Input và Output

- Các thiết bị vào ra
- Các tầng phần mềm thiết bị vào/ra
- Thiết bị đĩa
- Thiết bị màn hình cuối

Phân loại thiết bị vào/ra

- Block devices

```
brw-rw----    1 root     floppy     2,   0 Aug 24  2000 fd0
brw-rw----    1 root     disk       3,   0 Aug 24  2000 hda
```

- Character devices

```
crw-rw----    1 root     root      10,   3 Aug 24  2000
   atimouse
crw-r-----    1 root     sys       14,   4 Aug 24  2000 audio
```

- Ranh giới là tương đối
  - Thiết bị băng từ: hoạt động tuần tự nhưng cho phép đọc dữ liệu phân theo khối
- Những ngoại lệ
  - Clock: không phải block cũng như character

Thâm nhập trực tiếp - DMA

- Thông thường: CPU chịu trách nhiệm đọc và chuyển dữ liệu từ thiết bị vào bộ nhớ.
- Direct Memory Access:
  - CPU trao cho bộ điều khiển thiết bị: địa chỉ vùng dữ liệu cần đọc, số lượng và địa chỉ bộ nhớ
  - Bộ điều khiển thiết bị sẽ thực hiện độc lập yêu cầu, không tốn thời gian CPU

Phân tầng phần điều khiển thiết bị

1. Interrupt handlers
2. Device drivers
3. Device-independent OS sofware
4. User level software

Interrupt handlers

- Xử lý trao đổi theo dạng không đồng bộ
- Xu hướng lập trình đồng bộ dễ viết:
  - vd: while (chưa có tín hiệu) do sleep(5);
- Giải pháp: dùng semaphore hoặc dùng phương pháp gửi thông báo. Ví dụ:
  - down() để chờ
  - khi ngắt xảy ra, up() sẽ đánh thức tiến trình

Device drivers

- Nhiệm vụ: nhận các yêu cầu vào/ra mức cao và chuyển thành các lệnh cho bộ điều khiển thiết bị.
- Mã phụ thuộc vào thiết bị
- Tầng này làm trung gian để các tầng trên độc lập khỏi thiết bị. Tầng này biết các thông tin chi tiết điều khiển thiết bị, như với đĩa: tracks, cylinders, đầu từ, tốc độ quay ...
- Interrupt sẽ đánh thức device driver khi dữ liệu đã sẵn sàng

Phần mềm vào/ra độc lập thiết bị

- Cung cấp các dịch vụ:
  - Giao diện chung cho các thiết bị cùng loại
  - Đặt tên thiết bị
  - Bảo vệ thiết bị
  - Cung cấp đơn vị khối dữ liệu độc lập thiết bị
  - Lưu trữ với vùng đệm
  - Phân bổ không gian cho các thiết bị khối
  - Phân và giải phóng các thiết bị giành riêng
  - Báo cáo lỗi

...phần mềm vào/ra độc lập thiết bị

- Giao diện chung:
  - cung cấp một tập các hàm độc lập với 1 thiết bị cụ thể, sử dụng bởi tầng bên trên
- Đặt tên thiết bị:
  - tạo liên kết hình thức giữa các thiết bị với 1 tên
  - I-node chứa 2 trường *major device number* chỉ ra số hiệu thiết bị và *minor device number* chỉ ra đơn vị logic của khối dữ liệu

...phần mềm vào/ra độc lập thiết bị

- Bảo vệ thiết bị
  - bởi vì thiết bị được ánh xạ như 1 tệp, quyền thâm nhập thiết bị được quản lý như đối với tệp
  - quyền thâm nhập: người sở hữu, nhóm sở hữu, những người khác, 3 thuộc tính quyền cho mỗi nhóm: RWX
- Khối dữ liệu logic:
  - cung cấp dịch vụ đọc khối dữ liệu với kích thước logic chung cho các tầng trên, không phụ thuộc vào kích thước cụ thể của thiết bị

...phần mềm vào/ra độc lập thiết bị

- Lưu với vùng đệm:
  - như cache
  - nâng tốc độ vào/ra
- Phân bổ không gian lưu trữ
  - vd: quản lý các khối đĩa rỗi bằng bản đồ bit hay dsmn; thực thi các thuật toán cấp phát
- Phân bổ và giải phóng thiết bị giành riêng:
  - băng từ là thiết bị chỉ dùng bởi 1 tiến trình tại 1 thời điểm
  - quản lý quyền giành sử dụng thiết bị

