QUÁ TRÌNH

I Mục đích

Sau khi học xong chương này, người học nắm được những kiến thức sau:

- Hiểu các khái niệm về quá trình
- Hiểu cách lập thời biểu quá trình
- Biết các thao tác trên quá trình
- Hiểu cách giao tiếp liên quá trình

II Giới thiệu

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

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

III Khái niệm quá trình

Một vấn đề cần thảo luận là cái gì được gọi trong tất cả hoạt động của CPU? Một hệ thống bó thực thi công việc, trái lại một hệ thống chia thời thực thi chương trình người dùng hay tác vụ. Thậm chí trên hệ thống đơn người dùng như Microsoft Windows và Macintosh OS, một người dùng có thể chạy nhiều chương trình tại một thời điểm: bộ xử lý văn bản, trình duyệt web, e-mail. Thậm chí nếu người dùng có thể thực thi chỉ một quá trình tại một thời điểm, thì một hệ điều hành cần hỗ trợ những hoạt động được lập trình bên trong, như quản lý bộ nhớ. Trong nhiều khía cạnh, tất cả hoạt động là tương tự vì thế chúng ta gọi tất cả chúng là quá trình.

III.1 Quá trình

Thật vậy, một quá trình là một chương trình đang thực thi. Một quá trình không chỉ là mà chương trình, nó còn bao gồm hoạt động hiện hành như được hiện diện bởi giá trị của bộ đếm chương trình và nội dung các thanh ghi của bộ xử lý. Ngoài ra, một quá trình thường chứa ngăn xếp quá trình, chứa dữ liệu tạm thời (như các tham số phương thức, các địa chỉ trả về, các biến cục bộ) và phần dữ liệu chứa các biến toàn cục.

Chúng ta nhấn mạnh rằng, một chương trình không phải là một quá trình; một chương trình là một thực thể thụ động, như nội dung của các tập tin được lưu trên đĩa, trái lại một quá trình là một thực thể chủ động, với một bộ đếm chương trình xác định chỉ thị lệnh tiếp theo sẽ thực thi và tập hợp tài nguyên có liên quan.

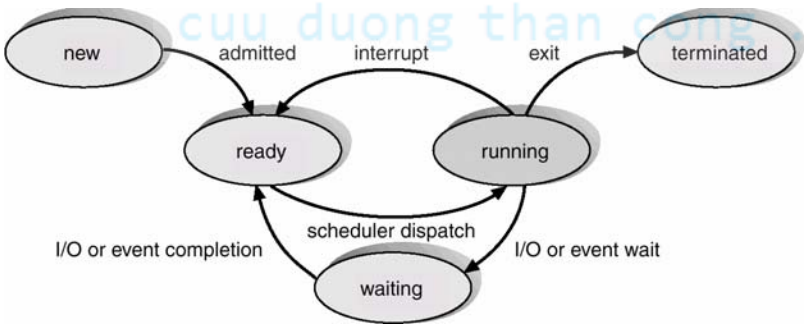
Mặc dù hai quá trình có thể được liên kết với cùng chương trình nhưng chúng được chứa hai thứ tự thực thi riêng rẻ. Thí dụ, nhiều người dùng có thể đang chạy các bản sao của chương trình gởi nhận thư, hay cùng người dùng có thể nạp lên nhiều bản sao của một chương trình soạn thảo văn bản. Mỗi bản sao của chúng là một quá trình riêng và mặc dù các phần văn bản là giống nhau, các phần dữ liệu khác nhau. Ngoài ra, một quá trình có thể tạo ra nhiều quá trình khi nó thực thi.

III.2 Trạng thái quá trình

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

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

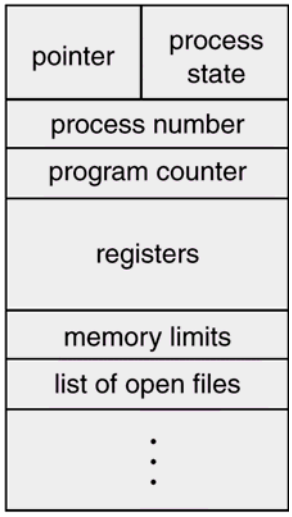
Các tên trạng thái này là bất kỳ, và chúng khác nhau ở các hệ điều hành khác nhau. Tuy nhiên, các trạng thái mà chúng hiện diện được tìm thấy trên tất cả hệ thống. Các hệ điều hành xác định mô tả trạng thái quá trình. Chỉ một quá trình có thể đang chạy tức thì trên bất kỳ bộ xử lý nào mặc dù nhiều quá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



Hình 0-1-Lưu đồ trạng thái quá trình

III.3 Khối điều khiển quá trình

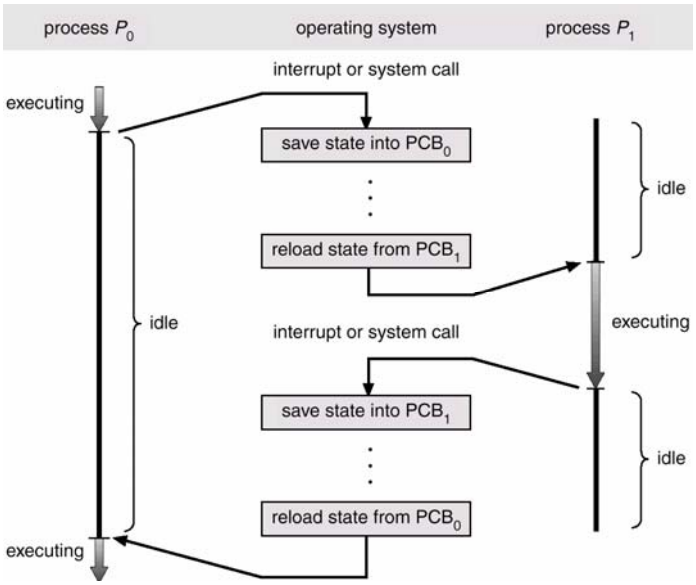
Mỗi quá trình được hiện diện trong hệ điều hành bởi một **khối điều khiển quá trình** (Process Control Block-PCB) – cũng được gọi khối điều khiển tác vụ. Một PCB được hiển thị trong hình III-2. Nó chứa nhiều phần thông tin được gắn liền với một quá trình xác định, gồm:



