MÔN HỆ ĐIỀU HÀNH

Chương 2

QUẢN LÝ PROCESS & THREAD

1.1 Khái niệm process
1.2 Tạo, xóa process
1.3 Trạng thái process
1.4 Khái niệm thread
1.5 Lập lịch chạy các process
1.6 Các phương pháp lập lịch

Tài liệu tham khảo : chương 2, sách "Modern Operating Systems", Andrew S. Tanenbaum: , 2nd ed, Prentice Hall

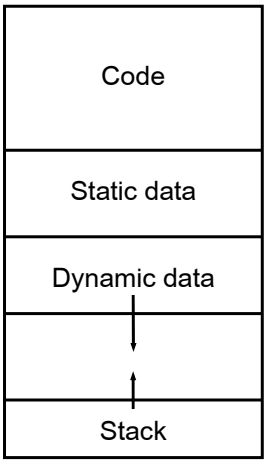
2.1 Giới thiệu process

- Chương trình (program) = danh sách các lệnh để giải quyết một vấn đề nào đó, được cất trên đĩa dưới dạng file.
- Khi chương trình được nạp vào RAM và CPU bắt đầu thi hành chương trình ở điểm nhập thì chương trình trở thành process, CPU thực thi hết lệnh này đến lệnh khác từ trên xuống hay theo sự điều khiển của lệnh đang thực thi.
- Process gồm 2 thành phần chính : danh sách các lệnh cấu thành thuật giải của chương trình và dữ liệu. Process tuần tự chỉ chứa 1 luồng thi hành lệnh cho 1 chương trình từ điểm nhập đến điểm kết thúc.

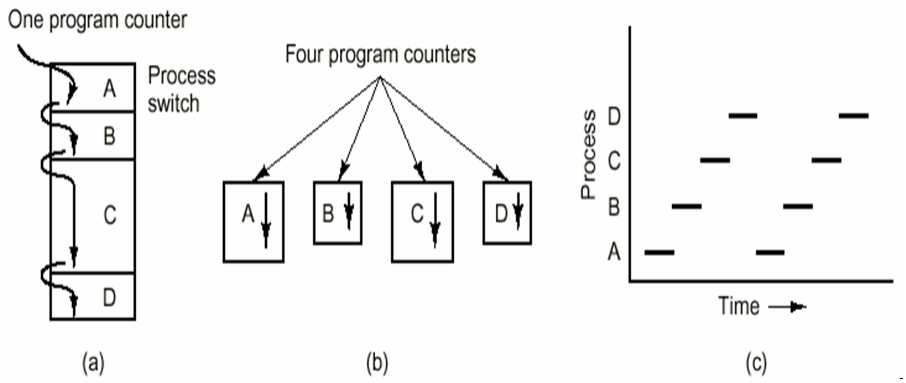
Giới thiệu process

Mỗi process có 1 vùng nhớ riêng, thường được chia thành các vùng thông tin khác nhau như sau :

- Vùng code : chứa danh sách mã lệnh của chương trình.
- Vùng static data : chứa các biến dữ liệu được khai báo tường minh trong chương trình.
- Vùng dynamic data : chứa các vùng nhớ dữ liệu được cấp phát động (thông qua new, maloc…). Kích thước vùng này biến động theo thời gian.
- Vùng stack : phục vụ cho việc gọi hàm trong chương trình. Kích thước vùng này biến động theo thời gian.



