỏ đến cùng một vị trí thực tế. Hai bảng phân đoạn phải có các con trỏ cơ sở giống hệt nhau và số phân đoạn được chia sẻ phải giống nhau trong hai quy trình.

**8.7**

Cả hai vấn đề này dẫn đến việc chương trình có thể tham chiếu cả mã và dữ liệu của chính nó mà không cần biết phân đoạn hoặc số trang được liên kết với địa chỉ. MULTICS đã giải quyết vấn đề này bằng cách liên kết bốn thanh ghi với mỗi quy trình. Một thanh ghi có địa chỉ của đoạn chương trình hiện tại, một thanh ghi khác có địa chỉ cơ sở cho ngăn xếp, một thanh ghi khác có địa chỉ cơ sở cho dữ liệu chung, v.v. Ý tưởng là tất cả các tham chiếu phải gián tiếp thông qua một thanh ghi ánh xạ tới phân đoạn hoặc số trang hiện tại. Bằng cách thay đổi các thanh ghi này, cùng một mã có thể thực thi cho các quy trình khác nhau mà không có cùng số trang hoặc số phân đoạn.

**8.8**

a. Bảo vệ không cần thiết, đặt khóa hệ thống thành 0.

b. Đặt phím hệ thống thành 0 khi ở chế độ giám sát.

c. Kích thước vùng phải được cố định theo gia số 2k byte, cấp phát khóa

với các khối bộ nhớ.

đ. Giống như trên.

đ. Kích thước khung hình phải tăng dần 2k byte, cấp phát khóa với

trang.

f. Kích thước phân đoạn phải tăng dần 2k byte, phân bổ khóa với

phân khúc.

**CHƯƠNG 9**

**9.1**

Lỗi trang xảy ra khi truy cập vào một trang chưa được đưa vào bộ nhớ chính. Hệ điều hành xác minh quyền truy cập bộ nhớ, hủy bỏ chương trình nếu nó không hợp lệ. Nếu nó hợp lệ, một khung trống được định vị và I/O được yêu cầu đọc trang cần thiết vào khung trống. Sau khi hoàn thành I/O, bảng quy trình và bảng trang được cập nhật và lệnh được khởi động lại.

**9.9**

Bạn có thể sử dụng bit hợp lệ/không hợp lệ được hỗ trợ trong phần cứng để mô phỏng bit tham chiếu. Ban đầu đặt bit thành không hợp lệ. Trên tài liệu tham khảo đầu tiên một cái bẫy cho hệ điều hành được tạo ra. Hệ điều hành sẽ đặt bit phần mềm thành 1 và đặt lại bit hợp lệ/không hợp lệ thành hợp lệ.

**9.10**

Không. Một thuật toán tối ưu sẽ không bị dị thường Belady bởi vì—theo định nghĩa—một thuật toán tối ưu sẽ thay thế trang không được sử dụng trong thời gian dài nhất. Sự bất thường của Belady xảy ra khi thuật toán thay thế trang loại bỏ một trang sẽ cần thiết trong tương lai gần. Một thuật toán tối ưu sẽ không chọn một trang như vậy.

**9.11**

Một. FIFO. Tìm phân khúc đầu tiên đủ lớn để chứa phân khúc sắp tới. Nếu không thể di dời và không có một đoạn nàođủ lớn, hãy chọn tổ hợp các phân đoạn có ký ức liền kề nhau, “gần nhất với đầu tiên của danh sách”và có thể đáp ứng phân khúc mới. Nếu di dời làcó thể, hãy sắp xếp lại bộ nhớ sao cho N phân đoạn đầu tiên lớnđủ cho phân đoạn đến nằm liền kề trong bộ nhớ. Thêm vàobất kỳ không gian còn lại nào vào danh sách không gian trống trong cả hai trường hợp.

b. LRU. Chọn đoạn không sử dụng lâu nhấtkhoảng thời gian và đủ lớn, thêm bất kỳ không gian còn lạivào danh sách dung lượng trống. Nếu không có phân đoạn nào đủ lớn, hãy chọnmột sự kết hợp của các phân khúc "cổ nhất" liền kề nhau trongbộ nhớ (nếu không có khả năng di chuyển) và đủ lớn.Nếu có thể di chuyển, hãy sắp xếp lại N phân đoạn cũ nhất thànhliền kề trong bộ nhớ và thay thế chúng bằng phân đoạn mới.

