Đề cương tham khảo Hệ điều hành

[CHƯƠNG 5B: ĐỊNH THỜI CPU NÂNG CAO 4](#_Toc136895736)

[Khái niệm lập lịch đa nhân. 4](#_Toc136895737)

[Cân bằng tải trên các hệ thống đa nhân sẽ cân bằng tải giữa các nhân CPU, mặc dù việc di chuyển các tiểu trình giữa các nhân để cân bằng tải có thể làm cho nội dung bộ nhớ cache bị vô hiệu và vì vậy có thể làm tăng thời gian truy cập bộ nhớ. 4](#_Toc136895738)

[Lập lịch thời gian thực mềm (Soft real-time) 4](#_Toc136895739)

[Lập lịch thời gian thực cứng (Hard realtime ) 4](#_Toc136895740)

[Lập lịch “Thời gian thực tỷ lệ đơn điệu” (Rate-monotonic real-time) 4](#_Toc136895741)

[Lập lịch “Tác vụ tới hạn” (EDF / Earliest-deadline-first) 4](#_Toc136895742)

[Lập lịch theo tỷ lệ phân bổ (Proportional share) 5](#_Toc136895743)

[Linux sử dụng bộ lập lịch “hoàn toàn công bằng”, cụ thể nó như thế nào? 5](#_Toc136895744)

[Bộ lặp lịch Windows có những đặc trưng gì. 5](#_Toc136895745)

[Mô hình hóa và mô phỏng có thể được sử dụng để đánh giá một thuật toán lập lịch CPU. 5](#_Toc136895746)

[CHƯƠNG 6: ĐỒNG BỘ HOÁ 5](#_Toc136895747)

[Cấu trúc của một chương trình gồm những phần code nào? 5](#_Toc136895748)

[Đoạn code nào là đối tượng cần đồng bộ? Tại sao? Bằng cách nào? 5](#_Toc136895749)

[Lệnh đơn nguyên là gì? Biến số đơn nguyên là gì? 6](#_Toc136895750)

[Rào cản (Barrier) là gì? Cách sử dụng? 6](#_Toc136895751)

[Mutex lock là gì? Cách sử dụng? 6](#_Toc136895752)

[Test\_and\_set() hoạt động ra sao? 6](#_Toc136895753)

[Semaphore là gì? Cách sử dụng? 6](#_Toc136895754)

[Monitor là gì? Cách nó hoạt động? 6](#_Toc136895755)

[Starvation nói đến tình trạng nào trong hệ thống? 6](#_Toc136895756)

[Giải thuật Peterson có cấu trúc ra sao? Đặc điểm và cách nó hoạt động. 7](#_Toc136895757)

[Tình trạng cạnh tranh (race condition) là gì? 7](#_Toc136895758)

[03 tiêu chí của một giải thuật đồng bộ là gì? 7](#_Toc136895759)

[Busy waiting là gì? Khi nào xuất hiện, tác động của nó đến hệ thống là gì? 7](#_Toc136895760)

[Tính sống còn (liveness) của các bài toán đồng bộ là gì? 7](#_Toc136895761)

[CHƯƠNG 7: CÁC VÍ DỤ ĐỒNG BỘ 7](#_Toc136895762)

[Xem lại bài toán Triết gia ăn tối. 7](#_Toc136895763)

[Xem lại bài Toán Bộ ghi - Bộ đọc. 8](#_Toc136895764)

[Xem lại bài toán Producer - Consumer. 8](#_Toc136895765)

[Với mỗi bài toán chú ý mô tả bài toán, vấn đề nằm ở đâu và giải pháp sử dụng là gì. 8](#_Toc136895766)

[CHƯƠNG 8: TẮC NGHẼN 8](#_Toc136895767)

[Đồ thị cấp phát tài nguyên RAG: mô hình, cách xây dựng. 8](#_Toc136895768)

[Khái niệm: hệ thống an toàn, hệ thống không an toàn, hệ thống bị tắc nghẽn được hiểu thế nào và tương quan của chúng ra sao? 9](#_Toc136895769)

[Giải thuật nhà băng. 9](#_Toc136895770)

[Thứ tự cấp phát và sử dụng tài nguyên của một loại tiến trình. 9](#_Toc136895771)

[4 điều kiện xảy ra tắc nghẽn là gì? 9](#_Toc136895772)

[Loại trừ tắc nghẽn được thực hiện ra sao? Cách nào là thực tế nhất? 10](#_Toc136895773)

[CHƯƠNG 9: BỘ NHỚ CHÍNH 10](#_Toc136895774)

[Khái niệm Cache / Memory / Register. 10](#_Toc136895775)

[Khái niệm trang (page), bảng trang / bảng phân trang (page table) và khung trang (frame) là gì? 10](#_Toc136895776)

[Địa chỉ luận lý, địa chỉ vật lý. 10](#_Toc136895777)

[Thanh ghi BASE, LIMIT có vai trò gì. 11](#_Toc136895778)

[3 giai đoạn gắn địa chỉ (binding) cho các biến số và lệnh là khi nào? Nguyên tắc và đặc tính của từng giai đoạn. 11](#_Toc136895779)