Hình 0-2-Khối điều khiển quá trình

- **Trạng thái quá trình** (process state): trạng thái có thể là mới, sẵn sàng, đang chạy, chờ đợi, kết thúc, …
- **Bộ đếm chương trình** (program counter): bộ đếm hiển thị địa chỉ của chỉ thị kế tiếp được thực thi cho quá trình này.
- **Các thanh ghi** (registers) CPU: các thanh ghi khác nhau về số lượng và loại, phụ thuộc vào kiến trúc máy tính. Chúng gồm các bộ tổng (accumulators), các thanh ghi chỉ mục, các con trỏ ngăn xếp, và các thanh ghi đa năng (general-purpose registers), cùng với thông tin mã điều kiện (condition-code information). Cùng với bộ đếm chương trình, thông tin trạng thái này phải được lưu khi một ngắt xảy ra, cho phép quá trình được tiếp tục một cách phù hợp sau đó (Hình III.3).
- **Thông tin lập thời biểu CPU** (CPU-scheduling information): thông tin gồm độ ưu tiên của quá trình, các con trỏ chỉ tới các hàng đợi lập thời biểu, và bất kỳ tham số lập thời biểu khác.
- **Thông tin quản lý bộ nhớ** (Memory-management information): thông tin này có thể gồm những thông tin như giá trị của các thanh ghi nền và thanh ghi giới hạn, các bảng trang hay các bảng phân đoạn, phụ thuộc hệ thống bộ nhớ được dùng bởi hệ điều hành.
- **Thông tin tính toán** (accounting information): thông tin này gồm lượng CPU và thời gian thực được dùng, công việc hay số quá trình,…
- **Thông tin trạng thái nhập/xuất** (I/O status information): thông tin này gồm danh sách của thiết bị nhập/xuất được cấp phát quá trình này, một danh sách các tập tin đang mở,..

PCB đơn giản phục vụ như kho chứa cho bất cứ thông tin khác nhau từ quá trình này tới quá trình khác.



Hình 0-3-Lưu đồ hiển thị việc chuyển CPU từ quá trình này tới quá trình khác

III.4 Luồng

Mô hình quá trình vừa được thảo luận ngụ ý rằng một quá trình là một chương trình thực hiện một luồng đơn thực thi. Thí dụ, nếu một quá trình đang chạy một chương trình xử lý văn bản, một luồng đơn của chỉ thị đang được thực thi. Đây là một luồng điều khiển đơn cho phép quá trình thực thi chỉ một tác vụ tại một thời điểm. Thí dụ, người dùng không thể cùng lúc nhập các ký tự và chạy bộ kiểm tra chính tả trong cùng một quá trình. Nhiều hệ điều hành hiện đại mở rộng khái niệm quá trình để cho phép một quá trình có nhiều luồng thực thi. Do đó, chúng cho phép thực hiện nhiều hơn một tác vụ tại một thời điểm.

IV Lập thời biểu quá trình

Mục tiêu của việc đa chương là có vài quá trình chạy tại tất cả thời điểm để tận dụng tối đa việc sử dụng CPU. Mục tiêu của chia thời là chuyển CPU giữa các quá trình thường xuyên để người dùng có thể giao tiếp với mỗi chương trình trong khi đang chạy. Một hệ thống đơn xử lý chỉ có thể chạy một quá trình. Nếu nhiều hơn một quá trình tồn tại, các quá trình còn lại phải chờ cho tới khi CPU rảnh và có thể lập thời biểu lại.

IV.1 Hàng đợi lập thời biểu

Khi các quá trình được đưa vào hệ thống, chúng được đặt vào hàng đợi công việc. Hàng đợi chứa tất cả quá trình trong hệ thống. Các quá trình đang nằm trong bộ nhớ chính sẵn sàng và chờ để thực thi được giữ trên một danh sách được gọi là hàng đợi sẵn sàng. Hàng đợi này thường được lưu như một danh sách liên kết. Đầu của hàng đợi sẵn sàng chứa hai con trỏ: một chỉ đến PCB đầu tiên và một chỉ tới PCB cuối cùng trong danh sách. Chúng ta bổ sung thêm trong mỗi PCB một trường con trỏ chỉ tới PCB kế tiếp trong hàng đợi sẵn sàng.

Hệ điều hành cũng có các hàng đợi khác. Khi một quá trình được cấp phát CPU, nó thực thi một khoảng thời gian và cuối cùng kết thúc, được ngắt, hay chờ một

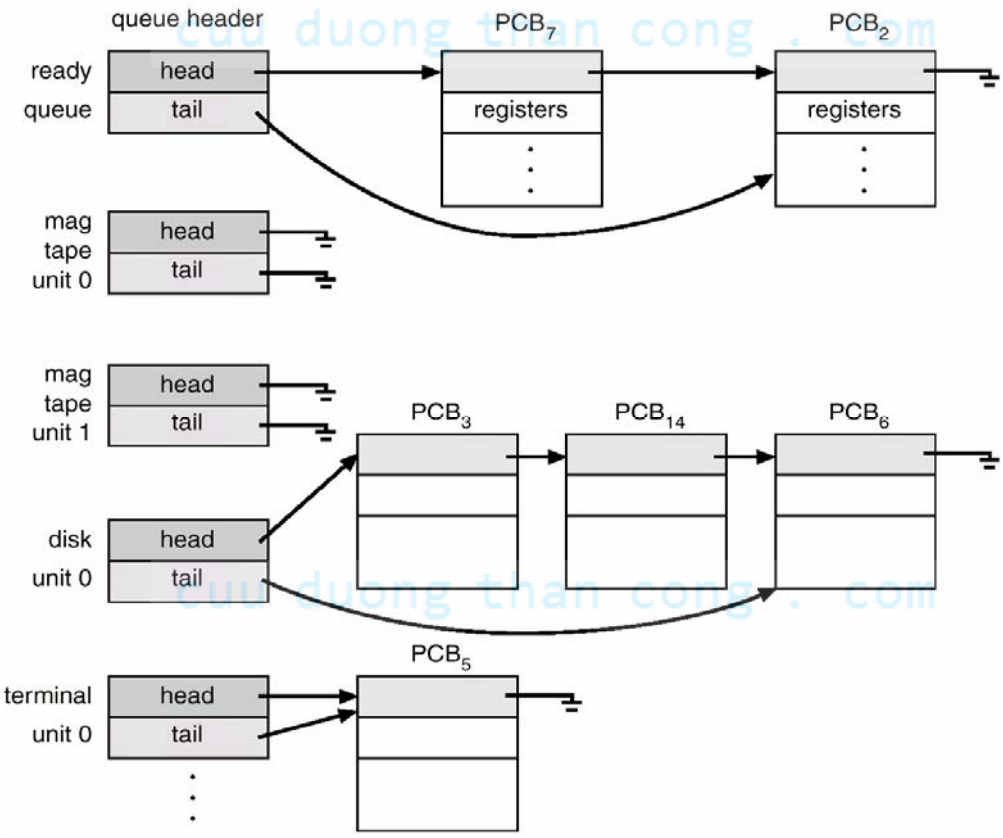
sự kiện xác định xảy ra, chẳng hạn như hoàn thành một yêu cầu nhập/xuất. Trong trường hợp yêu cầu nhập/xuất, một yêu cầu có thể là ổ đĩa băng từ tận hiến hay một thiết bị được chia sẻ như đĩa. Vì hệ thống có nhiều quá trình, đĩa có thể bận với yêu cầu nhập/xuất của các quá trình khác. Do đó, quá trình phải chờ đĩa. Danh sách quá trình chờ một thiết bị nhập/xuất cụ thể được gọi là hàng đợi thiết bị. Mỗi thiết bị có hàng đợi của chính nó như hình III.4.

