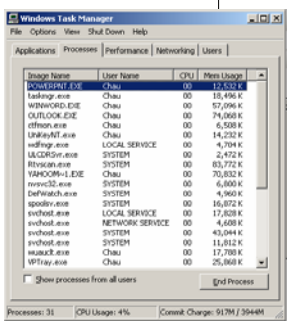
Nguyên lý hệ điều hành

Nguyễn Hải Châu
Khoa Công nghệ thông tin
Trường Đại học Công nghệ

Khái niệm tiến trình

Tiến trình là gì?

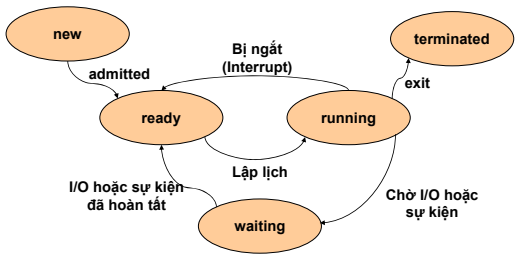
- Thuật ngữ: Process (tiến trình/quá trình)
- Là một *chương trình đang được thực hiện*
- Được xem là đơn vị làm việc trong các HĐH
- Có hai loại tiến trình:
  - Tiến trình của HĐH
  - Tiến trình của NSD



Tiến trình gồm có…

- Đoạn mã lệnh (code, có sách gọi là text)
- Đoạn dữ liệu
- Đoạn ngăn xếp và heap (stack/heap)
- Các *hoạt động hiện tại* được thể hiện qua con đếm lệnh (IP) và nội dung các thanh ghi (registers) của bộ xử lý
- Chú ý:
  - Tiến trình là *thực thể chủ động*
  - Chương trình là *thực thể bị động*

Trạng thái tiến trình



Khối điều khiển tiến trình

- **Thuật ngữ: Process Control Block (PCB)**
- **Các thông tin:**
  - **Trạng thái tiến trình**
  - **Con đếm**
  - **Các thanh ghi**
  - **Thông tin về lập lịch**
  - **Thông tin về bộ nhớ**
  - **Thông tin accounting**
  - **Thông tin vào/ra**

Con trỏ	Trạng thái tiến trình
Số hiệu tiến trình (Process number)	
Con đếm (program counter)	
Các thanh ghi (registers)	
Giới hạn bộ nhớ	
Danh sách các tệp đang mở	
.....	

Lập lịch tiến trình

7

Tại sao phải lập lịch?

- Số lượng NSD, số lượng tiến trình luôn lớn hơn số lượng CPU của máy tính rất nhiều
- Tại một thời điểm, chỉ có duy nhất một tiến trình được thực hiện trên một CPU
- Vấn đề:
  - Số lượng yêu cầu sử dụng nhiều hơn số lượng tài nguyên đang có (CPU)
  - Do đó cần lập lịch để phân phối thời gian sử dụng CPU cho các tiến trình của NSD và hệ thống

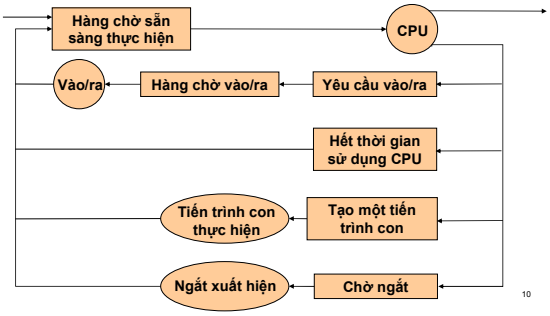
8

Hàng chờ lập lịch

- Thuật ngữ: Queue
- Các tiến trình chưa được phân phối sử dụng CPU sẽ được đưa vào *hàng chờ* (*queue*)
- Có thể có nhiều hàng chờ trong hệ thống: Hàng chờ sử dụng CPU, hàng chờ sử dụng máy in, hàng chờ sử dụng ổ đĩa CD…
- Trong suốt thời gian tồn tại, tiến trình phải di chuyển giữa các hàng chờ

