

Khái niệm tiến trình

- Một Hệ điều hành thực hiện nhiều chương trình:
 - + Hệ thống xử lý theo lô: công việc (job)
 - + Hệ thống chia sẻ thời gian: tác vụ (task)
- Ở đây chúng ta dùng tiến trình và công việc với cùng ý nghĩa

Tiến trình (Process)

- Chương trình đang được thực hiện, bao gồm
- + Phần văn bản (string)
- + Ngăn xếp (bộ nhớ vào trước ra sau)
- + Phần dữ liệu
- + Giá trị bộ đếm chương trình (PC: Program counter), thanh ghi
- CPU xử lý tiến trình tuần tự
- Là thực thể hoạt động:
- + đối lập với chương trình

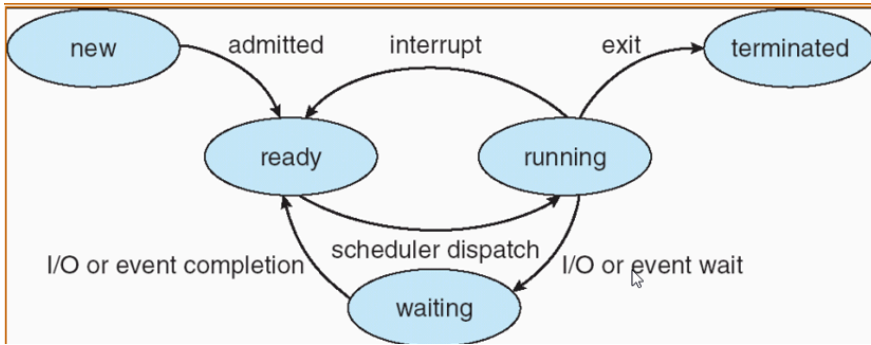
Cấu trúc bộ nhớ tiến trình



- Text Section: Xâu văn bản (ví dụ: Hà Nội)
- Data Section: Khi khai báo biến thì Data section sẽ biến thành nội dung cụ thể

Trạng thái tiến trình

- Tiến trình thay đổi trạng thái trong khi thực hiện
- + New → Ready → Running → Waiting → Terminated
- Tại một thời điểm chỉ có một tiến trình ở trạng thái running



- Một tiến trình được chấp nhận (admitted) sẽ được chuyển sang trạng thái “ready”
 - Khi đang ở trạng thái “ready”, nếu tiến trình nhận lệnh từ CPU sẽ chuyển sang trạng thái “running”
 - Một tiến trình đang ở “running” chuyển sang trạng thái ready khi có một tiến trình khác đến sau nhưng có độ ưu tiên lớn hơn
- Nếu đang ở “running: mà lại có yêu cầu vào ra từ các thiết bị vào ra, tiến trình sẽ chuyển sang trạng thái “waiting”, và khi việc vào/ ra hoàn thành, tiến trình sẽ quay về trạng thái “ready”

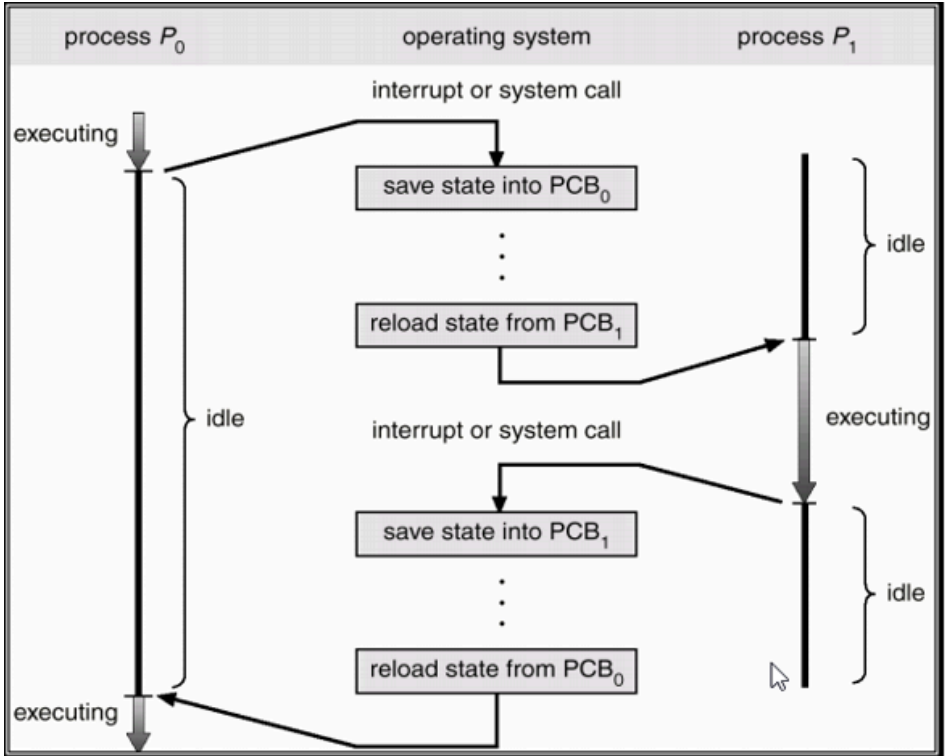
Khối điều khiển tiến trình (Process Control Block)