Giới thiệu process



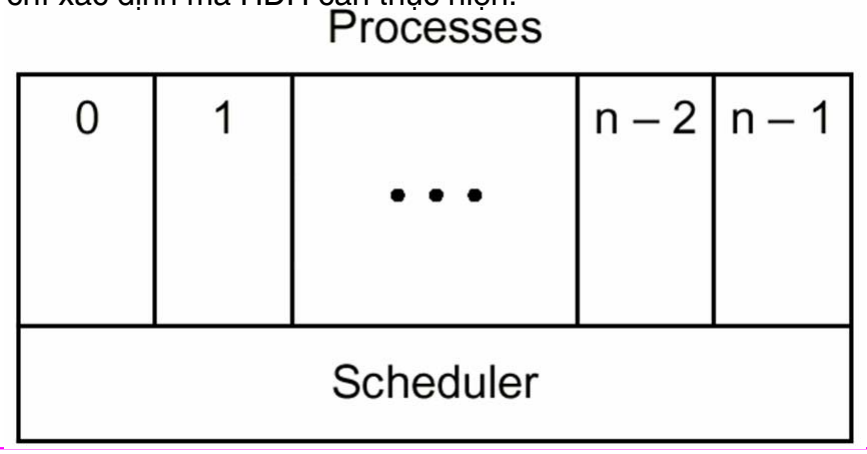
(a) Góc nhìn vật lý : từng thời điểm CPU chỉ chạy 1 process.

(b) Góc nhìn user : 4 process đang chạy song hành.

(c) đồ thị chạy của từng process theo thời gian.

Giới thiệu process

Trình lập lịch (Scheduler) là 1 module chức năng của HĐH, nó sẽ điều khiển việc chạy các process theo thời gian để thỏa mãn các tiêu chí xác định mà HĐH cần thực hiện.



2.2 Tạo process

Một process mới được tạo ra từ các sự kiện sau :

1. Do hệ thống tự tạo theo nhu cầu quản lý hệ thống (thường đây là các process hệ thống được tạo lúc khởi động HĐH).
2. Do người dùng kích hoạt chạy 1 phần mềm.
3. Do thuật giải của 1 phần mềm đang chạy, nó gọi dịch vụ CreateProcess để tạo process mới theo yêu cầu riêng.

Xóa process

Một process sẽ bị xóa từ các sự kiện sau :

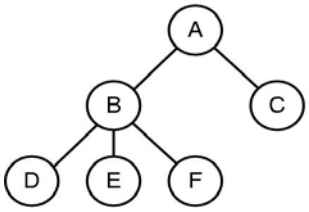
1. Nội tại :
    1. Khi chương trình chạy đến lệnh kết thúc bình thường của giải thuật.
    2. Khi chương trình chạy đến lệnh gây lỗi mà người lập trình trù liệu.
    3. Khi chương trình chạy đến lệnh gây lỗi mà người lập trình không trù liệu, trong trường hợp này hệ thống sẽ phát hiện lỗi và xóa process.
2. Bên ngoài : do process khác yêu cầu hệ thống giết.

Cây phân cấp các process

Miêu tả mối quan hệ "tạo mới" giữa các process :

Một cây process (process tree)

- A đã tạo hai process con : B và C
- B đã tạo ba process con : D, E, và F



Linux dùng khái niệm "process group" để quản lý cây process.

Windows không quản lý cây process, mọi process đầu ngang cấp.

2.3 Trạng thái process

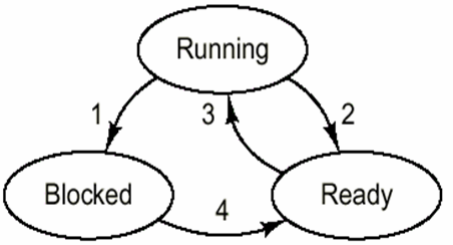
Theo thời gian hoạt động, process sẽ thay đổi trạng thái. Có 2 cấp độ trạng thái :

- Trạng thái vĩ mô : do HĐH đặt ra để quản lý process.
- Trạng thái vi mô : trạng thái chi tiết sau từng lệnh máy được thực thi.

Thường có 3 trạng thái vĩ mô phổ biến : Running (đang chiếm CPU và chạy), Ready (chờ CPU để chạy), Blocked (bị giam vì chờ I/O).

Các sự kiện gây ra chuyển trạng thái :

1. thực hiện I/O tốc độ chậm
2. Chạy hết khe thời gian
3. Được chọn để chạy khe thời gian kế.
4. I/O sẵn sàng phục vụ.