Một biểu diễn chung của lập thời biểu quá trình là một lưu đồ hàng đợi, như hình III.5. Mỗi hình chữ nhật hiện diện một hàng đợi. Hai loại hàng đợi được hiện diện: hàng đợi sẵn sàng và tập các hàng đợi thiết bị, vòng tròn hiện diện tài nguyên phục vụ hàng đợi, các mũi tên hiển thị dòng các quá trình trong hệ thống.

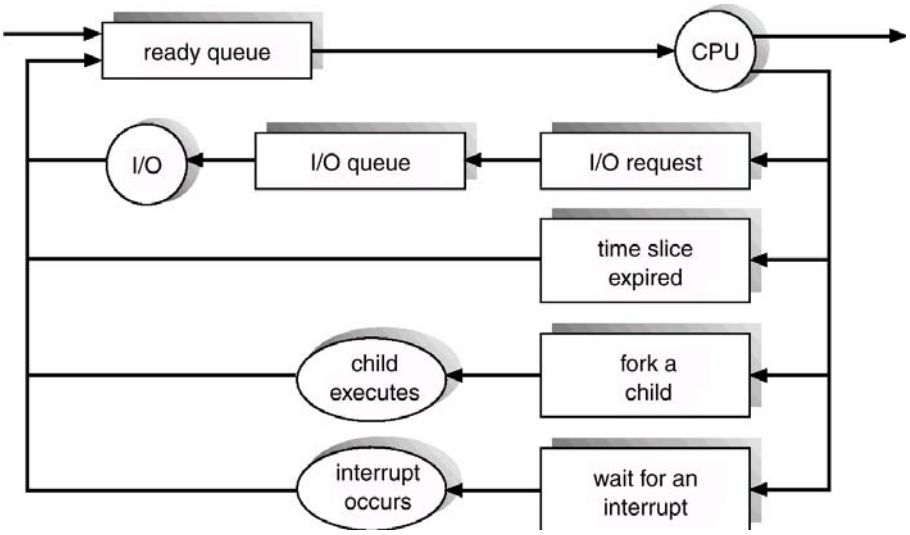
Một quá trình mới khởi đầu được đặt vào hàng đợi. Nó chờ trong hàng đợi sẵn sàng cho tới khi nó được chọn thực thi. Một khi quá trình được gán tới CPU và đang thực thi, một trong nhiều sự kiện có thể xảy ra:

- Quá trình có thể phát ra một yêu cầu nhập/xuất và sau đó được đặt vào trong hàng đợi nhập/xuất.
- Quá trình có thể tạo một quá trình con và chờ cho quá trình con kết thúc
- Khi một ngắt xảy ra, quá trình có thể bị buộc trả lại CPU và được đặt trở lại trong hàng đợi sẵn sàng.

Trong hai trường hợp đầu, cuối cùng quá trình chuyển từ trạng thái chờ tới trạng thái sẵn sàng và sau đó đặt trở lại vào hàng đợi sẵn sàng. Một quá trình tiếp tục chu kỳ này cho tới khi nó kết thúc. Tại thời điểm kết thúc nó bị xoá từ tất cả hàng đợi. Sau đó, PCB và tài nguyên của nó được thu hồi.



Hình 0-4-Hàng đợi sẵn sàng và các hàng đợi nhập/xuất khác nhau



Hình 0-5-Biểu diễn lưu đồ hàng đợi của lập thời biểu quá trình

IV.2 Bộ định thời biểu

Một quá trình di dời giữa hai hàng đợi định thời khác nhau suốt thời gian sống của nó. Hệ điều hành phải chọn, cho mục đích định thời, các quá trình từ các hàng đợi này. Quá trình chọn lựa này được thực hiện bởi bộ định thời hợp lý.

Trong hệ thống bó, thường nhiều hơn một quá trình được gởi đến hơn là có thể được thực thi tức thì. Các quá trình này được lưu tới thiết bị lưu trữ (như đĩa), nơi chúng được giữ cho việc thực thi sau đó. **Bộ định thời dài** (long-term scheduler) hay bộ định thời công việc (job scheduler), chọn các quá trình từ vùng đệm và nạp chúng vào bộ nhớ để thực thi. **Bộ định thời ngắn** (short-term scheduler) hay bộ định thời CPU chọn một quá trình từ các quá trình sẵn sàng thực thi và cấp phát CPU cho quá trình đó.

Sự khác biệt chủ yếu giữa hai bộ định thời là tính thường xuyên của việc thực thi. Bộ định thời CPU phải chọn một quá trình mới cho CPU thường xuyên. Một quá trình có thể thực thi chỉ một vài mili giây trước khi chờ yêu cầu nhập/xuất. Bộ định thời CPU thường thực thi ít nhất một lần mỗi 100 mili giây. Vì thời gian ngắn giữa việc thực thi nên bộ định thời phải nhanh. Nếu nó mất 10 mili giây để quyết định thực thi một quá trình 100 mili giây thì 10/(100+10) = 9 phần trăm của CPU đang được dùng (hay bị lãng phí) đơn giản cho định thời công việc.

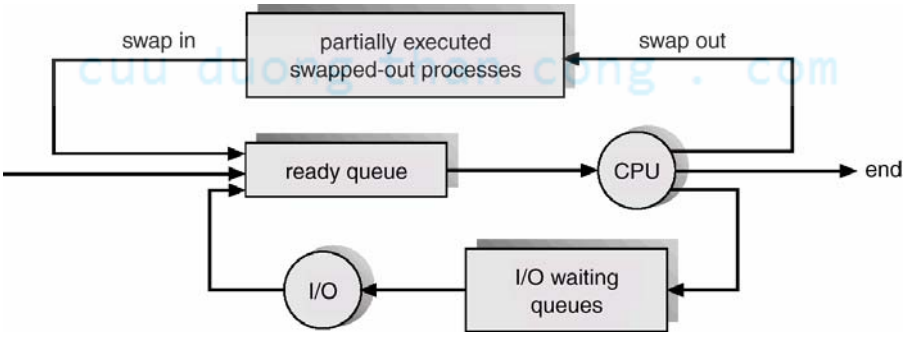
Ngược lại, bộ định thời công việc thực thi ít thường xuyên hơn. Có vài phút giữa việc tạo các quá trình mới trong hệ thống. Bộ định thời công việc điều khiển mức độ đa chương – số quá trình trong bộ nhớ. Nếu mức độ đa chương ổn định thì tốc độ trung bình của việc tạo quá trình phải bằng tốc độ khởi hành trung bình của quá trình rời hệ thống. Vì khoảng thời gian dài hơn giữa việc thực thi nên bộ định thời công việc có thể cấp nhiều thời gian hơn để chọn một quá trình thực thi.