pointer	process state
process number	
program counter	
registers	
memory limits	
list of open files	
⋮	

- Process state: lưu các trạng thái của chương trình (New - 0, Ready - 1, Running - 2, Waiting - 3, Terminated - 4), trạng thái Terminated-4 là trạng thái không cần ghi vào Process state (vì khi chương trình bị hủy rồi thì khối tiến trình sẽ bị thu lại nên không cần ghi lại trạng thái này)

- Memory limits:

Chuyển đổi CPU giữa các tiến trình



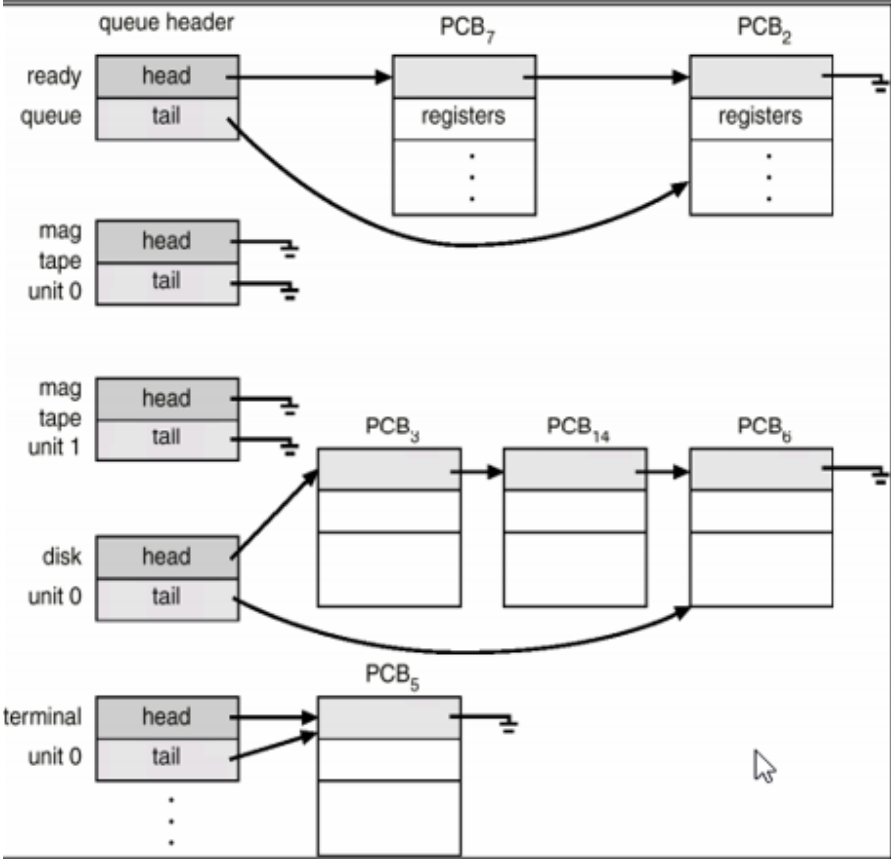
- System call: là lời gọi từ chương trình (thường là yêu cầu vào / ra dữ liệu)