Trạng thái vi mô của process

Process management	Memory management	File management
Registers	Pointer to text segment	UMASK mask
Program counter	Pointer to data segment	Root directory
Program status word	Pointer to bss segment	Working directory
Stack pointer	Exit status	File descriptors
Process state	Signal status	Effective uid
Time when process started	Process id	Effective gid
CPU time used	Parent process	System call parameters
Children's CPU time	Process group	Various flag bits
Time of next alarm	Real uid	
Message queue pointers	Effective	
Pending signal bits	Real gid	
Process id	Effective gid	
Various flag bits	Bit maps for signals	
	Various flag bits	

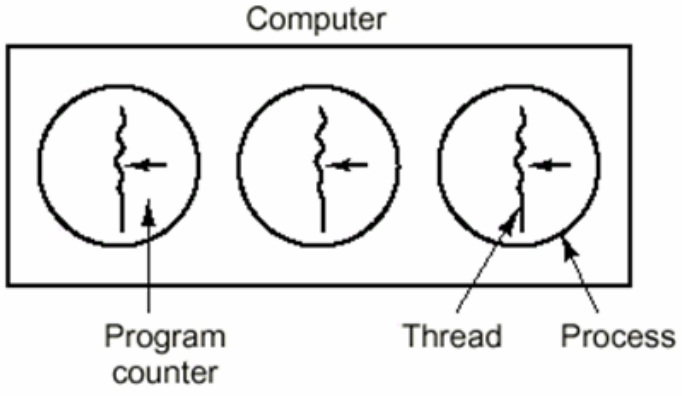
Trạng thái vi mô của process

- Để quản lý trạng thái của từng process, HĐH sẽ dùng 1 record dữ liệu gồm nhiều field, mỗi field chứa 1 thông tin trạng thái mà HĐH muốn quản lý.
- Một field đặc biệt trong record quản lý là field chứa mã trạng thái vĩ mô của process tương ứng để HĐH biết process đang chạy hay đang Ready | Blocked.
- Các record quản lý của các process sẽ được hợp thành 1 bảng quản lý : bảng này rất quan trọng của HĐH
- Bảng quản lý trạng thái thường được hiện thực bằng danh sách liên kết để việc thêm/bớt từng record dễ dàng và hiệu quả (vì tần suất tạo/xóa process rất cao theo thời gian).

2.4 Khái niệm Thread

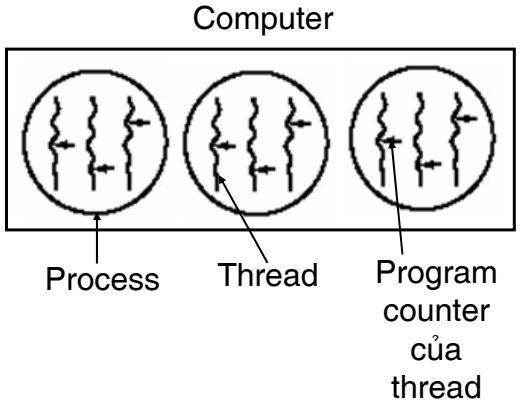
- Process tuần tự chỉ chứa 1 luồng thi hành lệnh (Thread) cho 1 chương trình từ điểm nhập đến điểm kết thúc.

Hình bên miêu tả 1 máy tính đang chạy 3 chương trình đồng thời, mỗi chương trình được viết bằng giải thuật tuần tự, nghĩa là chỉ chứa 1 thread.



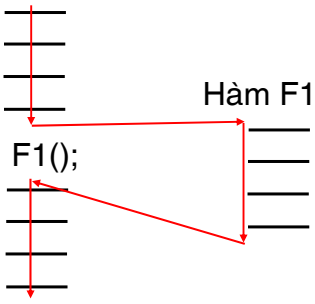
Process multi-thread

Hình bên miêu tả 1 máy tính đang chạy 3 chương trình đồng thời, mỗi chương trình được viết bằng giải thuật song song gồm 3 tác vụ chạy đồng thời, mỗi tác vụ tương ứng với 1 thread độc lập. Tuy nhiên các thread của cùng 1 process đều truy xuất đến không gian làm việc của process => có thể gây ra tranh chấp và làm hư hỏng tài nguyên bị tranh chấp.