[Phân trang trong Windows: 11](#_Toc136895780)

[Hiện tượng phân mảnh nội xảy ra khi nào? Cách khắc phục? 11](#_Toc136895781)

[Hiện tượng phân mảnh ngoại xảy ra khi nào? Cách khắc phục? 11](#_Toc136895782)

[Các cách chọn lổ trống khi cấp phát liên tục: 12](#_Toc136895783)

[Chia sẻ trang: 12](#_Toc136895784)

[Bộ MMU: 12](#_Toc136895785)

[Số trang, số khung, độ dịch / offset: 12](#_Toc136895786)

[TLB là gì? 12](#_Toc136895787)

[Bảng phân trang nhiều mức: 12](#_Toc136895788)

[Bảng phân trang băm, bảng phân trang đảo ngược: 12](#_Toc136895789)

[Khái niệm về hoán đổi (Swapping): 13](#_Toc136895790)

[CHƯƠNG 10: BỘ NHỚ ẢO 13](#_Toc136895791)

[Khái niệm bộ nhớ ảo: 13](#_Toc136895792)

[Lỗi trang, các bước xử lý của hệ thống: 13](#_Toc136895793)

[Tính địa phương của tham khảo: 13](#_Toc136895794)

[Phân trang theo yêu cầu: 13](#_Toc136895795)

[Quy ước sao chép khi ghi (Copy on Write): 14](#_Toc136895796)

[Các giải thuật chọn frame nạn nhân: 14](#_Toc136895797)

[Thay thế toàn cục / Thay thế địa phương: 14](#_Toc136895798)

[Thrashing: 14](#_Toc136895799)

[CHƯƠNG 11: LƯU TRỮ THỨ CẤP 14](#_Toc136895800)

[Đặc trưng của đĩa cứng HDD và các thông số hoạt động của nó. 14](#_Toc136895801)

[Đặc trưng của SSD. 15](#_Toc136895802)

[Các thiết bị NVM. 15](#_Toc136895803)

[Định thời đĩa: FCFS, SCAN, C-SCAN. 15](#_Toc136895804)

[Các cấp độ RAID. 16](#_Toc136895805)

[Xem lại toàn bộ code của slide CH06. 16](#_Toc136895806)

**Đề cương tham khảo Hệ điều hành**

# CHƯƠNG 5B: ĐỊNH THỜI CPU NÂNG CAO

## Khái niệm lập lịch đa nhân.

Lập lịch đa nhân là việc quản lý và phân phối các tiểu trình trên nhiều nhân CPU trong hệ thống đa nhân.

## Cân bằng tải trên các hệ thống đa nhân sẽ cân bằng tải giữa các nhân CPU, mặc dù việc di chuyển các tiểu trình giữa các nhân để cân bằng tải có thể làm cho nội dung bộ nhớ cache bị vô hiệu và vì vậy có thể làm tăng thời gian truy cập bộ nhớ.

Cân bằng tải trên hệ thống đa nhân giúp phân phối công việc đồng đều giữa các nhân CPU, tuy nhiên, việc di chuyển tiểu trình có thể làm tăng thời gian truy cập bộ nhớ.

## Lập lịch thời gian thực mềm (Soft real-time)

Lập lịch thời gian thực mềm là khi hệ thống cố gắng đáp ứng các yêu cầu thời gian thực nhưng có thể chấp nhận một số sai lệch.

## Lập lịch thời gian thực cứng (Hard realtime )

Lập lịch thời gian thực cứng là khi hệ thống phải đáp ứng chính xác các yêu cầu thời gian thực mà không chấp nhận sai lệch.

## Lập lịch “Thời gian thực tỷ lệ đơn điệu” (Rate-monotonic real-time)

Lập lịch "Thời gian thực tỷ lệ đơn điệu" (Rate-monotonic real-time) là một thuật toán lập lịch ưu tiên các tiểu trình dựa trên tần suất của chúng.

## Lập lịch “Tác vụ tới hạn” (EDF / Earliest-deadline-first)

Lập lịch "Tác vụ tới hạn" (EDF / Earliest-deadline-first) là một thuật toán lập lịch ưu tiên các tiểu trình dựa trên thời hạn gần nhất của chúng.

## Lập lịch theo tỷ lệ phân bổ (Proportional share)

Lập lịch theo tỷ lệ phân bổ (Proportional share) là phương pháp phân phối tài nguyên CPU cho các tiểu trình dựa trên tỷ lệ được xác định trước.

## Linux sử dụng bộ lập lịch “hoàn toàn công bằng”, cụ thể nó như thế nào?

Linux sử dụng bộ lập lịch "hoàn toàn công bằng" để phân phối công việc đồng đều cho các tiểu trình.

## Bộ lặp lịch Windows có những đặc trưng gì.

Bộ lập lịch Windows có các đặc trưng riêng như lập lịch đa cấp, ưu tiên tiểu trình và lập lịch theo sự ưu tiên của người dùng.