9

Hàng chờ lập lịch tiến trình



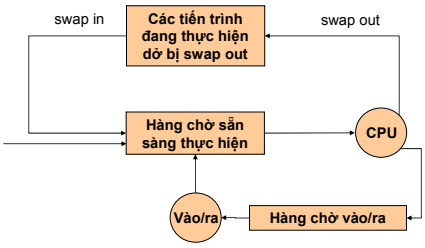
10

Phân loại các bộ lập lịch

- *Bộ lập lịch dài hạn* (long-term scheduler)
  - Thường dùng trong các hệ xử lý theo lô
  - Đưa tiến trình từ spool vào bộ nhớ trong
- *Bộ lập lịch ngắn hạn* (short-term scheduler)
  - Còn gọi là bộ lập lịch CPU
  - Lựa chọn tiến trình tiếp theo được sử dụng CPU
- *Bộ lập lịch trung hạn* (medium-term scheduler)
  - Hay còn gọi là swapping (tráo đổi)
  - Di chuyển tiến trình đang trong trạng thái chờ giữa bộ nhớ trong và bộ nhớ ngoài

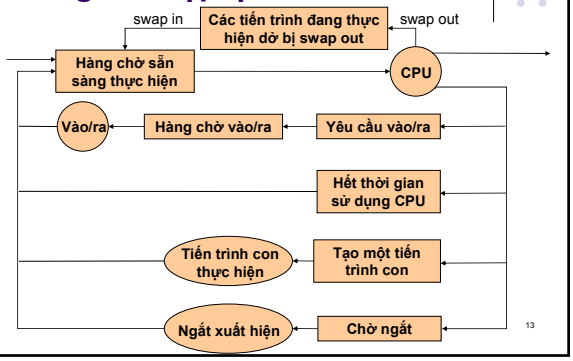
11

Minh họa bộ lập lịch trung hạn



12

Hàng chờ lập lịch tiến trình



13

Chuyển trạng thái

- Thuật ngữ: Context switch
- Xảy ra khi một tiến trình A bị ngắt ra khỏi CPU, tiến trình B bắt đầu được sử dụng CPU
- Cách thực hiện:
  - Nhân HĐH ghi lại toàn bộ trạng thái của A, lấy từ PCB (khối điều khiển tiến trình) của A
  - Đưa A vào hàng chờ
  - Nhân HĐH nạp trạng thái của B lấy từ PCB của B
  - Thực hiện B

14

Chuyển trạng thái

- Việc chuyển trạng thái, nói chung, là lãng phí thời gian của CPU
- Do đó việc chuyển trạng thái cần được thực hiện càng nhanh càng tốt
- Thông thường thời gian chuyển trạng thái mất khoảng 1-1000 micro giây

15

Các thao tác với tiến trình

16

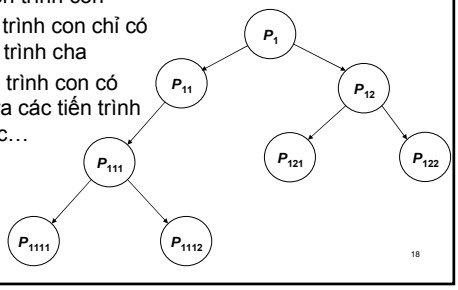
Tạo tiến trình

- HĐH cung cấp hàm **create-process** để tạo một tiến trình mới
  - Tiến trình gọi đến hàm **create-process** là tiến trình cha (parent process)
  - Tiến trình được tạo ra sau khi thực hiện hàm **create-process** là tiến trình con (child process)
- Sau khi tiến trình con được tạo, tiến trình cha có thể:
  - Chờ tiến trình con kết thúc rồi tiếp tục thực hiện
  - Thực hiện "song song" với tiến trình con