Lập lịch tiến trình

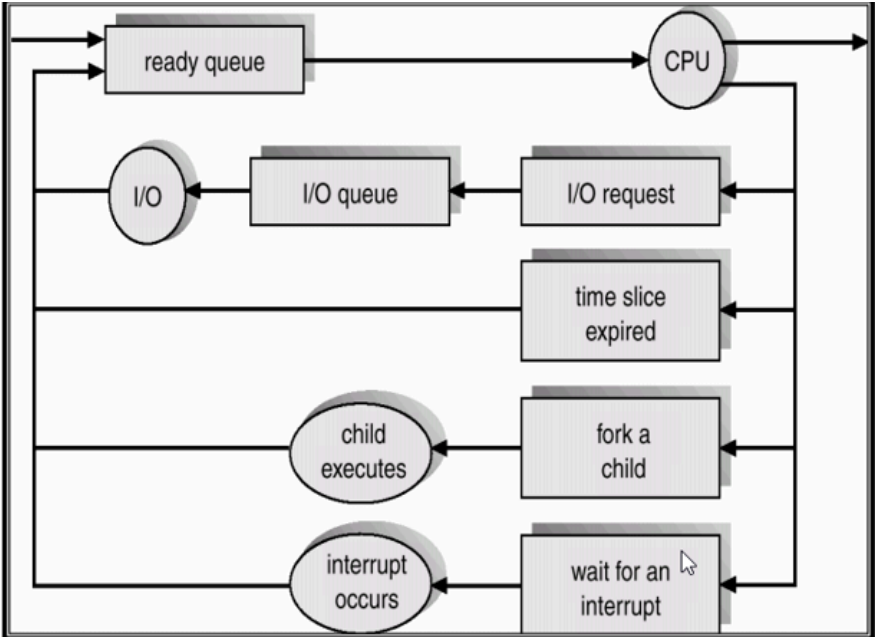
- Mục đích của đa chương trình: Tăng tính tận dụng CPU
- Mục đích của phân chia thời gian: Người dùng có thể tương tác với tiến trình trong lúc nó đang thực thi
- Bài toán xử lý nhiều tiến trình một lúc

Các hàng đợi lập lịch tiến trình

- Hàng đợi công việc: Một tập các tiến trình trong hệ thống
- Hàng đợi sẵn sàng: Tập các tiến trình trong bộ nhớ trong, sẵn sàng và chỉ chờ thực hiện
- Hàng đợi thiết bị: Tập các tiến trình chờ một thiết bị vào /ra
- Hàng đợi công việc lớn nhất, sau đó đến hàng đợi sẵn sàng, nhỏ nhất là hàng đợi thiết bị
- Các tiến trình di trú từ hàng đợi này đến hàng đợi khác
- Hàng đợi sẵn sàng và các hàng đợi thiết bị khác nhau



Biểu diễn việc lập lịch tiến trình-Biểu đồ hàng đợi



Vòng đời của một tiến trình

- Khởi tạo: hàng đợi sẵn sàng
- Các sự kiện có thể xảy ra khi tiến trình đã được gán CPU
 - + Sinh ra một yêu cầu I/O, đi vào hàng đợi I/O
 - + Tạo ra một tiến trình con và đợi cho nó kết thúc
 - + Bị tước quyền sử dụng CPU
- Tiếp tục vòng lặp đến khi kết thúc
 - + Bị xóa khỏi tất cả các hàng đợi
 - + PCB và tất cả các tài nguyên bị thu hồi