## Mô hình hóa và mô phỏng có thể được sử dụng để đánh giá một thuật toán lập lịch CPU.

Mô hình hóa và mô phỏng có thể được sử dụng để đánh giá hiệu suất của một thuật toán lập lịch CPU bằng cách mô phỏng các tình huống và xem xét các chỉ số như thời gian đáp ứng và sự công bằng trong phân phối tài nguyên.

# CHƯƠNG 6: ĐỒNG BỘ HOÁ

## Cấu trúc của một chương trình gồm những phần code nào?

Cấu trúc của một chương trình bao gồm các phần code như khai báo biến, hàm, đoạn mã chính và các khối điều khiển như vòng lặp và điều kiện.

## Đoạn code nào là đối tượng cần đồng bộ? Tại sao? Bằng cách nào?

Đoạn code cần đồng bộ là đoạn code có khả năng bị cạnh tranh (race condition) khi được thực thi bởi nhiều tiểu trình. Để đồng bộ hóa, có thể sử dụng các cơ chế như mutex lock, semaphore, hay monitor.

## Lệnh đơn nguyên là gì? Biến số đơn nguyên là gì?

Lệnh đơn nguyên (atomic operation) là một lệnh không bị ngắt giữa quá trình thực hiện, và biến số đơn nguyên (atomic variable) là biến số có thể được đọc và ghi một cách nguyên tử, tránh tình trạng cạnh tranh.

## Rào cản (Barrier) là gì? Cách sử dụng?

Rào cản (Barrier) là một điểm trong chương trình mà các tiểu trình phải đến trước khi tiếp tục thực hiện, tạo ra sự đồng bộ hóa. Rào cản được sử dụng để đảm bảo các tiểu trình không vượt qua nhau trước khi đạt tới rào cản.

## Mutex lock là gì? Cách sử dụng?

Mutex lock là một cơ chế đồng bộ hóa được sử dụng để bảo vệ các phần của chương trình khỏi sự cạnh tranh. Khi một tiểu trình giữ mutex lock, các tiểu trình khác phải chờ đến khi nó được giải phóng để thực hiện phần mã đồng bộ.

## Test\_and\_set() hoạt động ra sao?

Test\_and\_set() là một hàm hoạt động như một lệnh đơn nguyên, kiểm tra giá trị của biến và đặt nó thành giá trị khác. Hàm này thường được sử dụng để triển khai mutex lock.

## Semaphore là gì? Cách sử dụng?

Semaphore là một cơ chế đồng bộ hóa dùng để quản lý tài nguyên chia sẻ. Nó có thể có giá trị không âm và cho phép một số tiểu trình truy cập đồng thời, trong khi số lượng truy cập vượt quá giá trị semaphore, các tiểu trình khác phải chờ đợi.

## Monitor là gì? Cách nó hoạt động?

Monitor là một cấu trúc dữ liệu đồng bộ hóa kết hợp mutex lock và điều kiện (condition) để quản lý các tiểu trình. Chỉ một tiểu trình có thể thực hiện các thao tác trên monitor tại một thời điểm.

## Starvation nói đến tình trạng nào trong hệ thống?

Starvation là tình trạng khi một tiểu trình không thể tiếp tục thực thi hoặc không được ưu tiên trong việc truy cập tài nguyên, do bị ưu tiên thấp hơn hoặc bị chặn bởi các tiểu trình khác.

## Giải thuật Peterson có cấu trúc ra sao? Đặc điểm và cách nó hoạt động.

Giải thuật Peterson là một giải thuật đồng bộ hai tiểu trình, sử dụng các biến flag và turn để kiểm soát quyền truy cập vào phần mã đồng bộ. Nó có cấu trúc gồm hai quy tắc if và while để đảm bảo sự tuần tự trong việc truy cập tài nguyên.

## Tình trạng cạnh tranh (race condition) là gì?

Tình trạng cạnh tranh (race condition) xảy ra khi hai hoặc nhiều tiểu trình cùng truy cập và thay đổi cùng một tài nguyên, dẫn đến kết quả không xác định hoặc không đúng.

## 03 tiêu chí của một giải thuật đồng bộ là gì?

3 tiêu chí của một giải thuật đồng bộ là sự hoàn thành (safety), sự liên tục (liveness) và không bị bế tắc (freedom from deadlock).

## Busy waiting là gì? Khi nào xuất hiện, tác động của nó đến hệ thống là gì?

Busy waiting là khi một tiểu trình tiếp tục kiểm tra điều kiện đồng bộ trong một vòng lặp mà không chuyển quyền điều khiển cho tiểu trình khác. Busy waiting có thể tạo ra sự lãng phí tài nguyên CPU và làm tăng độ trễ.

## Tính sống còn (liveness) của các bài toán đồng bộ là gì?

Tính sống còn (liveness) của các bài toán đồng bộ đề cập đến khả năng các tiểu trình hoàn thành công việc của mình, không bị bế tắc hoặc chết lặp.