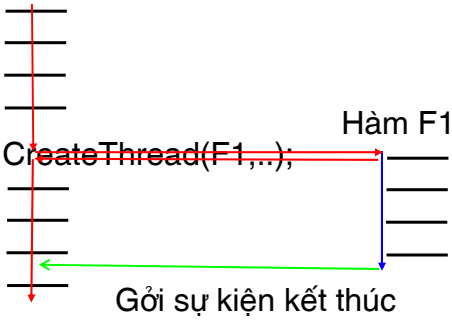


Khác biệt giữa gọi hàm và tạo thread mới

Process mono-thread (chỉ chứa 1 luồng màu đỏ)

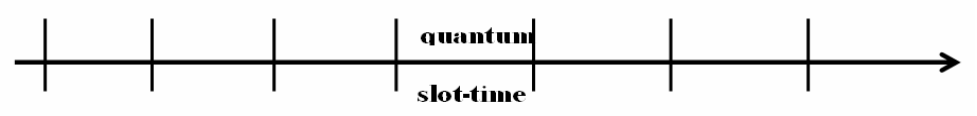


Process multi-thread (chứa 1 luồng chính màu đỏ và 1 luồng con màu xanh)



2.5 Lập lịch cho các process chạy

- Dùng cơ chế phân chia thời gian (time-sharing) để chạy các process đồng thời trên 1 CPU.



- Dùng timer để tính khe thời gian. Mỗi lần hết khe hay process hiện hành chờ I/O, trình lập lịch sẽ thực hiện công việc "chuyển ngữ cảnh process" để dừng process hiện hành và cho phép process khác chạy.

Lập lịch cho các process chạy

Công việc "chuyển ngữ cảnh process" để dừng process hiện hành và cho phép process khác chạy gồm các bước chính yếu sau đây :

1. Dừng process hiện hành (P1) lại.
2. Cất trạng thái của P1 vào record quản lý tương ứng để dùng sau.
3. Chọn 1 process (P2) trong bảng process để chạy tiếp (theo các tiêu chí nào đó).
4. Phục hồi trạng thái của P2 từ record quản lý.
5. Kích hoạt timer đếm khe thời gian.
6. Cho P2 tiếp tục chạy từ vị trí ngừng trước đây.

Công việc "chuyển ngữ cảnh process" là xác định, CPU có tốc độ càng cao thì thực hiện chuyển ngữ cảnh càng nhanh (càng tốt ít thời gian).

Lập lịch cho các process chạy

Độ lớn của khe thời gian được xác định bằng cách dùng hòa giữa 2 yêu cầu mâu thuẫn nhau :

1. Sao cho các ứng dụng chạy hiệu quả nhất : để được yêu cầu này, ta phải chọn khe càng lớn càng tốt để tỉ lệ thời gian process chạy thực sự/thời gian chuyển ngữ cảnh trong từng khe là lớn nhất.
2. Sao cho ứng dụng tương tác nhanh với người dùng để lừa họ rằng ứng dụng luôn luôn chạy (chứ không phải chạy cà giựt) : để được yêu cầu này, ta phải chọn khe càng nhỏ càng tốt để khoảng cách thời gian giữa 2 lần chạy liên tiếp của process là nhỏ nhất.

Hiện nay, khe thời gian thường ở mức vài ms để 1 process có thể chạy nhiều lần (>20) trong 1 giây.