...phần mềm vào/ra độc lập thiết bị

- Xử lý lỗi:
  - yêu cầu tầng device driver thử lại
  - sau 1 số lần thử lại, thông báo lỗi tới tầng trên

Vào/ra mức người dùng

- Là các hàm thư viện, có thể gọi trực tiếp bởi người dùng
- Mã các hàm được liên kết vào chương trình người dùng
- Phần lớn các hàm làm nhiệm vụ thiết lập tham số và gọi tầng dưới. Trừ một vài hàm đơn giản, vd: printf, gets...
- Ngoại lệ: làm việc với các thiết bị giành riêng phải sử dụng spool và daemon

Thiết bị đĩa

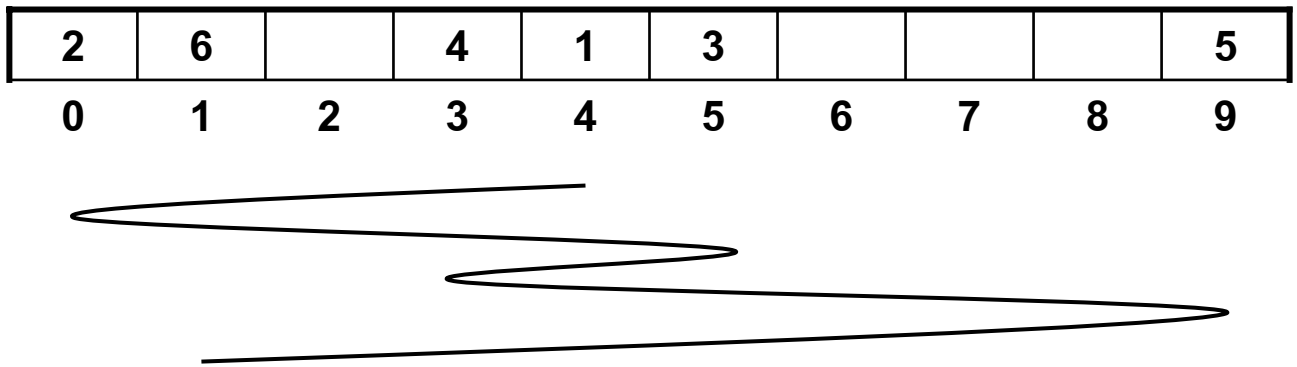
- Sơ lược về định dạng 1 sector 512bytes:
  - preamble + 4096bits + checksum/ECC
- Tổ chức chồng đĩa cứng:
  - các tracks cùng số hiệu tạo thành cylinder
  - tay dịch chuyển đầu từ: disk arm
- Thao tác seek:
  - có thể thực hiện song song trên nhiều ổ đĩa khác nhau, thậm chí cả với thao tác đọc/ghi
  - Tuy nhiên: không thể đọc/ghi trên cùng các ổ 1 lúc
  - Overlapped seek: trong khi bộ điều khiển thiết bị đang trong quá trình seek trên 1 ổ, có thể tiến hành seek trên 1 ổ khác

Các yếu tố tốc độ vào/ra đĩa

- Tốc độ vào/ra phụ thuộc:
  - thời gian seek
  - độ trễ quay vòng
  - tốc độ truyền dữ liệu