**9.13**

Bảng trang có thể được thiết lập để mô phỏng các thanh ghi cơ sở và giới hạn miễn là bộ nhớ được phân bổ trong các phân đoạn có kích thước cố định. Theo cách này, cơ sở của một phân đoạn có thể được nhập vào bảng trang và bit hợp lệ/không hợp lệ được sử dụng để chỉ ra phần đó của phân đoạn nằm trong bộ nhớ. Sẽ có một số vấn đề với sự phân mảnh nội bộ.

**CHƯƠNG** **10**

**10.1**

Việc xóa tất cả các tệp không được người dùng lưu cụ thể có ưu điểm là giảm thiểu dung lượng tệp cần thiết cho mỗi người dùng bằng cách không lưu các tệp không mong muốn hoặc không cần thiết. Lưu tất cả các tệp trừ khi bị xóa cụ thể sẽ an toàn hơn cho người dùng ở chỗ không thể vô tình làm mất tệp do quên lưu chúng

**10.2**

Một số hệ thống cho phép các thao tác tệp khác nhau dựa trên loại tệp (ví dụ: tệp ascii có thể được đọc dưới dạng luồng trong khi tệp tệp cơ sở dữ liệu có thể được đọc qua một chỉ mục cho một khối). Các hệ thống khác rời đi diễn giải như vậy về dữ liệu của tệp đối với quy trình và không cung cấp trợ giúp nào trong việc truy cập dữ liệu. Phương pháp “tốt hơn” phụ thuộc vào nhu cầu của các quy trình trên hệ thống và nhu cầu của người dùng đối với hệ điều hành. Nếu một hệ thống chủ yếu chạy các ứng dụng cơ sở dữ liệu, thì hệ điều hành có thể triển khai một tệp kiểu cơ sở dữ liệu và cung cấp các thao tác sẽ hiệu quả hơn thay vì làm cho mỗi chương trình triển khai cùng một thứ (có thể theo những cách khác nhau). Đối với các hệ thống có mục đích chung, có thể tốt hơn là chỉ triển khai các loại tệp cơ bản để giữ cho kích thước hệ điều hành nhỏ hơn và cho phép tự do tối đa đối với các quy trình trên hệ thống**.**

**10.3**

Một lợi thế của việc hệ thống hỗ trợ các cấu trúc tệp khác nhau là sự hỗ trợ đến từ hệ thống; các ứng dụng riêng lẻ không bắt buộc phải cung cấp hỗ trợ. Ngoài ra, nếu hệ thống cung cấp hỗ trợ cho các cấu trúc tệp khác nhau, thì nó có thể triển khai hỗ trợ có lẽ hiệu quả hơn một ứng dụng. Nhược điểm của việc hệ thống cung cấp hỗ trợ cho các loại tệp được xác định là nó làm tăng kích thước của hệ thống. Ngoài ra, các ứng dụng có thể yêu cầu các loại tệp khác với những gì được cung cấp bởi hệ thống có thể không chạy được trên các hệ thống như vậy.

Một chiến lược thay thế là để hệ điều hành xác định không hỗ trợ cấu trúc tệp và thay vào đó coi tất cả các tệp là một chuỗi byte. Đây là cách tiếp cận được thực hiện bởi các hệ thống UNIX. Ưu điểm của phương pháp này là nó đơn giản hóa việc hỗ trợ hệ điều hành cho các hệ thống tệp, vì hệ thống không còn phải cung cấp cấu trúc cho các loại tệp khác nhau.

Hơn nữa, nó cho phép các ứng dụng xác định cấu trúc tệp, do đó giảm bớt tình trạng hệ thống có thể không cung cấp định nghĩa tệp cần thiết cho một ứng dụng cụ thể.

**10.4**

Nếu có thể sử dụng tên dài tùy ý thì có thể mô phỏng cấu trúc thư mục đa cấp. Điều này có thể được thực hiện, ví dụ, bằng cách sử dụng ký tự “.” để chỉ ra sự kết thúc của một thư mục con. Do đó, ví dụ, tên jim.java.F1 xác định rằng F1 là một tệp trong thư mục con java, tệp này lại nằm trong thư mục gốc jim.