# CHƯƠNG 7: CÁC VÍ DỤ ĐỒNG BỘ

## Xem lại bài toán Triết gia ăn tối.

* Mô tả bài toán: Một nhóm triết gia ngồi xung quanh một bàn ăn chung. Trên bàn có một đĩa mì và một đĩa đũa. Mỗi triết gia cần lấy cả hai đĩa để ăn mì, nhưng chỉ có một số lượng hữu hạn đĩa đũa. Triết gia cần có cả hai đĩa mới có thể ăn, nếu không sẽ đói.
* Vấn đề: Mỗi triết gia cần có đủ cả đĩa mì và đĩa đũa để ăn.
* Giải pháp: Sử dụng thuật toán như "Triết gia số 1 lấy đĩa đũa trước rồi mới lấy đĩa mì" để tránh tình trạng xảy ra "đói".

## Xem lại bài Toán Bộ ghi - Bộ đọc.

* Mô tả bài toán: Có một bộ ghi (writer) và nhiều bộ đọc (readers) cùng truy cập vào một vùng nhớ chia sẻ. Bộ ghi có quyền ghi vào vùng nhớ, trong khi bộ đọc có quyền đọc từ vùng nhớ. Cần đảm bảo rằng không có bộ đọc nào đọc dữ liệu khi bộ ghi đang ghi.
* Vấn đề: Đảm bảo tính nhất quán và đồng thời truy cập an toàn vào vùng nhớ chia sẻ khi có sự giao tiếp giữa bộ ghi và các bộ đọc.
* Giải pháp: Sử dụng các phương pháp đồng bộ hóa như Semaphore, Mutex lock hoặc Monitor để kiểm soát quyền truy cập vào vùng nhớ chia sẻ. Ví dụ, sử dụng Semaphore để chỉ cho phép một bộ ghi hoặc một số bộ đọc truy cập vào vùng nhớ cùng một thời điểm.

## Xem lại bài toán Producer - Consumer.

* Mô tả bài toán: Có một sản xuất viên (producer) tạo ra các mục (items) và đặt chúng vào buffer, cùng với đó có một người tiêu thụ (consumer) lấy các mục từ buffer để sử dụng. Producer và consumer cần hoạt động đồng thời mà không gây xung đột dữ liệu hoặc truy cập vào buffer trống/hết.
* Vấn đề: Đảm bảo tính nhất quán và đồng thời truy cập an toàn vào buffer khi có sự giao tiếp giữa producer và consumer.
* Giải pháp: Sử dụng các phương pháp đồng bộ hóa như Semaphore, Mutex lock hoặc Monitor để kiểm soát truy cập vào buffer. Ví dụ, sử dụng Semaphore để đếm số lượng các mục có sẵn trong buffer và kiểm soát việc sản xuất và tiêu thụ các mục.

## 

## Với mỗi bài toán chú ý mô tả bài toán, vấn đề nằm ở đâu và giải pháp sử dụng là gì.

# CHƯƠNG 8: TẮC NGHẼN

## Đồ thị cấp phát tài nguyên RAG: mô hình, cách xây dựng.

* Mô hình: Đồ thị cấp phát tài nguyên (Resource Allocation Graph - RAG) là một đồ thị hướng đồng thời được sử dụng để biểu diễn mối quan hệ giữa các tiến trình và tài nguyên trong một hệ thống.
* Cách xây dựng: Đồ thị RAG được xây dựng bằng cách tạo các đỉnh đại diện cho tiến trình và tài nguyên, sau đó sử dụng các cạnh để biểu thị việc cấp phát và sử dụng tài nguyên giữa các tiến trình và tài nguyên tương ứng.

## Khái niệm: hệ thống an toàn, hệ thống không an toàn, hệ thống bị tắc nghẽn được hiểu thế nào và tương quan của chúng ra sao?

* Hệ thống an toàn: Là trạng thái trong đó có thể tìm thấy một thứ tự thỏa mãn tất cả các yêu cầu tài nguyên của các tiến trình mà không xảy ra tắc nghẽn.
* Hệ thống không an toàn: Là trạng thái mà không có thứ tự nào có thể thỏa mãn tất cả các yêu cầu tài nguyên của các tiến trình mà không xảy ra tắc nghẽn.
* Hệ thống bị tắc nghẽn: Là trạng thái trong đó không có thứ tự nào có thể thỏa mãn tất cả các yêu cầu tài nguyên của các tiến trình mà không xảy ra tắc nghẽn, và các tiến trình không thể tiếp tục thực thi.

## Giải thuật nhà băng.

* Giải thuật nhà băng (Banker's algorithm) là một giải thuật sử dụng để kiểm tra một yêu cầu cấp phát tài nguyên từ các tiến trình trong hệ thống có dẫn đến tình trạng tắc nghẽn hay không.
* Giải thuật này kiểm tra tính an toàn của một yêu cầu cấp phát tài nguyên bằng cách tạo một trạng thái giả lập và kiểm tra xem có tồn tại một thứ tự cấp phát tài nguyên sao cho không xảy ra tắc nghẽn.

## Thứ tự cấp phát và sử dụng tài nguyên của một loại tiến trình.

* Thứ tự cấp phát tài nguyên của một loại tiến trình xác định cách mà tiến trình đó yêu cầu và sử dụng tài nguyên trong hệ thống.
* Thông thường, tiến trình cần yêu cầu tài nguyên trước khi sử dụng và phải hoàn trả tài nguyên sau khi sử dụng xong.