Bộ định thời công việc phải thực hiện một chọn lựa cẩn thận. Thông thường, hầu hết các quá trình có thể được mô tả như là **quá trình hướng nhập/xuất** (I/O-bound proces) hay **quá trình hướng CPU** (CPU-bound process). Một quá trình hướng nhập/xuất mất nhiều thời gian hơn để thực hiện nhập/xuất hơn thời gian tính toán. Ngược lại, một quá trình hướng CPU phát sinh các yêu cầu nhập/xuất không thường xuyên, dùng thời gian để thực hiện việc tính toán hơn một quá trình hướng nhập/xuất dùng. Bộ định thời công việc nên chọn sự kết hợp hài hoà giữa quá trình

hướng nhập/xuất và quá trình hướng CPU. Nếu tất cả quá trình là hướng nhập/xuất thì hàng đợi sẵn sàng sẽ luôn rỗng và bộ định thời CPU sẽ có ít công việc để thực hiện. Nếu tất cả quá trình là hướng CPU thì hàng đợi nhập/xuất sẽ luôn rỗng, các thiết bị sẽ không được sử dụng và hệ thống sẽ mất cân bằng. Hệ thống với năng lực tốt nhất sẽ có sự kết hợp các quá trình hướng CPU và hướng nhập/xuất.

Trên một vài hệ thống, bộ định thời công việc có thể không có hay rất ít. Thí dụ, các hệ thống chia thời như UNIX thường không có bộ định thời công việc nhưng đơn giản đặt mỗi quá trình mới vào bộ nhớ cho bộ định thời CPU. Khả năng ổn định của hệ thống này phụ thuộc vào giới hạn vật lý (như số lượng thiết bị cuối sẵn dùng) hay tính tự nhiên tự chuyển đổi của người dùng. Nếu năng lực thực hiện giảm tới mức độ không thể chấp nhận được thì một số người dùng sẽ thoát khỏi hệ thống.

Một số hệ thống như hệ chia thời có thể đưa vào một cấp độ định thời bổ sung hay định thời trung gian. **Bộ định thời trung gian** (medium-term process) này (được hiển thị trong lưu đồ hình III-6) xóa các quá trình ra khỏi bộ nhớ (từ sự canh tranh CPU) và do đó giảm mức độ đa chương. Tại thời điểm sau đó, quá trình có thể được đưa trở lại bộ nhớ và việc thực thi của nó có thể được tiếp tục nơi nó bị đưa ra. Cơ chế này được gọi là **hoán vị** (swapping). Quá trình được hoán vị ra và sau đó được hoán vị vào bởi bộ định thời trung gian. Hoán vị là cần thiết để cải tiến sự trộn lẫn quá trình (giữa các quá trình hướng nhập/xuất và hướng CPU), hay vì một thay đổi trong yêu cầu bộ nhớ vượt quá kích thước bộ nhớ sẵn dùng. Hoán vị sẽ được thảo luận trong chương sau.



IV.3 Chuyển ngữ cảnh

Chuyển CPU tới một quá trình khác yêu cầu lưu trạng thái của quá trình cũ và nạp trạng thái được lưu cho quá trình mới. Tác vụ này được xem như **chuyển ngữ cảnh** (context switch). **Ngữ cảnh** của quá trình được hiện diện trong PCB của quá trình; Nó chứa giá trị các thanh ghi, trạng thái quá trình (hình III.1) và thông tin quản lý bộ nhớ. Khi chuyển ngữ cảnh ngữ cảnh xảy ra, nhân lưu ngữ cảnh của quá trình cũ trong PCB của nó và nạp ngữ cảnh được lưu của quá trình mới được định thời để chạy. Thời gian chuyển ngữ cảnh là chi phí thuần vì hệ thống không thực hiện công việc có ích trong khi chuyển. Tốc độ của nó khác từ máy này tới máy khác phụ thuộc vào tốc độ bộ nhớ, số lượng thanh ghi phải được chép và sự tồn tại của các chỉ thị đặc biệt (như chỉ thị để nạp và lưu tất cả thanh ghi). Điển hình dãy tốc độ từ 1 tới 1000 mili giây.

Những lần chuyển đổi ngữ cảnh phụ thuộc nhiều vào hỗ trợ phần cứng. Thí dụ, vài bộ xử lý (như Sun UltraSPARC) cung cấp nhiều tập thanh ghi. Một chuyển ngữ cảnh đơn giản chứa chuyển đổi con trỏ tới tập thanh ghi hiện hành. Dĩ nhiên, nếu quá trình hoạt động vượt quá tập thanh ghi thì hệ thống sắp xếp lại dữ liệu thanh ghi tới và từ bộ nhớ. Cũng vì thế mà hệ điều hành phức tạp hơn và nhiều công việc được làm

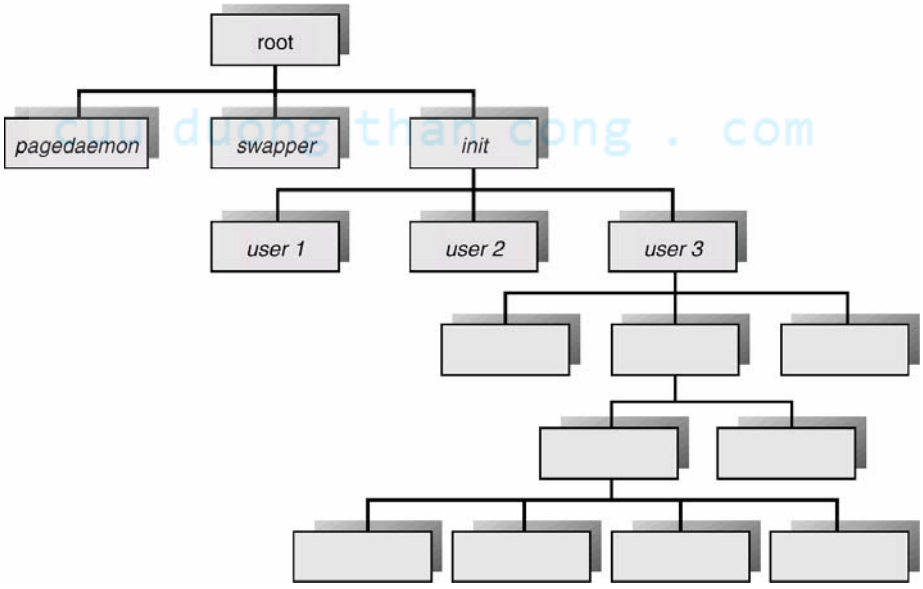
hơn trong khi chuyển ngữ cảnh. Kỹ thuật quản lý bộ nhớ nâng cao có thể yêu cầu dữ liệu bổ sung để được chuyển với mỗi ngữ cảnh. Thí dụ, không gian địa chỉ của quá trình hiện hành phải được lưu khi không gian của tác vụ kế tiếp được chuẩn bị dùng. Không gian địa chỉ được lưu như thế nào và lượng công việc được yêu cầu để lưu nó phụ thuộc vào phương pháp quản lý bộ nhớ của hệ điều hành. Chuyển ngữ cảnh có thể dẫn đến thắt cổ chai năng lực thực hiện vì thế các lập trình viên đang sử dụng các cấu trúc mới để tránh nó bất cứ khi nào có thể.