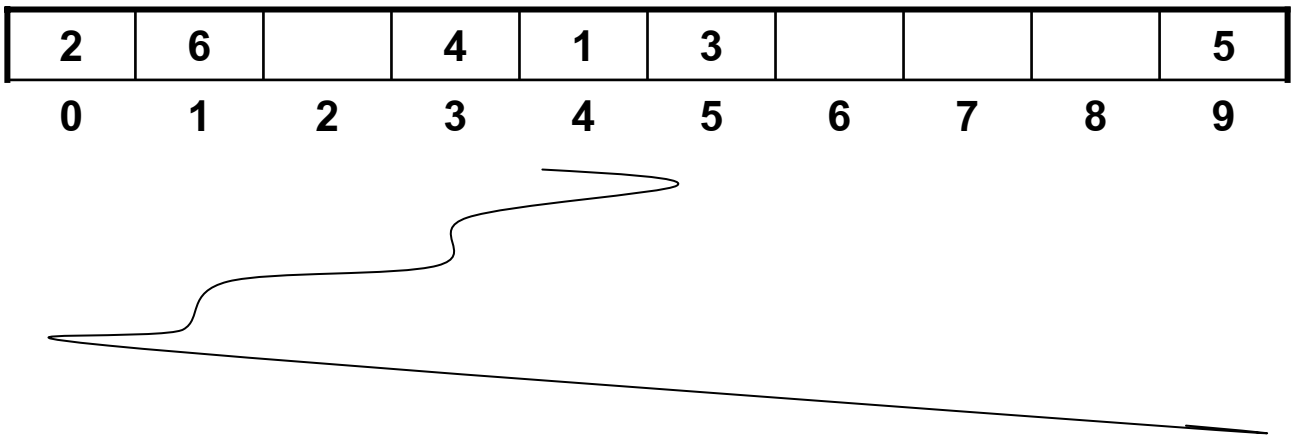
Thuật toán dịch chuyển đầu từ

- First Come First Serve
  - không kết hợp tốt khoảng cách dịch chuyển



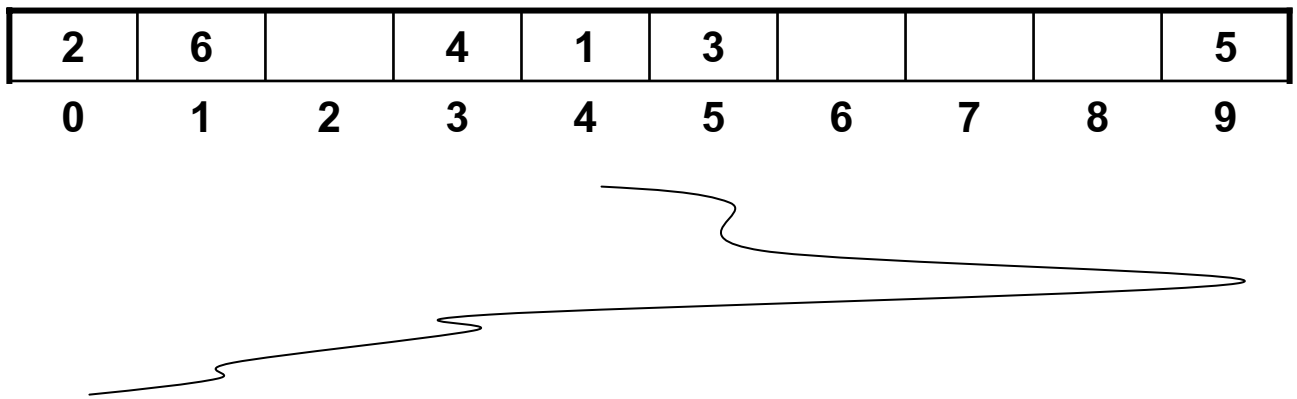
...thuật toán dịch chuyển đầu từ

- Shortest Seek First:
  - Sự cố đối với 2 vùng biên



...thuật toán dịch chuyển đầu từ

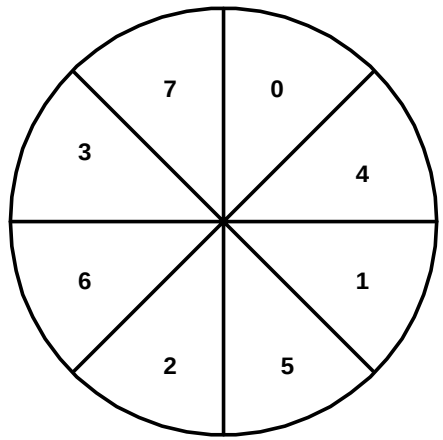
- Thuật toán elevator
  - ghi nhớ hướng đi: up/down
  - nếu không có yêu cầu up thì có thể down
  - nếu không có yêu cầu down thì có thể up



Bố trí sector

- Interleaving: thời gian dữ liệu chuyển từ bộ điều khiển tới bộ nhớ làm trễ việc đọc sector tiếp theo
- Giải pháp: bố trí sector theo thứ tự cách quãng.