## 4 điều kiện xảy ra tắc nghẽn là gì?

* Sự yêu cầu không được thỏa mãn: Một tiến trình yêu cầu tài nguyên không có sẵn và phải chờ đợi.
* Không có khả năng chờ: Một tiến trình không thể chờ đợi và giữ tài nguyên đã có khi yêu cầu thêm tài nguyên khác.
* Không ưu tiên: Các tài nguyên không được ưu tiên cho những tiến trình đang chờ đợi.
* Vòng chờ: Có một chuỗi các tiến trình đang chờ đợi tài nguyên theo một cách hình thành vòng, mỗi tiến trình chờ đợi tài nguyên do tiến trình tiếp theo trong vòng sở hữu.

## Loại trừ tắc nghẽn được thực hiện ra sao? Cách nào là thực tế nhất?

* Để loại trừ tắc nghẽn, các phương pháp có thể được sử dụng bao gồm giải thuật nhà băng, sử dụng tài nguyên có ưu tiên, tăng cường thứ tự cấp phát tài nguyên và giới hạn số lượng tài nguyên mà mỗi tiến trình có thể yêu cầu.
* Một trong những cách thực tế nhất để loại trừ tắc nghẽn là sử dụng giải thuật nhà băng kết hợp với việc kiểm soát các yêu cầu tài nguyên, quản lý ưu tiên và tối ưu hóa việc cấp phát tài nguyên.

## 

# CHƯƠNG 9: BỘ NHỚ CHÍNH

## Khái niệm Cache / Memory / Register.

* Cache: Là một bộ nhớ tạm được sử dụng để lưu trữ các dữ liệu và hướng dẫn mà CPU sử dụng thường xuyên để tăng tốc độ truy cập dữ liệu.
* Memory: Là bộ nhớ chính trong máy tính, nơi lưu trữ dữ liệu và chương trình trong quá trình thực thi.
* Register: Là các vùng nhớ nhỏ và nhanh nằm trong CPU, được sử dụng để lưu trữ các giá trị trung gian và thực hiện các phép toán nhanh chóng.

## Khái niệm trang (page), bảng trang / bảng phân trang (page table) và khung trang (frame) là gì?

* Trang (page): Là một đơn vị nhỏ của bộ nhớ ảo, được chia thành các khối có cùng kích thước với khung trang.
* Bảng trang / bảng phân trang (page table): Là một cấu trúc dữ liệu được sử dụng để ánh xạ giữa địa chỉ ảo và địa chỉ vật lý. Nó chứa thông tin về các trang trong bộ nhớ ảo và khung trang tương ứng trong bộ nhớ vật lý.
* Khung trang (frame): Là các khối nhỏ của bộ nhớ vật lý, có cùng kích thước với trang. Khung trang được sử dụng để lưu trữ dữ liệu và chương trình khi chúng được đưa từ bộ nhớ ảo vào bộ nhớ vật lý.

## Địa chỉ luận lý, địa chỉ vật lý.

* Địa chỉ luận lý: Là địa chỉ được sử dụng trong không gian địa chỉ của một tiến trình. Nó là địa chỉ mà một tiến trình truy cập và sử dụng trong quá trình thực thi.
* Địa chỉ vật lý: Là địa chỉ thực tế trong bộ nhớ vật lý của hệ thống. Khi một tiến trình truy cập vào địa chỉ luận lý, hệ điều hành sẽ ánh xạ nó sang địa chỉ vật lý tương ứng để truy cập dữ liệu thực tế.

## Thanh ghi BASE, LIMIT có vai trò gì.

* Thanh ghi BASE: Lưu trữ địa chỉ bắt đầu của vùng nhớ được gán cho một tiến trình trong bộ nhớ vật lý.
* Thanh ghi LIMIT: Xác định kích thước của vùng nhớ được gán cho một tiến trình, giới hạn các địa chỉ có thể truy cập.

## 3 giai đoạn gắn địa chỉ (binding) cho các biến số và lệnh là khi nào? Nguyên tắc và đặc tính của từng giai đoạn.

* Giai đoạn biên dịch (compile-time binding): Địa chỉ được gán vào biến và lệnh trong quá trình biên dịch và không thay đổi sau đó.
* Giai đoạn liên kết (link-time binding): Địa chỉ được gán vào biến và lệnh trong quá trình liên kết và có thể thay đổi khi chương trình được tạo ra.
* Giai đoạn thực thi (run-time binding): Địa chỉ được gán vào biến và lệnh trong quá trình thực thi chương trình và có thể thay đổi trong suốt quá trình chạy.

## Phân trang trong Windows:

Phân trang là một kỹ thuật quản lý bộ nhớ ảo trong hệ điều hành Windows, trong đó không gian địa chỉ của mỗi tiến trình được chia thành các trang có kích thước cố định và được ánh xạ vào các khung trang trong bộ nhớ vật lý.