V Thao tác trên quá trình

Các quá trình trong hệ thống có thể thực thi đồng hành và chúng phải được tạo và xóa tự động. Do đó, hệ điều hành phải cung cấp một cơ chế (hay phương tiện) cho việc tạo quá trình và kết thúc quá trình.

V.1 Tạo quá trình

Quá trình có thể tạo nhiều quá trình mới, bằng một lời gọi hệ thống **create-process**, trong khi thực thi. Quá trình tạo gọi là quá trình cha, ngược lại các quá trình mới được gọi là quá trình con của quá trình đó. Mỗi quá trình mới này sau đó có thể tạo các quá trình khác, hình thành một cây quá trình (hình III-7).



Hình 0-7-Cây quá trình trên một hệ thống UNIX điển hình

Thông thường, một quá trình sẽ cần các tài nguyên xác định (như thời gian CPU, bộ nhớ, tập tin, thiết bị nhập/xuất) để hoàn thành tác vụ của nó. Khi một quá trình tạo một quá trình con, quá trình con có thể nhận tài nguyên của nó trực tiếp từ hệ điều hành hay nó có thể bị ràng buộc tới một tập con các tài nguyên của quá trình cha. Quá trình cha có thể phải phân chia các tài nguyên giữa các quá trình con hay có thể chia sẻ một số tài nguyên (như bộ nhớ và tập tin) giữa nhiều quá trình con. Giới hạn một quá trình con tới một tập con tài nguyên của quá trình cha ngăn chặn bất cứ quá trình từ nạp chồng hệ thống bằng cách tạo quá nhiều quá trình con.

Ngoài ra, khi một quá trình được tạo nó lấy tài nguyên vật lý và luận lý khác nhau, dữ liệu khởi tạo (hay nhập) có thể được truyền từ quá trình cha tới quá trình con. Thí dụ, xét một quá trình với toàn bộ chức năng là hiển thị trạng thái của một tập

tin F_1 trên màn hình. Khi nó được tạo, nó sẽ lấy dữ liệu vào từ quá trình cha, tên của tập tin F_1 và nó sẽ thực thi dùng dữ liệu đó để lấy thông tin mong muốn.

Nó có thể cũng lấy tên của thiết bị xuất. Một số hệ điều hành chuyển tài nguyên tới quá trình con. Trên hệ thống như thế, quá trình mới có thể lấy hai tập tin đang mở, F_1 và thiết bị cuối, và có thể chỉ cần chuyển dữ liệu giữa hai tập tin.

Khi một quá trình tạo một quá trình mới, hai khả năng có thể tồn tại trong thuật ngữ của việc thực thi:

- Quá trình cha tiếp tục thực thi đồng hành với quá trình con của nó.
- Quá trình cha chờ cho tới khi một vài hay tất cả quá trình con kết thúc.

Cũng có hai khả năng trong thuật ngữ không gian địa chỉ của quá trình mới:

- Quá trình con là bản sao của quá trình cha.
- Quá trình con có một chương trình được nạp vào nó.

Để hiển thị việc cài đặt khác nhau này, chúng ta xem xét hệ điều hành UNIX. Trong UNIX, mỗi quá trình được xác định bởi **danh biểu quá trình** (process identifier), là số nguyên duy nhất. Một quá trình mới được tạo bởi lời gọi hệ thống **fork**. Quá trình mới chứa bản sao của không gian địa chỉ của quá trình gốc. Cơ chế này cho phép quá trình cha giao tiếp dễ dàng với quá trình con. Cả hai quá trình (cha và con) tiếp tục thực thi tại chỉ thị sau khi lời gọi hệ thống **fork**, với một sự khác biệt: mã trả về cho lời gọi hệ thống fork là không cho quá trình mới (con), ngược lại danh biểu quá trình (khác không) của quá trình con được trả về tới quá trình cha.

Điển hình lời gọi hệ thống **execlp** được dùng sau lời gọi hệ thống **fork** bởi một trong hai quá trình để thay thế không gian bộ nhớ với quá trình mới. Lời gọi hệ thống **execlp** nạp tập tin nhị phân vào trong bộ nhớ-xóa hình ảnh bộ nhớ của chương trình chứa lời gọi hệ thống **execlp** - và bắt đầu việc thực thi của nó. Trong cách thức này, hai quá trình có thể giao tiếp và sau đó thực hiện cách riêng của nó. Sau đó, quá trình cha có thể tạo nhiều hơn quá trình con, hay nếu nó không làm gì trong thời gian quá trình con chạy thì nó sẽ phát ra lời gọi hệ thống **wait** để di chuyển nó vào hàng đợi sẵn sàng cho tới khi quá trình con kết thúc. Chương trình C (hình III-8 dưới đây) hiển thị lời gọi hệ thống UNIX được mô tả trước đó. Quá trình cha tạo một quá trình con sử dụng lời gọi hệ thống **fork**. Bây giờ chúng ta có hai quá trình khác nhau chạy một bản sao của cùng chương trình. Giá trị **pid** cho quá trình con là 0; cho quá trình cha là một số nguyên lớn hơn 0. Quá trình con phủ lắp không gian địa chỉ của nó với lệnh của UNIX là **/bin/ls** (được dùng để liệt kê thư mục) dùng lời gọi hệ thống **execlp**. Quá trình cha chờ cho quá trình con hoàn thành với lời gọi hệ thống **wait**. Khi quá trình con hoàn thành, quá trình cha bắt đầu lại từ lời gọi hệ thống **wait** nơi nó hoàn thành việc sử dụng lời gọi hệ thống **exit**.

Ngược lại, hệ điều hành DEC VMS tạo một quá trình mới, nạp chương trình xác định trong quá trình đó và bắt đầu thực thi nó. Hệ điều hành Microsoft Windows NT hỗ trợ cả hai mô hình: không gian địa chỉ của quá trình cha có thể được sao lại hay quá trình cha có thể xác định tên của một chương trình cho hệ điều hành nạp vào không gian địa chỉ của quá trình mới.

```
#include<stdio.h>
main(int argc, char* argv[])
{
        int     pid;
        /*fork another process*/
        pid     =       fork();
        if(pid<0){      /*error occurred */
                fprintf(stderr, "Fork Failed");
                exit(-1);
}
else if (pid==0){       /*child process*/
                execlp("/bin/ls","ls",NULL);
}
else {  /*parent process*/
        /*parent will wait for the child to complete*/
        wait(NULL);
        printf("Child Complete");
        exit(0);
}
}
```

Hình 0-8-Chương trình C tạo một quá trình riêng rẻ

V.2 Kết thúc quá trình

Một quá trình kết thúc khi nó hoàn thành việc thực thi câu lệnh cuối cùng và yêu cầu hệ điều hành xóa nó bằng cách sử dụng lời gọi hệ thống **exit**. Tại thời điểm đó, quá trình có thể trả về dữ liệu (đầu ra) tới quá trình cha (bằng lời gọi hệ thống **wait**). Tất cả tài nguyên của quá trình –gồm bộ nhớ vật lý và luận lý, các tập tin đang mở, vùng đệm nhập/xuất-được thu hồi bởi hệ điều hành.