17

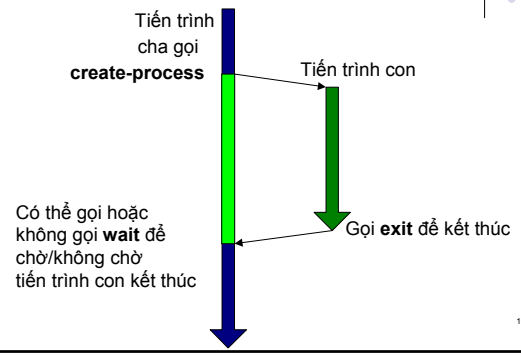
Cây tiến trình

- Tiến trình cha có thể có nhiều tiến trình con
- Mỗi tiến trình con chỉ có một tiến trình cha
- Các tiến trình con có thể tạo ra các tiến trình con khác…



18

Minh họa tiến trình cha và con



19

Kết thúc tiến trình

- Một tiến trình kết thúc khi:
  - Thực hiện xong và gọi hàm hệ thống **exit** (kết thúc bình thường)
  - Gọi đến hàm **abort** hoặc **kill** (kết thúc bất thường khi có lỗi hoặc có sự kiện)
  - Bị hệ thống hoặc tiến trình cha áp dụng hàm **abort** hoặc **kill** do:
    - Sử dụng quá quota tài nguyên
    - Tiến trình con không còn cần thiết
    - Khi tiến trình cha đã kết thúc (trong một số HĐH)

20

Minh họa tiến trình trong UNIX

```
#include <stdio.h>
main()
{
    int pid=fork(); /* Tạo tiến trình mới bằng hàm fork() */
    if (pid<0) { perror("Cannot create process"); return(-1); }
    else if (pid==0) { /* Tiến trình con */
        execlp();
    }
    else { /* Tiến trình cha */
        wait(NULL);  /* Nếu không có lệnh này tiến trình cha thực hiện
                        "song song" với tiến trình con */
        printf("Child completed\n");
        return(0);
    }
}
```

21

Hợp tác giữa các tiến trình

- Các tiến trình có thể hoạt động *độc lập* hoặc *hợp tác* với nhau
- Các tiến trình cần hợp tác khi:
  - Sử dụng chung thông tin
  - Thực hiện một số nhiệm vụ chung
  - Tăng tốc độ tính toán
  - …
- Để hợp tác các tiến trình, cần có các cơ chế truyền thông/liên lạc giữa các tiến trình (Interprocess communication – IPC)

22

Truyền thông giữa các tiến trình (IPC)

23

Các hệ thống truyền thông điệp

- Cho phép các tiến trình truyền thông với nhau qua các toán tử **send** và **receive**
- Các tiến trình cần có *tên* để tham chiếu
- Cần có một kết nối (*logic)* giữa tiến trình *P* và *Q* để truyền thông điệp
- Một số loại truyền thông:
  - Trực tiếp hoặc gián tiếp
  - Đối xứng hoặc không đối xứng
  - Sử dụng vùng đệm tự động hoặc không tự động

24

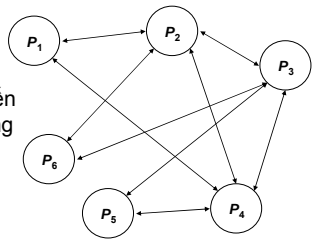
Truyền thông trực tiếp

- Hai toán tử
  - **send**(*P*, *msg*): Gửi *msg* đến tiến trình *P*
  - **receive**(*Q*, *msg*): Nhận *msg* từ tiến trình *Q*
- Cải tiến:
  - **send**(*P*, *msg*): Gửi *msg* đến tiến trình *P*
  - **receive**(*id*, *msg*): Nhận *msg* từ bất kỳ tiến trình nào