2.6 Các phương pháp lập lịch cho các process

Trong việc "chuyển ngữ cảnh process", bước chọn 1 process trong bảng process để chạy tiếp cần được nói rõ thêm. Tùy theo các tiêu chí cần đạt được mà ta sẽ chọn phương pháp nào. Hiện người ta đã đề nghị nhiều phương pháp, nhưng mỗi phương pháp chỉ đáp ứng 1 vài tiêu chí cú thể. Sau đây chúng ta sẽ nghiên cứu 4 phương pháp điển hình :

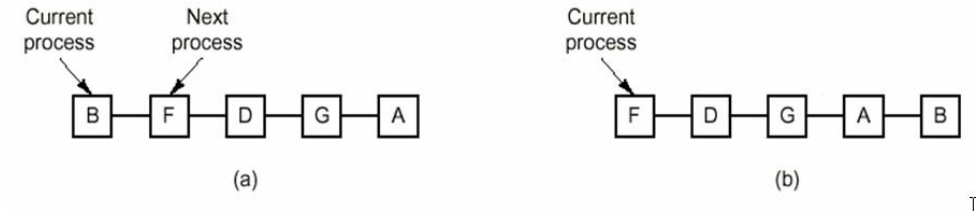
1. Phương pháp Round-robin
2. Phương pháp dựa vào quyền ưu tiên
3. Phương pháp dùng nhiều hàng chờ quyền ưu tiên
4. Phương pháp cho process ngắn chạy trước.

Phương pháp Round-robin

Tiêu chí đạt được : xử lý các process theo tinh thần bình đẳng.

Ý tưởng :

- Dùng 1 hàng chờ trật tự (danh sách liên kết), process nào đến yêu cầu chạy trước được nằm trước, process nào yêu cầu sau phải nằm sau.
- Khi hết khe thời gian mà vẫn cần chạy tiếp (thường là vậy), process phải quay về đuôi hàng chờ để được giải quyết cho lần chạy sau.



Phương pháp Round-robin

- Tùy góc nhìn mà tiêu chí bình đẳng có đạt được không ?
- Theo góc nhìn toán học và lý thuyết thì phương pháp Round-robin đạt được tiêu chí bình đẳng hầu như tuyệt đối.
- Tuy nhiên trong thực tế, vì yêu cầu dùng CPU để chạy giải thuật của từng process rất khác nhau, có process rất phức tạp chạy nhiều ngày mới xong, có process rất đơn giản chỉ cần vài giây, thậm chí vài ms là xong rồi nên phương pháp Round-robin trở nên rất bất bình đẳng thực tế.

Phương pháp dựa vào quyền ưu tiên

Tiêu chí đạt được : xử lý các process theo quyền ưu tiên của từng process : process có quyền cao thì được chạy nhiều hơn process có độ ưu tiên thấp.

Ý tưởng :

- Vị trí chờ của process trong hàng chờ không quan trọng,
- Mỗi process có chỉ số quyền ưu tiên xác định (lấy từ quyền ưu tiên của user kích hoạt process).
- Khi cần chọn process chạy tiếp, hệ thống sẽ chọn process có độ ưu tiên cao nhất trong hàng chờ, bất chấp nó đến trước hay sau.
- Khi hết khe thời gian mà vẫn cần chạy tiếp (thường là vậy), process phải quay về hàng chờ để được giải quyết cho lần chạy sau.

Phương pháp dựa vào quyền ưu tiên

- Khuyết điểm chính của phương pháp này là rất dễ dẫn đến tình trạng "tắt nghẽn có chọn lọc" : process có quyền ưu tiên thấp sẽ phải chờ rất lâu, thậm chí là không thời hạn vì luôn có process quyền ưu tiên cao hơn mình nằm trong hàng chờ (trình trạng này dễ xảy ra).