Việc kết thúc xảy ra trong các trường hợp khác. Một quá trình có thể gây kết thúc một quá trình khác bằng một lời gọi hệ thống hợp lý (thí dụ, abort). Thường chỉ có quá trình cha bị kết thúc có thể gọi lời gọi hệ thống như thế. Ngược lại, người dùng có thể tùy ý giết (kill) mỗi công việc của quá trình còn lại. Do đó, quá trình cha cần biết các định danh của các quá trình con. Vì thế khi một quá trình tạo một quá trình mới, định danh của mỗi quá trình mới được tạo được truyền tới cho quá trình cha.

Một quá trình cha có thể kết thúc việc thực thi của một trong những quá trình con với nhiều lý do khác nhau:

- Quá trình con sử dụng tài nguyên vượt quá mức được cấp. Điều này yêu cầu quá trình cha có một cơ chế để xem xét trạng thái của các quá trình con.
- Công việc được gán tới quá trình con không còn cần thiết nữa.
- Quá trình cha đang kết thúc và hệ điều hành không cho phép một quá trình con tiếp tục nếu quá trình cha kết thúc. Trên những hệ thống như thế, nếu một quá trình kết thúc (bình thường hoặc không bình thường), thì tất cả quá trình con cũng phải kết thúc. Trường hợp này được xem như kết thúc xếp tầng (cascading termination) thường được khởi tạo bởi hệ điều hành.

Để hiển thị việc thực thi và kết thúc quá trình, xem xét hệ điều hành UNIX chúng ta có thể kết thúc một quá trình dùng lời gọi hệ thống **exit**; nếu quá trình cha có thể chờ cho đến khi quá trình con kết thúc bằng lời gọi hệ thống **wait**. Lời gọi hệ thống **wait** trả về định danh của quá trình con bị kết thúc để quá trình cha có thể xác

định những quá trình con nào có thể kết thúc. Tuy nhiên, nếu quá trình cha kết thúc thì tất cả quá trình con của nó được gán như quá trình cha mới của chúng, quá trình khởi tạo (init process). Do đó, các quá trình con chỉ có một quá trình cha để tập hợp trạng thái và thống kê việc thực thi.

V.3 Hợp tác 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.

Thực thi đồng hành của các quá trình hợp tác yêu cầu các cơ chế cho phép các quá trình giao tiếp với các quá trình khác và đồng bộ hóa các hoạt động của chúng. Để minh họa khái niệm của các quá trình cộng tác, chúng ta xem xét bài toán người sản xuất-người tiêu thụ, là mô hình chung cho các quá trình hợp tác. Quá trình người sản xuất cung cấp thông tin được tiêu thụ bởi quá trình người tiêu thụ. Thí dụ, một chương trình in sản xuất các ký tự được tiêu thụ bởi trình điều khiển máy in. Một trình biên dịch có thể sản xuất mã hợp ngữ được tiêu thụ bởi trình hợp ngữ. Sau đó, trình hợp ngữ có sản xuất module đối tượng, được tiêu thụ bởi bộ nạp.

Để cho phép người sản xuất và người tiêu thụ chạy đồng hành, chúng ta phải có sẵn một vùng đệm chứa các sản phẩm có thể được điền vào bởi người sản xuất và được lấy đi bởi người tiêu thụ. Người sản xuất có thể sản xuất một sản phẩm trong khi người tiêu thụ đang tiêu thụ một sản phẩm khác. Người sản xuất và người tiêu thụ phải được đồng bộ để mà người tiêu thụ không cố gắng tiêu thụ một sản phẩm mà chưa được sản xuất. Trong trường hợp này, người tiêu thụ phải chờ cho tới khi các sản phẩm mới được tạo ra.

Bài toán người sản xuất-người tiêu thụ với vùng đệm không bị giới hạn (**unbounded-buffer**) thiết lập không giới hạn kích thước của vùng đệm. Người tiêu thụ có thể phải chờ các sản phẩm mới nhưng người sản xuất có thể luôn tạo ra sản phẩm mới. Vấn đề người sản xuất-người tiêu thụ với vùng đệm có kích thước giới hạn

(**bounded-buffer**) đảm bảo một kích thước cố định cho vùng đệm. Trong trường hợp này, người tiêu thụ phải chờ nếu vùng đệm rỗng, và người sản xuất phải chờ nếu vùng đệm đầy.

Vùng đệm có thể được cung cấp bởi hệ điều hành thông qua việc sử dụng phương tiện **giao tiếp liên quá trình** (interprocess-communication-IPC), hay được mã hóa cụ thể bởi người lập trình ứng dụng với việc sử dụng bộ nhớ được chia sẻ. Để chúng ta hiển thị một giải pháp chia sẻ bộ nhớ đối với vấn đề vùng đệm bị giới hạn (bounded-buffer). Quá trình người sản xuất và người tiêu thụ chia sẻ các biến sau:

```
#define BUFFER_SIZE     10
typedef struct{
        …
} item;
item buffer[BUFFER_SIZE];
int     in      =       0;
int     out     =       0;
```

Vùng đệm được chia sẻ được cài đặt như một mảng vòng với hai con trỏ luận lý: **in** và **out**. Biến **in** chỉ tới vị trí trống kế tiếp trong vùng đệm; **out** chỉ tới vị trí đầy đầu tiên trong vùng đệm. Vùng đệm là rỗng khi **in==out**; vùng đệm là đầy khi **((in + 1)%BUFFER_SIZE) ==out**.

Mã cho quá trình người sản xuất và người tiêu thụ được trình bày dưới đây. Quá trình người sản xuất có một biến **nextProduced** trong đó sản phẩm mới được tạo ra và được lưu trữ:

```
while (1) {
        /*produce an item in nextProduced*/
        while (((in + 1)%BUFFER_SIZE) ==out)
                ; /*do nothing*/
        buffer[in]      =       nextProduced;
        in              =       (in + 1) % BUFFER_SIZE;
}
```

Quá trình người tiêu thụ có biến cục bộ **nextConsumed** trong đó sản phẩm được tiêu thụ và được lưu trữ:

```
while (1){
        while (in==out)
                ; //nothing
        nextConsumed =      buffer[out];
        out     =       (out    +       1)      % BUFFER_SIZE;
        /*consume the item in nextConsumed*/
}
```

Cơ chế này cho phép nhiều nhất BUFFER_SIZE –1 trong vùng đệm tại cùng một thời điểm.

VI Giao tiếp liên quá trình

Trong phần trên chúng ta đã hiển thị cách mà các quá trình hợp tác có thể giao tiếp với nhau trong một môi trường chia sẻ bộ nhớ. Cơ chế yêu cầu các quá trình này chia sẻ nhóm vùng đệm chung và mã cho việc cài đặt vùng đệm được viết trực tiếp bởi người lập trình ứng dụng. Một cách khác đạt được cùng ảnh hưởng cho hệ điều hành là cung cấp phương tiện cho các quá trình hợp tác giao tiếp với nhau bằng một phương tiện giao tiếp liên quá trình (IPC).