  VD: 0 4 1 5 2 6 3 7



Redundant Array of Inexpensive Disks

- RAID: dùng nhiều đĩa để hứng chịu hỏng bằng mã sửa sai (khác với mirroring). VD:
  - 38 ổ đĩa cùng chạy song song
  - Thao tác đọc/ghi thực hiện trên tất cả các ổ đĩa theo đơn vị thứ tự từng bit
  - 38 bits được tạo thành từ một thanh 32 bits dữ liệu
  - Sử dụng thuật toán rải bit parity tại các vị trí 1, 2, 4, 8, 16 và 32

Thuật toán rải bit sửa sai

- Một số r parity bit sẽ được raỉ thêm vào trong xâu dư˜ liệu được truyền đi.

  Ví dụ: xâu trước khi raỉ là 10101011001 (11 bit) sẽ được raỉ thêm r=4 bit tại các vị trí 2^0, 2^1, 2^2, $2^3=2^{r-1}$.

  1 0 1 0 1 0 1 **?** 1 0 0 **?** 1 **?** **?**

- Giá trị các bít này sẽ được xác định như sau:

  Vị trí của các bit gốc có giá trị 1, sau khi đã raỉ là 15, 13, 11, 9, 7, 3. Cộng các số này theo modulo 2:

  $$(15 \oplus 13 \oplus 11 \oplus 9 \oplus 7 \oplus 3) = 4$$

  tức là 0100, chính là số checksum của các số ứng với vị trí có bit 1.

- Xâu bit được truyền đi là:

  1 0 1 0 1 0 1 **0** 1 0 0 **1** 1 **0** **0**

...thuật toán rải bit sửa sai

- Nếu khi nhận, có 1 bit sai, ví dụ tại vị trí 11:

1 0 1 0 0 0 1 **0** 1 0 0 **1** 1 **0** **0**

- bên nhận sẽ thực hiện tính lại số checksum các bit raỉ:

  15⊕13 ⊕ 9 ⊕ 7 ⊕ 3 ⊕ 4 = 11 chính là vị trí xaỷ ra bit sai.

- Tại sao vậy? Lý do: phép cộng modulo 2 chính là phép XOR, và

$$15 \oplus 13 \oplus 11 \oplus 9 \oplus 7 \oplus 3 \oplus 4 = 0$$

và $(15 \oplus 13 \oplus 11 \oplus 9 \oplus 7 \oplus 3 \oplus 4) \oplus 11 = 11$

- Ưu điểm: không chậm lắm, dễ làm
- Nhược điểm: chỉ khắc phục được 1 bit sai.

Thiết bị clock

- Duy trì giờ hệ thống
- Tạo bộ đếm hẹn giờ
- Ngăn tiến trình sử dụng quá thời gian
- Kiểm toán thời hạn sử dụng CPU của các tiến trình

Duy trì giờ hệ thống

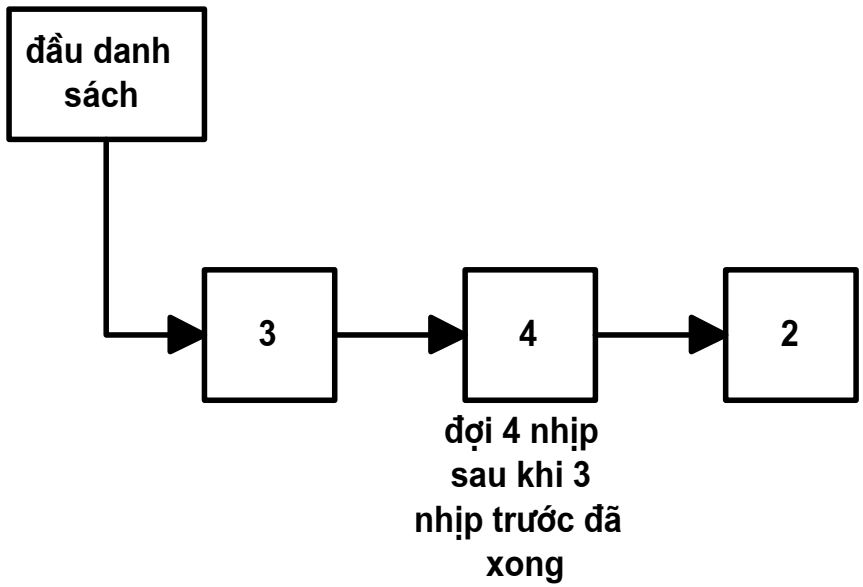
- Tạo một thanh đếm số giây từ 1 mốc đã định
  - vd: unix đếm số giây từ 01/01/1970 và lưu trong 1 thanh 32 bit có dấu, đủ cho 68 năm. Y2K+38 sẽ là sự cố cho Unix vào năm 2038 (khi đó Unix đã gần 68 tuổi)