## Hiện tượng phân mảnh nội xảy ra khi nào? Cách khắc phục?

* Hiện tượng phân mảnh nội xảy ra khi không gian địa chỉ của một tiến trình trong bộ nhớ ảo bị chia thành nhiều khối nhỏ không liền kề, gây ra sự lãng phí không gian và khó khăn trong việc cấp phát bộ nhớ cho các tiến trình.
* Cách khắc phục phân mảnh nội có thể bao gồm thu gom rác (garbage collection), nén bộ nhớ và tái cấu trúc không gian địa chỉ của tiến trình.

## Hiện tượng phân mảnh ngoại xảy ra khi nào? Cách khắc phục?

* Hiện tượng phân mảnh ngoại xảy ra khi không gian bộ nhớ vật lý không đủ để chứa các tiến trình hoạt động. Các khung trang trong bộ nhớ vật lý không liên tục và gây ra sự lãng phí tài nguyên.
* Cách khắc phục phân mảnh ngoại có thể bao gồm việc sử dụng kỹ thuật hoán đổi (swapping), nâng cấp bộ nhớ vật lý hoặc sử dụng các kỹ thuật quản lý bộ nhớ hiệu quả hơn.

## Các cách chọn lổ trống khi cấp phát liên tục:

Có thể sử dụng các thuật toán như Best-Fit, First-Fit hoặc Next-Fit để chọn lổ trống phù hợp nhất cho quá trình cấp phát liên tục của các khối bộ nhớ.

## Chia sẻ trang:

Chia sẻ trang là kỹ thuật cho phép nhiều tiến trình chia sẻ cùng một trang trong bộ nhớ ảo, giúp tiết kiệm tài nguyên và tăng hiệu suất hệ thống.

## Bộ MMU:

Bộ MMU (Memory Management Unit) là một thành phần trong CPU, được sử dụng để quản lý việc ánh xạ giữa địa chỉ ảo và địa chỉ vật lý trong quá trình truy cập bộ nhớ.

## Số trang, số khung, độ dịch / offset:

Số trang là số lượng trang trong không gian địa chỉ ảo, số khung là số lượng khung trang trong bộ nhớ vật lý. Độ dịch (offset) là khoảng cách giữa địa chỉ trang và địa chỉ bắt đầu của khung trang.

## TLB là gì?

TLB (Translation Lookaside Buffer) là một bộ nhớ cache nhỏ, được sử dụng để lưu trữ các bản dịch gần đây của bộ MMU, nhằm tăng tốc độ truy cập vào bộ nhớ và giảm thời gian ánh xạ địa chỉ.

## Bảng phân trang nhiều mức:

Bảng phân trang nhiều mức là một cấu trúc dữ liệu được sử dụng trong quá trình ánh xạ giữa địa chỉ ảo và địa chỉ vật lý, trong đó bảng phân trang được chia thành nhiều cấp để tối ưu hóa việc quản lý bộ nhớ.

## Bảng phân trang băm, bảng phân trang đảo ngược:

Bảng phân trang băm (hashed page table) là một cấu trúc dữ liệu sử dụng hàm băm để ánh xạ địa chỉ ảo vào địa chỉ vật lý. Bảng phân trang đảo ngược (inverted page table) lưu trữ ánh xạ ngược từ địa chỉ vật lý vào địa chỉ ảo, giúp tiết kiệm không gian bộ nhớ.

## Khái niệm về hoán đổi (Swapping):

Hoán đổi là kỹ thuật chuyển đổi dữ liệu và chương trình từ bộ nhớ vật lý sang bộ nhớ trên đĩa và ngược lại, nhằm giải phóng tài nguyên và tạo không gian cho các tiến trình khác.

# CHƯƠNG 10: BỘ NHỚ ẢO

## Khái niệm bộ nhớ ảo:

Bộ nhớ ảo là một kỹ thuật trong hệ điều hành cho phép mở rộng không gian bộ nhớ bằng cách sử dụng ổ cứng như một phần của bộ nhớ. Nó cho phép chương trình chạy trong không gian địa chỉ lớn hơn thực tế có sẵn trong bộ nhớ vật lý.

## Lỗi trang, các bước xử lý của hệ thống:

Lỗi trang xảy ra khi chương trình yêu cầu một trang không có sẵn trong bộ nhớ vật lý. Các bước xử lý của hệ thống bao gồm:

1. Kiểm tra trang có trong bộ nhớ vật lý hay không.
2. Nếu trang không có trong bộ nhớ vật lý, tạo lỗi trang.
3. Xác định trang nạn nhân để thay thế (nếu cần).
4. Tải trang từ bộ nhớ ảo vào bộ nhớ vật lý.
5. Cập nhật bảng phân trang và tiếp tục chương trình.

## Tính địa phương của tham khảo:

Tính địa phương của tham khảo chỉ ra rằng trong một khoảng thời gian ngắn, chương trình thường chỉ sử dụng một phần nhỏ của bộ nhớ, không sử dụng tất cả các trang cùng một lúc.

## Phân trang theo yêu cầu:

Phân trang theo yêu cầu (demand paging) là một kỹ thuật trong bộ nhớ ảo, trong đó các trang chỉ được tải vào bộ nhớ vật lý khi chúng thực sự được yêu cầu bởi chương trình.