25

Minh họa truyền thông trực tiếp

- Mỗi kết nối được thiết lập cho một cặp tiến trình duy nhất
- Mỗi tiến trình chỉ cần biết tên/số hiệu của tiến trình kia là truyền thông được
- Tồn tại duy nhất một kết nối giữa một cặp tiến trình



26

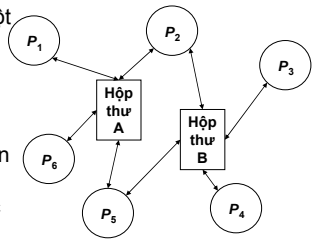
Truyền thông gián tiếp

- Các thông điệp được gửi và nhận qua các *hộp thư (*mailbox) hoặc qua các *cổng* (port)
- Hai toán tử:
  - **send**(*A*, *msg*): Gửi *msg* đến hộp thư *A*
  - **receive**(*B*, *msg*): Nhận *msg* từ hộp thư *B*
- Minh họa: Topo mạng hình sao

27

Minh họa truyền thông gián tiếp

- Hai tiến trình có kết nối nếu sử dụng chung một hộp thư
- Một kết nối có thể sử dụng cho nhiều tiến trình (>=2)
- Nhiều kết nối có thể tồn tại giữa một cặp tiến trình (nếu sử dụng các hộp thư khác nhau)



28

Vấn đề đồng bộ hóa

- Thuật ngữ: Synchronization
- Liên quan tới phương thức cài đặt các toán tử **send** và **receive**:
  - Phương thức có chờ (blocking)
  - Phương thức không chờ (non-blocking)

29

Các phương thức send/receive

	send(*P*, *msg*)	**receive**(*Q*, *msg*)
Blocking	Tiến trình truyền thông điệp chờ đến khi *msg* được nhận hoặc *msg* được phân phát đến hộp thư	Tiến trình nhận tạm dừng thực hiện cho đến khi *msg* được chuyển tới
Non-blocking	Tiến trình truyền không phải chờ *msg* đến đích để tiếp tục thực hiện	Tiến trình nhận trả lại kết quả là *msg* (nếu nhận được) hoặc báo lỗi (nếu chưa nhận được)

30

Vấn đề sử dụng vùng đệm

- Các thông điệp nằm trong hàng chờ tạm thời
- Cỡ của hàng chờ:
  - Chứa được 0 thông điệp: **send** blocking
  - Chứa được *n* thông điệp: **send** non-blocking cho đến khi hàng chờ có *n* thông điệp, sau đó **send** blocking
  - Vô hạn: **send** non-blocking

31

Luồng (thread)

- Sinh viên tự tìm hiểu trong giáo trình trang

32

Tóm tắt

- Khái niệm tiến trình
- Các trạng thái, chuyển trạng thái tiến trình
- Khối điều khiển tiến trình
- Lập lịch tiến trình, các loại bộ lập lịch
- Truyền thông giữa các tiến trình
  - Gián tiếp, trực tiếp
  - Blocking và non-blocking (đồng bộ hóa)
  - Vấn đề sử dụng vùng đệm (buffer)

33

Bài tập

- Viết chương trình C trong Linux/Unix tạo ra 16 tiến trình con. Tiến trình cha chờ cho 16 tiến trình con này kết thúc rồi mới kết thúc bằng hàm **exit.** Sử dụng các hàm **fork** và **wait** để thực hiện yêu cầu.
- Hãy tìm một số ví dụ thực tế minh họa cho các khái niệm lập lịch/hàng chờ trong tình huống có nhiều người sử dụng và ít tài nguyên.

34

Bài tập

- Hãy viết chương trình minh họa cho các cơ chế truyền thông non-blocking, blocking
- Hãy viết chương trình minh họa các cơ chế truyền thông điệp sử dụng buffer có độ dài *n* trong hai trường hợp: *n*>0 và *n*=0
- Chú ý: Để làm hai bài tập trên cần sử dụng hai tiến trình; có thể thực hiện bài tập với UNIX/Linux hoặc Windows

35