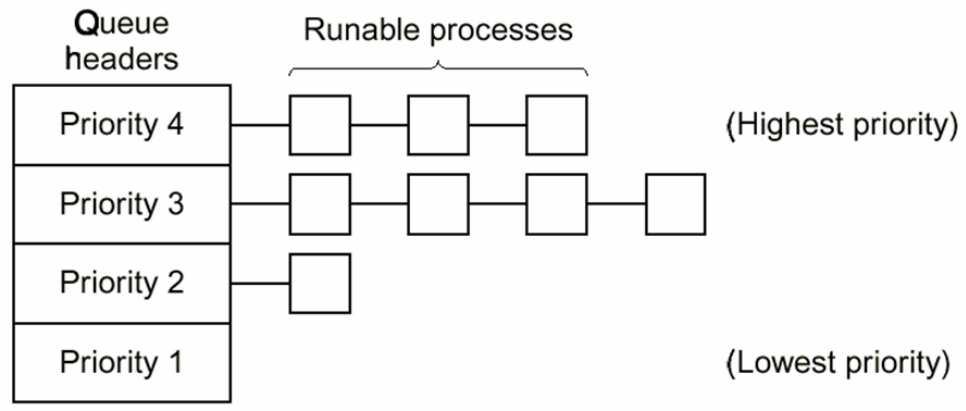
Phương pháp dùng nhiều hàng chờ

Tiêu chí đạt được : dung hòa giữa 2 phương pháp Round-robin và dựa vào quyền ưu tiên để khắc phục các khuyết điểm của chúng.

Ý tưởng :

- Tạo nhiều hàng chờ khác nhau, mỗi hàng dành cho các process cùng quyền ưu tiên tương ứng. Trong hàng, process nào đến trước được xử lý trước.
- Khi cần chọn 1 process chạy tiếp, hệ thống sẽ dò tuần tự từng hàng chờ theo thứ tự quyền ưu tiên từ cao xuống thấp, hàng chờ nào có process thì chọn process ở đầu hàng đề chạy tiếp.
- Khi hết khe thời gian mà vẫn cần chạy tiếp (thường là vậy), process phải quay về hàng chờ để được giải quyết cho lần chạy sau.

Phương pháp dùng nhiều hàng chờ



Phương pháp dùng nhiều hàng chờ

Ý tưởng (tt) :

- Quyền ưu tiên của process được thiết lập lần đầu dựa vào quyền của user kích hoạt nó. Theo thời gian chạy, tùy tính chất chạy của process mà hệ thống sẽ hiệu chỉnh lại cho phù hợp.
- Process sẽ được tăng quyền ưu tiên khi (1) chờ N khe thời gian mà chưa chạy được, (2) dùng CPU rất hiệu quả chứ không chờ I/O, (3)…
- Process sẽ bị giảm quyền ưu tiên khi (1) chạy được N khe thời gian rồi, (2) dùng CPU rất kém hiệu quả vì phải chờ I/O quá nhiều, (3)…

Phương pháp cho process ngắn chạy trước

Tiêu chí đạt được : cho người dùng cảm thấy hệ thống chạy hiệu quả nhất.

Ý tưởng :

- Theo góc nhìn người dùng, họ nói hệ thống chạy càng hiệu quả khi thấy thời gian đáp ứng cho phần mềm của họ nhanh nhất.
- Thời gian đáp ứng phần mềm của hệ thống là khoảng thời gian từ lúc phần mềm được kích hoạt chạy đến lúc nó hoàn thành nhiệm vụ.
- Thời gian đáp ứng cho từng phần mềm riêng lẻ không có độ tin cậy cao, do đó người ta dùng thời gian đáp ứng trung bình của hệ thống cho n phần mềm trong khoảng thời gian khảo sát làm thước đo độ hiệu quả của hệ thống.

Phương pháp cho process ngắn chạy trước

Ý tưởng (tt) :

- Để hệ thống đáp ứng tốt nhất, ta nên chọn process ngắn nhất chạy trước rồi cứ thế đến process dài nhất sau cùng.
- Thí dụ tại thời điểm t0, có 4 process sau cần chạy : P1 chạy tốn 70s, P2 chạy tốn 10s, P3 chạy tốn 2s, P4 chạy tốn 1s.
- Nếu ta lập lịch cho P1 chạy trước rồi mới tới P2, rồi P3 rồi P4 thì thời gian đáp ứng trung bình cho 4 phần mềm trên là : (70+80+82+83)/4 = 78.75s.
- Còn nếu ta lập lịch theo phương pháp cho process ngắn nhất chạy trước thì thứ tự chạy sẽ là P4 $\rightarrow$ P3 $\rightarrow$ P2 $\rightarrow$ P1 và thời gian đáp ứng trung bình cho 4 phần mềm trên là : (1+3+13+83)/4 = 25s.