Kiểm toán thời gian sử dụng CPU

- Lưu bộ đếm giờ riêng rẽ cho tiến trình, cất giữ và khôi phục mỗi khi có ngắt
- Hoặc: lưu vào trong bảng của mỗi tiến trình và tăng lên sau mỗi nhịp đồng hồ. Khó khăn: không trừ thời gian interrupt

Bộ hẹn giờ

- Lập một danh sách móc nối lưu số nhịp đếm còn lại trước khi "gọi" một tiến trình



Thiết bị màn hình cuối

- Vùng Memory-mapped screen
- Tệp tty (teletype) cho phép ánh xạ thiết bị màn hình cuối vào một tệp.
  - Trên Linux tệp tty được đặt tên pts. VD:

    USER TTY FROM

    root pts/1 -

    root pts/2 -

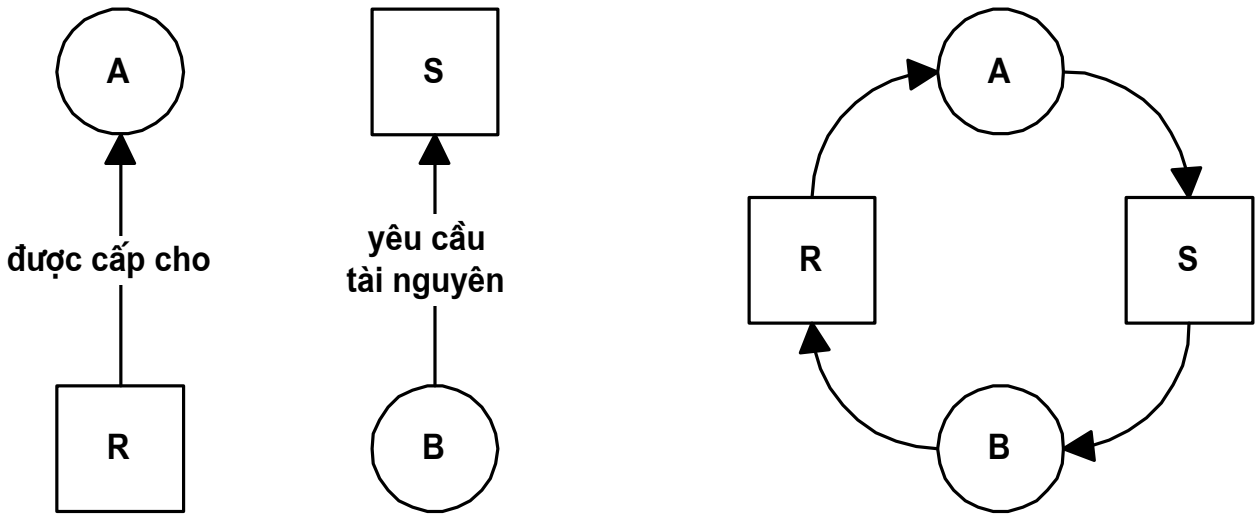
    hunghq pts/3 172.16.15.1

    lệnh *clear* > */dev/pts/3* sẽ gửi xâu escape xoá màn hình ra tệp màn hình cuối của hunghq
- Màn hình cuối câm: thụ động nhận dữ liệu hiển thị
- Các trạm cuối đồ hoạ cho phép thực hiện một số lệnh điều khiển

Deadlock

- Deadlock xảy ra khi có 4 điều kiện
  - Điều kiện luật loại trừ (Mutex)
  - Điều kiện giữ và chờ
  - Điều kiện không có quyền ưu tiên
  - Điều kiện chờ vòng tròn
- Thiết bị có thể phân thành 2 loại
  - chấp nhận quyền ưu tiên, vd: bộ nhớ
  - không chấp nhận quyền ưu tiên, vd: máy in