Các bộ lập lịch

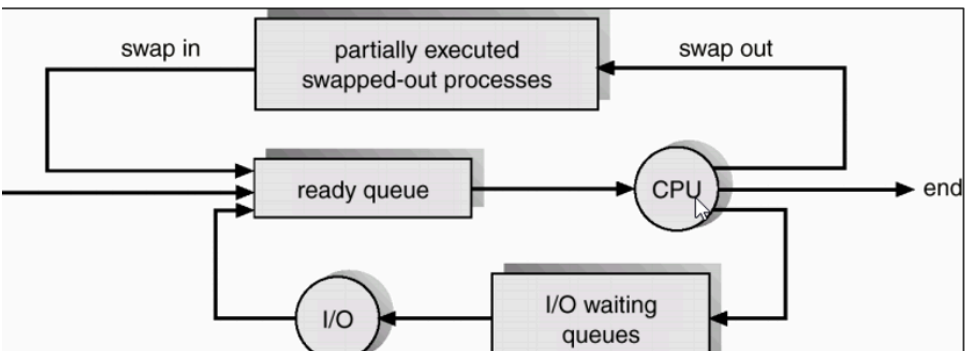
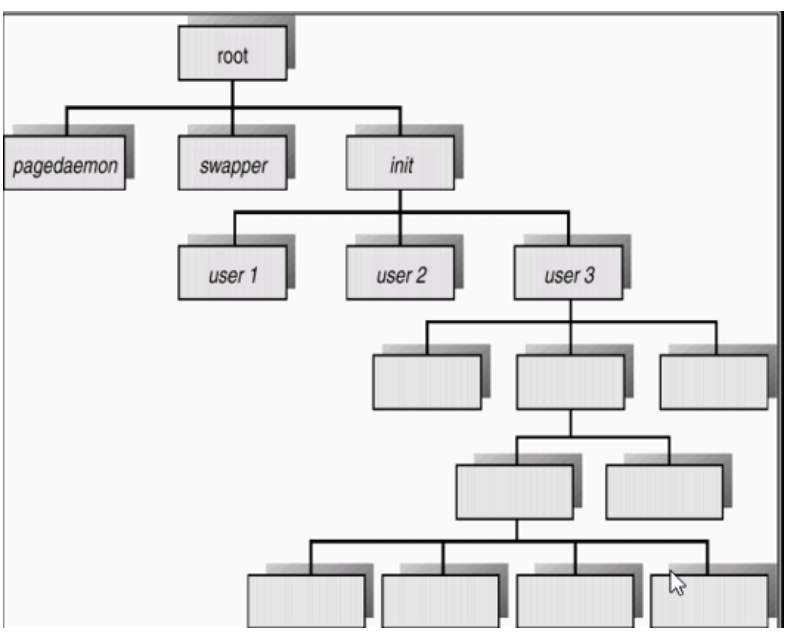
- Tiến trình lưu trú trong nhiều loại hàng đợi
- + Các bộ lập lịch chọn các tiến trình từ các hàng đợi
- Bộ lập lịch dài hạn
 - + Lập lịch công việc – job scheduler
 - + Chọn các tiến trình trong tập tiến trình và tải nó vào bộ nhớ để thực hiện.
- Bộ lập lịch ngắn hạn (lập lịch CPU)
 - + Chọn trong số các tiến trình trong hàng đợi sẵn sàng để thực hiện

Bộ lập lịch ngắn hạn vs. Dài hạn

- Tần số thực thi:
 - + Ngắn hạn: Thường xuyên, Đòi hỏi thực thi nhanh
 - + Dài hạn: Không thường xuyên bằng, Thể hiện mức độ “đa chương trình”

Bộ lập lịch dài hạn:

- Hai loại tiến trình:
 - + Giới hạn I/O
 - + Giới hạn CPU: cho bạn khả năng thiết lập 1 giới hạn cố định trên CPU [resource entitlement](#) cho công việc trên 1 CSDL hoặc lớp dịch vụ (service class)
- Chọn một kết hợp tốt các tiến trình giới hạn vào /ra và các tiến trình giới hạn CPU.
- Một số hệ thống phân chia thời gian không có bộ lập lịch dài hạn (Unix)
- Bộ lập lịch trung hạn
 - + Sử dụng trong một số HĐH phân chia thời gian