## Quy ước sao chép khi ghi (Copy on Write):

Quy ước sao chép khi ghi là một kỹ thuật trong bộ nhớ ảo, trong đó các trang được chia sẻ giữa các tiến trình và chỉ được sao chép khi tiến trình ghi vào trang đó.

## Các giải thuật chọn frame nạn nhân:

* FIFO (First-In-First-Out): Chọn khung trang được tạo ra lâu nhất để thay thế.
* Optimal: Chọn khung trang mà trang sẽ không được sử dụng trong thời gian dài nhất trong tương lai.
* LRU (Least Recently Used): Chọn khung trang mà trang không được sử dụng lâu nhất.
* Second Chance: Kết hợp giữa FIFO và LRU, cho phép trang có "cơ hội thứ hai" trước khi bị thay thế.

## Thay thế toàn cục / Thay thế địa phương:

Thay thế toàn cục (global replacement) là khi cần thay thế một trang, hệ thống có thể chọn một trang ở bất kỳ tiến trình nào. Thay thế địa phương (local replacement) là khi chỉ các trang của tiến trình đang chạy có thể được chọn để thay thế.

## Thrashing:

Thrashing là hiện tượng khi hệ thống dành nhiều thời gian swap trang giữa bộ nhớ vật lý và ổ đĩa mà không thực hiện được công việc chính. Nó xảy ra khi bộ nhớ không đủ lớn để chứa tất cả các trang cần thiết cho các tiến trình đang hoạt động, và hệ thống mắc kẹt trong việc swap trang liên tục, dẫn đến giảm hiệu suất và tăng thời gian thực hiện công việc.

# CHƯƠNG 11: LƯU TRỮ THỨ CẤP

## Đặc trưng của đĩa cứng HDD và các thông số hoạt động của nó.

Đĩa cứng HDD (Hard Disk Drive) là thiết bị lưu trữ dùng cơ chế quay đĩa để ghi và đọc dữ liệu. Các đặc trưng và thông số hoạt động của HDD bao gồm: tốc độ quay, dung lượng lưu trữ, thời gian truy xuất, tốc độ truyền dữ liệu, giao diện kết nối (ví dụ: SATA), độ ồn, tuổi thọ.

## Đặc trưng của SSD.

Đặc trưng của SSD: SSD (Solid State Drive) là thiết bị lưu trữ không có bộ phận cơ khí, sử dụng bộ nhớ flash để lưu trữ dữ liệu. Các đặc trưng của SSD bao gồm: tốc độ truy xuất nhanh, tốc độ truyền dữ liệu cao, không có bộ phận cơ khí nên ít ồn và tiêu thụ năng lượng thấp, tuổi thọ cao hơn so với HDD.

## Các thiết bị NVM.

Các thiết bị NVM: NVM (Non-Volatile Memory) là dạng bộ nhớ không bay hơi, tức là dữ liệu vẫn được lưu trữ ngay cả khi không có nguồn điện. Các thiết bị NVM bao gồm các loại như MRAM (Magnetoresistive RAM), PCM (Phase Change Memory), ReRAM (Resistive RAM), và 3D XPoint.

## Định thời đĩa: FCFS, SCAN, C-SCAN.

Định thời đĩa là các thuật toán được sử dụng để quản lý việc truy xuất dữ liệu trên đĩa cứng. Các thuật toán phổ biến bao gồm FCFS (First-Come, First-Served), SCAN và C-SCAN. Dưới đây là giải thích về mỗi thuật toán:

1. FCFS (First-Come, First-Served): Đây là thuật toán đơn giản nhất trong định thời đĩa. Khi có yêu cầu truy xuất, nó được xử lý theo thứ tự đến trước đến trước. Nghĩa là yêu cầu truy xuất đầu tiên được đưa vào hàng đợi sẽ được xử lý trước. FCFS không quan tâm đến vị trí của các yêu cầu trên đĩa, điều này có thể dẫn đến hiện tượng "cân nặng" (head starvation) khi các yêu cầu phía sau phải chờ đợi lâu hơn.
2. SCAN: Thuật toán SCAN di chuyển đầu đọc/ghi trên đĩa từ một đầu đến đầu kia và xử lý các yêu cầu truy xuất trên đường đi đó. Khi nó đến đầu đĩa, nó sẽ đảo hướng và tiếp tục xử lý các yêu cầu truy xuất còn lại. SCAN giảm thiểu hiện tượng "cân nặng" bằng cách xử lý các yêu cầu truy xuất ở cả hai hướng của đĩa. Tuy nhiên, nó có thể dẫn đến hiện tượng "đơ" (head starvation) nếu có nhiều yêu cầu tập trung ở một khu vực của đĩa.
3. C-SCAN (Circular SCAN): Tương tự như thuật toán SCAN, C-SCAN cũng di chuyển đầu đọc/ghi từ một đầu đến đầu kia của đĩa. Tuy nhiên, khi nó đến đầu cuối, nó sẽ quay trở lại đầu đĩa mà không xử lý bất kỳ yêu cầu truy xuất nào trên đường đi đó. Điều này giúp tránh hiện tượng "đơ" và đảm bảo rằng mọi yêu cầu truy xuất được xử lý. C-SCAN đảm bảo công bằng hơn cho tất cả các yêu cầu truy xuất mà không tạo ra sự chênh lệch lớn giữa thời gian đợi của chúng.