Mô hình deadlock



...mô hình deadlock

A	B	C
yêu cầu R	yêu cầu S	yêu cầu T
yêu cầu S	yêu cầu T	yêu cầu R
giải phóng R	giải phóng S	giải phóng T
giải phóng S	giải phóng T	giải phóng R

1. A yêu cầu R
2. B yêu cầu S
3. C yêu cầu T
4. A yêu cầu S
5. B yêu cầu T
6. C yêu cầu R

1. A yêu cầu R
2. C yêu cầu T
3. A yêu cầu S
4. C yêu cầu R
5. A giải phóng R
6. A giải phóng S
B tiếp tục ...

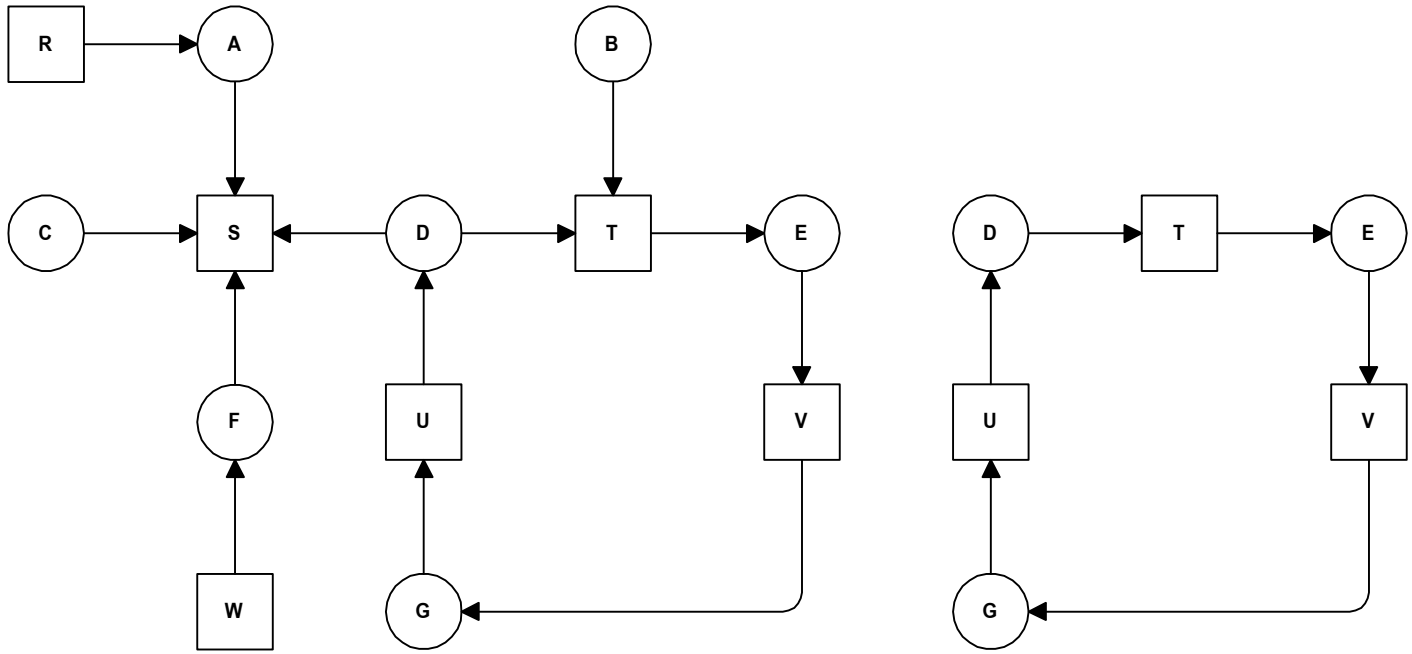
Đương đầu với deadlock

- Chấp nhận
- Phát hiện và khôi phục
- Ngăn chặn trong khi chạy bằng cách giám sát cấp phát tài nguyên
- Ngăn chặn bằng cách bẻ gẫy 1 trong 4 điều kiện gây deadlock

Phương pháp đà điểu

- Lờ đi, coi không có sự cố
- Khi có sự cố, cần can thiệp từ người quản trị bằng cách xoá 1 số tiến trình, đặt tạm kết quả in ra để in các tài liệu khác và tiếp tục lại sau đó...
- Căn cứ theo tần số xuất hiện sự cố và chi phí thực hiện, phương pháp này không phải là tồi