	
<p>Chuyển đổi ngữ cảnh...</p> <ul style="list-style-type: none"> - Chuyển đổi ngữ cảnh là thay đổi trạng thái (ví dụ, các học sinh chuyển phòng học sang phòng khác để học) - Chuyển đổi CPU cho một tiến trình khác - Ngữ cảnh tiến trình - Hoạt động chuyển đổi ngữ cảnh - Kernel lưu lại ngữ cảnh của tiến trình cũ trong PCB và tải ngữ cảnh được lưu của tiến trình mới được lập lịch <ul style="list-style-type: none"> - Thời gian chuyển đổi ngữ cảnh: lãng phí + Phụ thuộc vào máy, thông thường từ 1 đến 1000 micro giây + Phụ thuộc vào sự hỗ trợ của phần cứng + Các kĩ thuật quản lý bộ nhớ + Bottleneck sử dụng các cấu trúc mới như thread để tránh nút cổ chai này 	
<p>Các thao tác trên tiến trình</p> <p>Tạo tiến trình</p>	
<ul style="list-style-type: none"> - Một tiến trình có thể tạo ra nhiều tiến trình con, qua lời gọi hệ thống <code>create_process</code> + Tiến trình cha + Tiến trình con <ul style="list-style-type: none"> - Chia sẻ tài nguyên + Tất cả các tài nguyên + Một phần tài nguyên - Cũng có thể không chia sẻ tài nguyên - Thực thi + Thực thi đồng thời + Thực thi tuần tự 	<p>Cây tiến trình</p>  <ul style="list-style-type: none"> - Root là người dùng - Swapper: chương trình quản lý bộ nhớ ảo
<p>Tạo tiến trình trong Unix</p> <ul style="list-style-type: none"> - Mỗi tiến trình có một ID (PID) - Gọi lời gọi hệ thống <code>fork()</code> để tạo tiến trình mới - Tiến trình cha có thể đợi hoặc thực hiện đồng thời với tiến trình con - Không gian địa chỉ của tiến trình con là một bản sao của không gian địa chỉ tiến trình cha - Mã trả về từ <code>fork()</code> - Tiến trình con có thể gọi lời gọi hệ thống <code>execp()</code> để tải một chương trình mới vào thực hiện 	
<p>Hủy tiến trình</p>	
<ul style="list-style-type: none"> - Tiến trình thực thi xong + Hệ điều hành thực hiện lệnh <code>exit</code> + (có thể) trả về dữ liệu cho tiến trình cha + Hủy các tài nguyên đã được phân phối cho tiến trình - Tiến trình bị hủy bởi một tiến trình khác (tiến trình cha) + Tiến trình cha cần biết chỉ số của tiến trình con <ul style="list-style-type: none"> - Tiến trình cha dừng sự hoạt động của tiến trình con vì + Tiến trình con dùng quá tài nguyên cho phép 	<p>Hợp tác giữa các tiến trình</p> <ul style="list-style-type: none"> - Các tiến trình cộng tác + Tiến trình độc lập: Không bị ảnh hưởng bởi tiến trình khác, Không chia sẻ dữ liệu + Tiến trình hợp tác: Bị ảnh hưởng bởi tiến trình khác, Dùng chung dữ liệu - Cần các kĩ thuật giao tiếp/ đồng bộ tiến trình <p>Vì sao hợp tác tiến trình?</p> <ul style="list-style-type: none"> - Chia sẻ thông tin: Đồng thời truy cập đến tài nguyên chia sẻ - Tăng tốc độ tính toán

<ul style="list-style-type: none"> + Nhiệm vụ của tiến trình con không còn quan trọng + Tiến trình cha thoát và Hệ điều hành thực thi cơ chế “hủy theo dây chuyền” (cascading termination) - Không thực hiện cơ chế hủy theo dây chuyền, tiến trình init trở thành tiến trình cha 	<ul style="list-style-type: none"> + Chia thành các bài toán con, thực thi song song + Chỉ có được trong các hệ thống có nhiều thành phần xử lý (đa CPU, đa kênh vào /ra) - Tính module hóa: Chia nhỏ các chức năng - Tiện dụng: Có thể thực hiện nhiều nhiệm vụ tại một thời điểm
Bài toán “Producer - Consumer” <ul style="list-style-type: none"> - Nhà sản xuất (producer): Sinh sản phẩm (thông tin/hàng hoá) - Người tiêu dùng (consumer): Dùng thông tin/hàng hoá do Nhà sản xuất tạo ra - Bộ đệm chung + Không giới hạn + Giới hạn + Hỗ trợ bởi hệ điều hành (thông qua IPC- Inter Process Communication) + Do lập trình viên tạo ra bằng cách sử dụng bộ nhớ chia sẻ 	

Lập lịch CPU

- Lập lịch trong chỉ trong các trường hợp 1 và 4 gọi là lập lịch không chiếm đoạt (non-preemptive)
--

Các thuật toán lập lịch

First Come First Serve <ul style="list-style-type: none"> - Dùng trong việc thanh toán ở các cửa hàng (ai đến trước thanh toán trước) 	Shortest Job First
	Round Robin
<ul style="list-style-type: none"> - Cách làm các thuật toán ở trong phần bài tập 	

--

--

--