IPC cung cấp một cơ chế cho phép một quá trình giao tiếp và đồng bộ các hoạt động của chúng mà không chia sẻ cùng không gian địa chỉ. IPC đặc biệt có ích trong môi trường phân tán nơi các quá trình giao tiếp có thể thường trú trên các máy tính khác được nối kết qua mạng. Thí dụ chương trình **chat** được dùng trên World Wide Web.

IPC được cung cấp bởi hệ thống truyền thông điệp, và các hệ thống truyền thông điệp có thể được định nghĩa trong nhiều cách. Trong phần này chúng ta sẽ xem xét những vấn đề khác nhau khi thiết kế các hệ thống truyền thông điệp.

VI.1 Hệ thống truyền thông điệp

Chức năng của hệ thống truyền thông điệp là cho phép các quá trình giao tiếp với các quá trình khác mà không cần sắp xếp lại dữ liệu chia sẻ. Chúng ta xem truyền thông điệp được dùng như một phương pháp giao tiếp trong vi nhân. Trong cơ chế này, các dịch vụ được cung cấp như các quá trình người dùng thông thường. Nghĩa là, các dịch vụ hoạt động bên ngoài nhân. Giao tiếp giữa các quá trình người dùng được thực hiện thông qua truyền thông điệp. Một phương tiện IPC cung cấp ít nhất hai hoạt động: **send(message)** và **receive(message)**.

Các thông điệp được gởi bởi một quá trình có thể có kích thước cố định hoặc biến đổi. Nếu chỉ các thông điệp có kích thước cố định được gởi, việc cài đặt cấp hệ thống là đơn giản hơn. Tuy nhiên, hạn chế này làm cho tác vụ lập trình sẽ phức tạp hơn. Ngoài ra, các thông điệp có kích thước thay đổi yêu cầu việc cài đặt mức hệ thống phức tạp hơn nhưng tác vụ lập trình trở nên đơn giản hơn.

Nếu quá trình P và Q muốn giao tiếp, chúng phải gởi các thông điệp tới và nhận thông điệp từ với nhau; một liên kết giao tiếp phải tồn tại giữa chúng. Liên kết này có thể được cài đặt trong những cách khác nhau. Ở đây chúng ta quan tâm đến cài đặt luận lý hơn là cài đặt vật lý. Có vài phương pháp cài đặt một liên kết và các hoạt động send/receive:

- Giao tiếp trực tiếp hay gián tiếp
- Giao tiếp đối xứng hay bất đối xứng
- Gởi bằng bản sao hay tham chiếu
- Thông điệp có kích thước cố định hay thay đổi

VI.2 Đặt tên

Các quá trình muốn giao tiếp phải có cách tham chiếu với nhau. Chúng có thể dùng giao tiếp trực tiếp hay gián tiếp.

VI.2.1 Giao tiếp trực tiếp

Với giao tiếp trực tiếp, mỗi quá trình muốn giao tiếp phải đặt tên rõ ràng người gởi và người nhận của giao tiếp. Trong cơ chế này, các hàm cơ sở send và receive được định nghĩa như sau:

- **Send(P, message)**: gởi một thông điệp tới quá trình P
- **Receive(Q, message)**: nhận một thông điệp từ quá trình Q

Một liên kết giao tiếp trong cơ chế này có những thuộc tính sau:

- Một liên kết được thiết lập tự động giữa mỗi cặp quá trình muốn giao tiếp. Các quá trình cần biết định danh của nhau khi giao tiếp.
- Một liên kết được nối kết với chính xác hai quá trình
- Chính xác một liên kết tồn tại giữa mỗi cặp quá trình.

Cơ chế này hiển thị tính đối xứng trong việc đánh địa chỉ: nghĩa là, cả hai quá trình gởi và nhận phải biết tên nhau để giao tiếp. Một thay đổi trong cơ chế này thực hiện tính bất đối xứng trong việc đánh địa chỉ. Chỉ người gởi biết tên của người nhận; người nhận không yêu cầu tên của người gởi. Trong cơ chế này các hàm cơ sở được định nghĩa như sau:

- **Send(P, message)**: gởi một thông điệp tới quá trình P
- **Receive(id, message)**: nhận một thông điệp từ bất kỳ quá trình nào; id khác nhau được đặt tên của quá trình mà giao tiếp xảy ra.

Sự bất lợi trong cả hai cơ chế đối xứng và không đối xứng là tính điều chỉnh của việc định nghĩa quá trình bị giới hạn. Thay đổi tên của một quá trình có thể cần xem xét tất cả định nghĩa quá trình khác. Tất cả tham chiếu tới tên cũ phải được tìm thấy để mà chúng có thể được thay đổi thành tên mới. Trường hợp này là không mong muốn từ quan điểm biên dịch riêng.

VI.2.2 Giao tiếp gián tiếp

Với giao tiếp gián tiếp, một thông điệp được gởi tới và nhận từ các hộp thư (mailboxes), hay cổng (ports). Một hộp thư có thể được hiển thị trừu tượng như một đối tượng trong đó các thông điệp có thể được đặt bởi các quá trình và sau đó các thông điệp này có thể được xóa đi. Mỗi hộp thư có một định danh duy nhất. Trong cơ chế này, một quá trình có thể giao tiếp với một vài quá trình khác bằng một số hộp thư khác nhau. Hai quá trình có thể giao tiếp chỉ nếu chúng chia sẻ cùng một hộp thư. Hàm cơ sở **send** và **receive** được định nghĩa như sau:

- **Send(A, message)**: gởi một thông điệp tới hộp thư A.
- **Receive(A, message)**: nhận một thông điệp từ hộp thư A.

Trong cơ chế này, một liên kết giao tiếp có các thuộc tính sau:

- Một liên kết được thiết lập giữa một cặp quá trình chỉ nếu cả hai thành viên của cặp có một hộp thư được chia sẻ.
- Một liên kết có thể được nối kết với nhiều hơn hai quá trình.
- Số các liên kết khác nhau có thể tồn tại giữa mỗi cặp quá trình giao tiếp với mỗi liên kết tương ứng với một hộp thư

Giả sử các quá trình P_1, P_2 và P_3 chia sẻ một hộp thư A. Quá trình P_1 gởi một thông điệp tới A trong khi P_2 và P_3 thực thi việc nhận từ A. Quá trình nào sẽ nhận thông điệp được gởi bởi P_1? Câu trả lời phụ thuộc cơ chế mà chúng ta chọn:

- Cho phép một liên kết được nối kết với nhiều nhất hai quá trình
- Cho phép nhiều nhất một quá trình tại một thời điểm thực thi thao tác nhận.
- Cho phép hệ thống chọn bất kỳ quá trình nào sẽ nhận thông điệp (nghĩa là, hoặc P_1 hoặc P_3 nhưng không phải cả hai sẽ nhận thông điệp). Hệ thống này có thể xác định người nhận tới người gởi.