Nếu tên tệp bị giới hạn trong bảy ký tự, thì lược đồ trên không thể được sử dụng và do đó, nói chung, câu trả lời là không. tốt nhất tiếp theo cách tiếp cận trong tình huống này sẽ là sử dụng một tệp cụ thể làm bảng biểu tượng (thư mục) để ánh xạ các tên dài tùy ý (chẳng hạn như jim.java.F1) thành các tên tùy ý ngắn hơn (chẳng hạn như XX00743), sau đó được sử dụng để truy cập tệp thực tế .

CHƯƠNG 12

**12.2**

Trong môi trường một người dùng, hàng đợi I/O thường trống.

Các yêu cầu thường đến từ một quy trình duy nhất cho một khối hoặc cho một

dãy các khối liên tiếp. Trong những trường hợp này, FCFS là một giải pháp kinh tế

phương pháp lập lịch đĩa. Nhưng LOOK gần như dễ lập trình

và sẽ cho hiệu suất tốt hơn nhiều khi nhiều quy trình được

thực hiện I/O đồng thời, chẳng hạn như khi trình duyệt Web truy xuất dữ liệu

ở chế độ nền trong khi hệ điều hành đang phân trang và một số khác

ứng dụng đang hoạt động ở phía trước.

**12.3**

Tâm đĩa là vị trí có khoảng cách trung bình nhỏ nhất đến tất cả các rãnh còn lại. Do đó, đầu đĩa có xu hướng di chuyển ra khỏi các cạnh của đĩa. Đây là một cách khác để nghĩ về nó. Vị trí hiện tại của đầu chia các xi lanh thành hai nhóm. Nếu phần đầu không nằm ở giữa đĩa và một yêu cầu mới đến, thì yêu cầu mới có nhiều khả năng nằm trong nhóm bao gồm phần giữa của đĩa; do đó, đầu có nhiều khả năng di chuyển theo hướng đó.

**12.4**

Hầu hết các đĩa không xuất thông tin vị trí quay của chúng tới máy chủ. Ngay cả khi họ đã làm, thời gian để thông tin này đạt được bộ lập lịch sẽ không chính xác và thời gian mà bộ lập lịch sử dụng có thể thay đổi, vì vậy thông tin vị trí quay sẽ trở nên không chính xác. Hơn nữa, các yêu cầu đĩa thường được đưa ra dưới dạng số khối logic và ánh xạ giữa các khối logic và vị trí vật lý là rất phức tạp.

**12.5**

Lập kế hoạch đĩa cố gắng giảm thời gian định vị đầu đĩa. Do đĩa RAM có thời gian truy cập thống nhất nên việc lập lịch trình phần lớn là không cần thiết. Việc so sánh giữa đĩa RAM và bộ nhớ chính đĩa-cache không có ý nghĩa gì đối với việc lập lịch cho đĩa cứng bởi vì chúng tôi chỉ lập lịch cho bộ nhớ cache bị lỗi chứ không phải yêu cầu tìm dữ liệu của họ trong bộ nhớ chính

**12.6**

Một hệ thống chỉ có thể hoạt động ở tốc độ của nút cổ chai chậm nhất. Đĩa hoặc bộ điều khiển đĩa thường là nút cổ chai trong các hệ thống hiện đại vì hiệu suất riêng lẻ của chúng không thể theo kịp hiệu suất của CPU và bus hệ thống. Bằng cách cân bằng I/O giữa các đĩa và bộ điều khiển, cả đĩa riêng lẻ và bộ điều khiển đều không bị quá tải, do đó tránh được hiện tượng thắt cổ chai

**12.7**

Nếu các trang mã được lưu trữ trong không gian hoán đổi, chúng có thể được chuyển nhanh hơn vào bộ nhớ chính (vì phân bổ không gian trao đổi được điều chỉnh để có hiệu suất nhanh hơn phân bổ hệ thống tệp chung). Sử dụng trao đổikhông gian có thể yêu cầu thời gian khởi động nếu các trang được sao chép ở đó trong quá trình lời gọi thay vì chỉ được phân trang để trao đổi không gian theo yêu cầu. Ngoài ra, phải phân bổ nhiều không gian hoán đổi hơn nếu nó được sử dụng cho cả mã và trang dữ liệu