Tùy thuộc vào tình huống và yêu cầu của hệ thống, mỗi thuật toán định thời đĩa có ưu điểm và hạn chế riêng, và việc chọn thuật toán phù hợp sẽ phụ thuộc vào các yếu tố như độ ưu tiên, hiệu suất và quy mô hệ thống.

## Các cấp độ RAID.

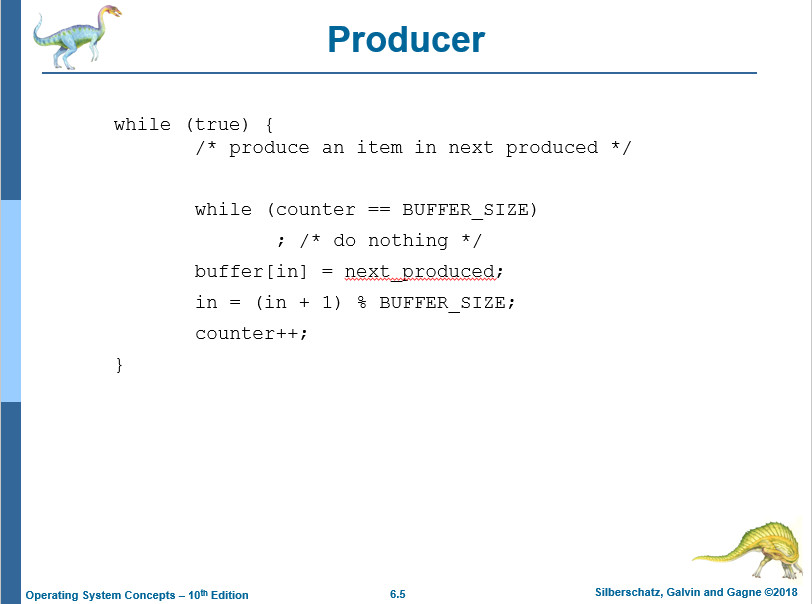
Các cấp độ RAID: RAID (Redundant Array of Independent Disks) là một phương pháp kết hợp nhiều ổ đĩa để cung cấp khả năng dự phòng và tăng hiệu suất đọc/ghi dữ liệu. Các cấp độ RAID bao gồm: RAID 0, RAID 1, RAID 5, RAID 6, RAID 10, và nhiều hơn nữa. Mỗi cấp độ RAID có cách hoạt động và mục đích sử dụng khác nhau.

# 

# Xem lại toàn bộ code của slide CH06.

* **Đoạn code trên là một phần của quá trình "tiêu thụ" (consume) trong bài toán "producer-consumer" (người sản xuất - người tiêu thụ). Đoạn code này được sử dụng để lấy (consume) một mục (item) từ buffer (vùng đệm).,đoạn code thực hiện việc sau:**

1. Vòng lặp while (true) đảm bảo việc tiêu thụ các mục liên tục.
2. Trong vòng lặp, nếu counter (biến đếm) đạt đến giá trị 0, nghĩa là buffer đang trống, vòng lặp trong while (counter == 0) sẽ chờ cho đến khi buffer có mục để tiêu thụ.
3. Sau khi buffer có mục để tiêu thụ, mục được lấy từ buffer tại vị trí out (biến chỉ số out) và gán cho biến next\_consumed.
4. Biến out được cập nhật bằng cách tăng giá trị lên một đơn vị và lấy phần dư khi chia cho BUFFER\_SIZE để quay vòng trong buffer.
5. Biến counter (biến đếm) được giảm đi một đơn vị để thể hiện sự giảm số lượng các mục trong buffer.



* **Đoạn code trên là một phần của quá trình "sản xuất" (produce) trong bài toán "producer-consumer" (người sản xuất - người tiêu thụ). Đoạn code này được sử dụng để thêm (produce) một mục (item) mới vào buffer (vùng đệm), đoạn code thực hiện việc sau:**

1. Vòng lặp while (true) đảm bảo việc sản xuất các mục liên tục.
2. Trong vòng lặp, nếu counter (biến đếm) đạt đến giới hạn BUFFER\_SIZE (kích thước của buffer), nghĩa là buffer đã đầy, vòng lặp trong while (counter == BUFFER\_SIZE) sẽ chờ cho đến khi buffer có chỗ trống.
3. Sau khi buffer có chỗ trống, mục next\_produced được thêm vào buffer tại vị trí in (biến chỉ số in).
4. Biến in được cập nhật bằng cách tăng giá trị lên một đơn vị và lấy phần dư khi chia cho BUFFER\_SIZE để quay vòng trong buffer.
5. Biến counter (biến đếm) được tăng lên để thể hiện sự tăng số lượng các mục trong buffer.