Một hộp thư có thể được sở hữu bởi một quá trình hay bởi hệ điều hành. Nếu hộp thư được sở hữu bởi một quá trình (nghĩa là, hộp thư là một phần không gian địa chỉ của quá trình), sau đó chúng ta phân biệt giữa người sở hữu (người chỉ nhận thông điệp thông qua hộp thư này) và người dùng (người có thể chỉ gởi thông điệp tới hộp thư).

Vì mỗi hộp thư có một người sở hữu duy nhất nên không có sự lẫn lộn về người nhận thông điệp được gởi tới hộp thư này. Khi một quá trình sở hữu một hộp thư kết thúc, hộp thư biến mất. Sau đó, bất kỳ quá trình nào gởi thông điệp tới hộp thư này được thông báo rằng hộp thư không còn tồn tại nữa.

Ngoài ra, một hộp thư được sở hữu bởi hệ điều hành độc lập và không được gán tới bất kỳ quá trình xác định nào. Sau đó, hệ điều hành phải cung cấp một cơ chế cho phép một quá trình thực hiện như sau:

- Tạo một hộp thư mới
- Gởi và nhận các thông điệp thông qua hộp thư
- Xóa hộp thư

Mặc định, quá trình tạo hộp thư mới là người sở hữu hộp thư đó. Ban đầu, người sở hữu chỉ là một quá trình có thể nhận thông điệp thông qua hộp thư. Tuy nhiên, việc sở hữu và quyền nhận thông điệp có thể được chuyển tới các quá trình khác thông qua lời gọi hệ thống hợp lý. Dĩ nhiên, sự cung cấp này có thể dẫn đến nhiều người nhận cho mỗi hộp thư.

VI.2.3 Đồng bộ hóa

Giao tiếp giữa hai quá trình xảy ra bởi lời gọi hàm cơ sở **send** và **receive**. Có các tùy chọn thiết kế khác nhau cho việc cài đặt mỗi hàm cơ sở. Truyền thông điệp có thể là nghẽn (block) hay không nghẽn (nonblocking)-cũng được xem như đồng bộ và bất đồng bộ.

- **Hàm send nghẽn**: quá trình gởi bị nghẽn cho đến khi thông điệp được nhận bởi quá trình nhận hay bởi hộp thư.
- **Hàm send không nghẽn**: quá trình gởi gởi thông điệp và thực hiện tiếp hoạt động
- **Hàm receive nghẽn**: người nhận nghẽn cho đến khi thông điệp sẵn dùng
- **Hàm receive không nghẽn**: người nhận nhận lại một thông điệp hợp lệ hay rỗng
-

Sự kết hợp khác nhau giữa send và receive là có thể. Khi cả hai send và receive là nghẽn chúng ta có sự thống nhất giữa người gởi và người nhận.

VI.2.4 Tạo vùng đệm

Dù giao tiếp có thể là trực tiếp hay gián tiếp, các thông điệp được chuyển đổi bởi các quá trình giao tiếp thường trú trong một hàng đợt tạm thời. Về cơ bản, một hàng đợi như thế có thể được cài đặt trong ba cách:

- **Khả năng chứa là 0** (zero capacity): hàng đợi có chiều dài tối đa là 0; do đó liên kết không thể có bất kỳ thông điệp nào chờ trong nó. Trong trường hợp này, người gởi phải nghẽn cho tới khi người nhận nhận thông điệp.
- **Khả năng chứa có giới hạn** (bounded capacity): hàng đợi có chiều dài giới hạn *n*; do đó, nhiều nhất n thông điệp có thể thường trú trong nó. Nếu hàng đợi không đầy khi một thông điệp mới được gởi, sau đó nó được đặt trong hàng đợi (thông điệp được chép hay một con trỏ thông điệp được giữ) và người gởi có thể tiếp tục thực thi không phải chờ.

Tuy nhiên, liên kết có khả năng chứa giới hạn. Nếu một liên kết đầy, người gởi phải nghẽn cho tới khi không gian là sẵn dùng trong hàng đợi

- **Khả năng chứa không giới hạn** (unbounded capacity): Hàng đợi có khả năng có chiều dài không giới hạn; do đó số lượng thông điệp bất kỳ có thể chờ trong nó. Người gởi không bao giờ nghẽn.

Trường hợp khả năng chứa là 0 thường được xem như hệ thống thông điệp không có vùng đệm; hai trường hợp còn lại được xem như vùng đệm tự động.

VII Tóm tắt

Quá trình là một chương trình đang thực thi. Khi một quá trình thực thi, nó thay đổi trạng thái. Trạng thái của một quá trình được định nghĩa bởi một hoạt động hiện tại của quá trình đó. Mỗi quá trình có thể ở một trong những trạng thái sau: mới (new), sẵn sàng (ready), đang chạy (running), chờ (waiting), hay kết thúc (terminated). Mỗi quá trình được biểu diễn trong hệ điều hành bởi khối điều khiển quá trình của chính nó (PCB).

Một quá trình khi không thực thi, được đặt vào hàng đợi. Hai cấp chủ yếu của hàng đợi trong hệ điều hành là hàng đợi yêu cầu nhập/xuất và hàng đợi sẵn sàng. Hàng đợi sẵn sàng chứa tất cả quá trình sẵn sàng để thực thi và đang chờ CPU. Mỗi quá trình được biểu diễn bởi một PCB và các PCB có thể được liên kết với nhau để hình thành một hàng đợi sẵn sàng. **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.

Các quá trình trong hệ thống có thể thực thi đồng hành. Có nhiều lý do các thực thi đồng hành: chia sẻ thông tin, tăng tốc độ tính toán, hiệu chỉnh và tiện dụng. Thực thi đồng hành yêu cầu cơ chế cho việc tạo và xóa quá trình.

Quá trình thực thi trong hệ điều hành có thể là các quá trình độc lập hay các quá trình hợp tác. Các quá trình hợp tác phải có phương tiện giao tiếp với nhau. Chủ yếu, có hai cơ chế giao tiếp bổ sung cho nhau cùng tồn tại: chia sẻ bộ nhớ và hệ thống truyền thông điệp. Phương pháp chia sẻ bộ nhớ yêu cầu các quá trình giao tiếp chia sẻ một số biến. Các quá trình được mong đợi trao đổi thông tin thông qua việc sử dụng các biến dùng chung này. Trong hệ thống bộ nhớ được chia sẻ, nhiệm vụ cho việc cung cấp giao tiếp tách rời với người lập trình ứng dụng; chỉ hệ điều hành cung cấp hệ thống bộ nhớ được chia sẻ. Phương pháp truyền thông điệp cho phép các quá trình trong đổi thông điệp. Nhiệm vụ cung cấp giao tiếp có thể tách rời với hệ điều hành. Hai cơ chế này không loại trừ hỗ tương và có thể được dùng cùng một lúc trong phạm vi một hệ điều hành.