**12.8**

Lưu trữ thực sự ổn định sẽ không bao giờ mất dữ liệu. Kỹ thuật cơ bản để lưu trữ ổn định là duy trì nhiều bản sao của dữ liệu để nếu một bản sao bị hủy, một số bản sao khác vẫn có sẵn để sử dụng. Nhưng đối với bất kỳ kế hoạch nào, chúng ta có thể tưởng tượng ra một thảm họa đủ lớn khiến tất cả các bản sao bị phá hủy

**CHƯƠNG 13**

**13.1**

Ba ưu điểm: Lỗi ít có khả năng gây ra lỗi hoạt động

sự cố hệ thống Hiệu suất có thể được cải thiện bằng cách sử dụng phần cứng chuyên dụng và thuật toán mã hóa cứng Nhân được đơn giản hóa bằng cách di chuyển các thuật toán ra khỏi nó Ba nhược điểm: Lỗi khó sửa hơn—một phiên bản phần sụn mới hoặc phần cứng mới là cần thiết Việc cải thiện các thuật toán cũng yêu cầu cập nhật phần cứng thay vì

chỉ là bản cập nhật kernel hoặc trình điều khiển thiết bị Các thuật toán nhúng có thể xung đột với việc sử dụng thiết bị của ứng dụng, làm giảm hiệu suất

**13.2**

Có thể, sử dụng thuật toán sau. Hãy giả sử chúng tôi chỉ cần sử dụng bit bận (hoặc bit sẵn sàng cho lệnh; câu trả lời này là như nhau bất kể). Khi bit tắt, bộ điều khiển không hoạt động. Các máy chủ ghi dữ liệu ra và đặt bit để báo hiệu rằng một hoạt động đang sẵn sàng (tương đương với việc thiết lập bit sẵn sàng cho lệnh). Khi mà bộ điều khiển kết thúc, nó sẽ xóa bit bận. Máy chủ sau đó bắt đầu quá trình hoạt động tiếp theo. Giải pháp này yêu cầu cả máy chủ và bộ điều khiển đã đọc và ghi quyền truy cập vào cùng một bit, điều này có thể làm phức tạp mạch và tăng chi phí của bộ điều khiển.

**13.3**

Bỏ phiếu có thể hiệu quả hơn so với I/O điều khiển ngắt. Đây là trường hợp khi I/O diễn ra thường xuyên và trong thời gian ngắn. Mặc dù một cổng nối tiếp duy nhất sẽ thực hiện I/O tương đối không thường xuyên và do đó nên sử dụng các ngắt, một tập hợp các cổng nối tiếp chẳng hạn như các cổng trong bộ tập trung đầu cuối có thể tạo ra nhiều thao tác I/O ngắn và việc ngắt đối với từng cổng có thể tạo ra tải nặng trên hệ thống. Vòng lặp bỏ phiếu đúng thời điểm có thể giảm tải mà không lãng phí nhiều tài nguyên thông qua vòng lặp mà không cần I/O.

**13.4**

Một cách tiếp cận kết hợp có thể chuyển đổi giữa bỏ phiếu và ngắt tùy thuộc vào độ dài của thời gian chờ thao tác I/O. Ví dụ: chúng tôi có thể thăm dò và lặp lại N lần và nếu thiết bị vẫn bận ở mức N+1, chúng tôi có thể đặt ngắt và ngủ. Cách tiếp cận này sẽ tránh chu kỳ chờ đợi bận rộn kéo dài. Phương pháp này sẽ là tốt nhất cho thời gian bận rất dài hoặc rất ngắn. Sẽ không hiệu quả nếu I/O hoàn thành ở N+T (trong đó T là một số chu kỳ nhỏ) do chi phí bỏ phiếu cộng với việc thiết lập và bắt các ngắt. Bỏ phiếu thuần túy là tốt nhất với thời gian chờ đợi rất ngắn. Ngắt là tốt nhất với thời gian chờ đợi lâu đã biết.