Phát hiện deadlock với 1 tài nguyên mỗi loại



giải pháp: kiểm tra tính chu trình của đồ thị

Phát hiện deadlock với nhiều tài nguyên mỗi loại

Danh sách tài nguyên: E1,..., Em

Tài nguyên sẵn sàng: A1,..., Am

C_{11}	C_{12}	C_{13}	...	C_{1m}
C_{21}	C_{22}	C_{23}	...	C_{2m}
...	...	...	...	...
C_{n1}	C_{n2}	C_{n3}		C_{nm}

$$\Sigma C_{ij} + A_j = E_j$$

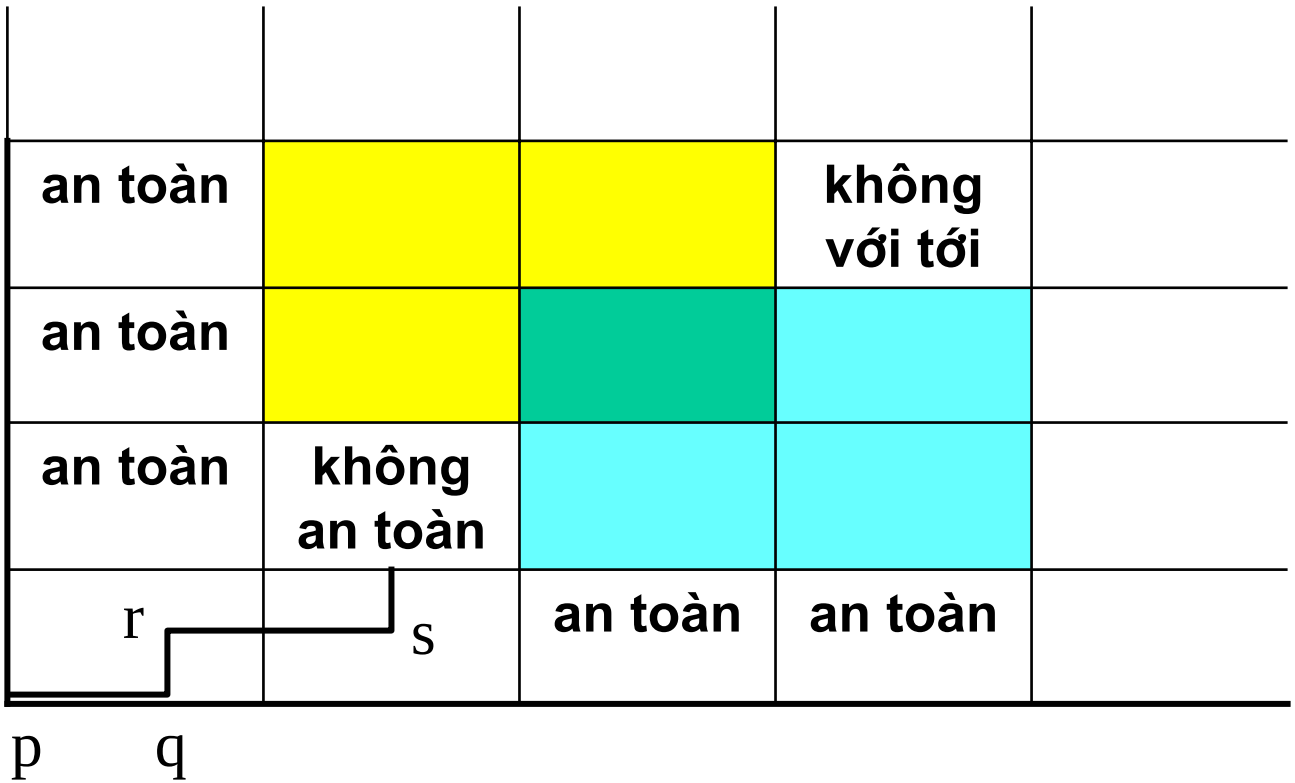
R_{11}	R_{12}	R_{13}	...	R_{1m}
R_{21}	R_{22}	R_{23}	...	R_{2m}
...	...	...	...	...
R_{n1}	R_{n2}	R_{n3}		R_{nm}