**13.5**

DMA tăng tính đồng thời của hệ thống bằng cách cho phép CPU thực hiện các tác vụ trong khi hệ thống DMA truyền dữ liệu qua hệ thống

và bus bộ nhớ. Thiết kế phần cứng rất phức tạp vì bộ điều khiển DMA phải được tích hợp vào hệ thống và hệ thống phải cho phép bộ điều khiển DMA làm chủ xe buýt. Đánh cắp chu kỳ cũng có thể cần thiết để cho phép bộ điều khiển CPU và DMA chia sẻ việc sử dụng bus bộ nhớ.

**13.6**

Hãy xem xét một hệ thống thực hiện 50% I/O và 50% tính toán. Tăng gấp đôi hiệu suất CPU trên hệ thống này sẽ tăng tổng hiệu suất hệ thống chỉ bằng 50%. Nhân đôi cả hai khía cạnh hệ thống sẽ tăng hiệu suất lên 100%. Nói chung, điều quan trọng là phải loại bỏ tắc nghẽn hệ thống hiện tại và tăng tổng thể hệ thống hiệu suất, thay vì tăng hiệu suất của các thành phần hệ thống riêng lẻ một cách mù quáng.

**13.7**

Trình điều khiển STREAMS điều khiển một thiết bị vật lý có thể tham gia vào hoạt động STREAMS. Mô-đun STREAMS sửa đổi luồng dữ liệu giữa đầu (giao diện người dùng) và trình điều khiển.

CHƯƠNG 21

21.1

Có hai nhược điểm chính với việc sử dụng các mô-đun. Đầu tiên là kích thước: quản lý mô-đun tiêu thụ kernel không thể phân trang bộ nhớ và một hạt nhân cơ bản với một số mô-đun được tải sẽ tiêu thụ nhiều bộ nhớ hơn một hạt nhân tương đương với trình điều khiển được biên dịch thành hình ảnh hạt nhân. Đây có thể là một điều rất quan trọng sự cố trên các máy có bộ nhớ vật lý hạn chế. Hạn chế thứ hai là các mô-đun có thể làm tăng độ phức tạp của quá trình bootstrap kernel. Thật khó để tải lên một bộ mô-đun từ đĩa nếu trình điều khiển cần truy cập đĩa đó, một mô-đun cần phải được tải. Kết quả là, việc quản lý kernel bootstrap với các mô-đun có thể yêu cầu thêm công việc từ phía quản trị viên: các mô-đun cần thiết để khởi động cần được đặt vào một hình ảnh ramdisk được tải cùng với hình ảnh hạt nhân ban đầu khi hệ thống khởi tạo.

Trong một số trường hợp, tốt hơn là sử dụng hạt nhân mô-đun và trong trường hợp khác trường hợp tốt hơn là sử dụng nhân với trình điều khiển thiết bị được liên kết trước. Ở đâu giảm thiểu kích thước của hạt nhân là rất quan trọng, sự lựa chọn sẽ phụ thuộc vào về tần suất sử dụng các trình điều khiển thiết bị khác nhau. Nếu chúng được sử dụng liên tục, thì các mô-đun không phù hợp. Điều này đặc biệt đúng khi trình điều khiển là cần thiết cho chính quá trình khởi động. Mặt khác, nếu một số trình điều khiển không phải lúc nào cũng cần thiết, sau đó cơ chế mô-đun cho phép tải và dỡ các trình điều khiển đó theo yêu cầu, có khả năng cung cấp một tiết kiệm ròng trong bộ nhớ vật lý.

Trường hợp hạt nhân được xây dựng phải có thể sử dụng được trên nhiều loại của các máy rất khác nhau, thì việc xây dựng nó với các mô-đun rõ ràng là tốt hơn là sử dụng một hạt nhân với hàng tá trình điều khiển không cần thiết tốn bộ nhớ. Đây là trường hợp đặc biệt đối với các hạt nhân được phân phối thương mại, nơi hỗ trợ nhiều loại phần cứng nhất trong cách đơn giản nhất có thể là ưu tiên. Tuy nhiên, nếu một hạt nhân đang được xây dựng cho một máy duy nhất có cấu hình được biết trước, sau đó biên dịch và sử dụng các mô-đun có thể chỉ đơn giản là một sự phức tạp không cần thiết. Trong những trường hợp như thế này, việc sử dụng mô-đun cũng có thể là một vấn đề của hương vị.