- **ma trận cấp phát tài nguyên hiện tại**
- **Hàng k ứng với tài nguyên cấp cho tiến trình k**
- **Hàng k ứng với nhu cầu của tiến trình k**

...phát hiện deadlock với nhiều tài nguyên mỗi loại

- Giải pháp:
  - Tìm trong R một hàng i, ứng với tiến trình i chưa xét, có yêu cầu tài nguyên nhỏ hơn tài nguyên hiện có trong A
  - Cộng hàng i trong C vào A và đánh dấu tiến trình i

Tránh deadlock



Banker's Algorithm với tài nguyên đơn

	Có	Max
A	3	9
B	2	4
C	2	7

còn 3

	Có	Max
A	4	9
B	2	4
C	2	7

còn 2

	Có	Max
A	4	9
B	4	4
C	2	7

còn 0

	Có	Max
A	4	9
B	0	-
C	2	7

còn 4

Giải pháp:

- trạng thái an toàn: có thể dẫn tới thoả mãn tất cả các yêu cầu
- trạng thái không an toàn: có thể dẫn tới deadlock
- trước khi cấp phát: so sánh phần còn lại, sau khi sẽ cấp phát, có thể vẫn thoả mãn toàn bộ yêu cầu tất cả khách hàng hay không. Nếu có, trạng thái sẽ đạt tới vẫn là an toàn

Banker's Algorithm với đa tài nguyên

	R1	R2	R3	R4
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

tài nguyên được cấp

	R1	R2	R3	R4
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

tài nguyên còn cần

Danh sách tài nguyên: E (6 3 4 2)

Tài nguyên còn rỗi: A (1 0 2 0)

Số tài nguyên dùng bởi tiến trình P (5 3 2 2)

...banker's Algorithm với đa tài nguyên

- Giải pháp:
  - Tra bảng xanh: tìm dòng i đáp ứng được bởi A và tiến trình Pi chưa được đánh dấu
  - Cộng hàng i của bảng vàng vào A, đồng thời đánh dấu Pi
  - Lặp lại bước trên đối với các tiến trình khác
  - Kết quả: nếu còn tiến trình chưa đánh dấu thì trạng thái là *không an toàn*

Ngăn ngừa deadlock – phá luật mutex -

- Dùng kỹ thuật xếp hàng đợi spooling và chương trình nền Daemon
- Nhược điểm:
  - không phải tất cả các thiết bị đều có thể dùng spool, vd: bảng danh sách tiến trình

Ngăn ngừa deadlock – phá luật giữ và đợi -

- Chiến thuật:
  - Pha 1: tiến trình liệt kê danh sách các tài nguyên sẽ dùng. Hệ thống sẽ thử cấp phát tất cả cùng 1 lúc. Nếu chưa được thì chờ 1 lúc và lặp lại
  - Pha 2: giải phóng được thực hiện khi tiến trình dùng xong tài nguyên
- Nhược điểm:
  - khó có thể liệt kê trước toàn bộ yêu cầu
  - lãng phí thời gian sử dụng tài nguyên

Ngăn ngừa deadlock – phá điều kiện không ưu tiên

-

- Đặt quyền ưu tiên sử dụng thiết bị cho các tiến trình
- Nếu deadlock xảy ra, tiến trình có mức cao có thể lấy tài nguyên từ các tiến trình khác
- Nhược điểm: thiết bị giành riêng, vd:máy in không thể bị tước đoạt giữa chừng

Ngăn ngừa deadlock – phá điều kiện chu trình -

- Giải pháp:
  - đánh số thứ tự các tài nguyên
  - tiến trình chỉ được thuê tài nguyên có số hiệu lớn hơn số hiệu các tài nguyên đang chiếm giữ và theo thứ tự tăng dần cho mỗi công việc
- Nhược điểm:
  - đánh số toàn bộ tài nguyên hệ thống (kể cả các bản ghi CSDL...) là quá lớn để trở thành hiện thực

Case study

- Unix áp dụng phương pháp đà điểu
- Lý do: chi phí đề phòng cao
- Do vậy unix hoàn toàn có thể bị tắc nghẽn. Ví dụ:
  - Bảng quản lý tiến trình có 100 ô. Có 10 tiến trình, mỗi ứng dụng đẻ ra 10 tiến trình con. Sự cố: 10 cha + 90 con đầy các ngăn của bảng.