**NỘI DUNG ÔN TẬP CUỐI KỲ**

**Môn:** 502047 - Nhập môn Hệ Điều Hành

**Hình thức thi:** Trắc nghiệm (Làm bài trên máy tính)

**Số lượng câu:** 45 câu (Câu hỏi bằng tiếng Việt)

**Thời gian:** 60 phút

**Nội dung thi:** Từ chương 3 đến 11

**Nguồn tài liệu:**

*- Slide bài giảng: các thuật ngữ in đậm, mô hình, biểu đồ, ví du.*

*- Tóm tắt bài giảng tuần, tải về bằng cách đăng nhập thư viện,*

*- Bài tập tuần: xem ở sách giáo trình chính sau mỗi chương (bằng Tiếng Anh), hoặc tại thư viện online (bằng tiếng Việt).*

**Chương 3 - Tiến trình**

- Khái niệm, thuật ngữ, định nghĩa.

+ Khái niệm : là một chương trìnhg  
- Khái niệm tiến trình và các thành phần của nó

+ **Tiến trình** (tiếng Anh: **process**) là một thực thể (instance) của một chương trình máy tính đang được thực thi bởi một hoặc nhiều [luồng](https://vi.wikipedia.org/wiki/Lu%E1%BB%93ng_(%C4%91i%E1%BB%87n_to%C3%A1n)) (thread)..

+ Các thành phần của tiến trình:

* **Text Section (Program code) :** Chứa những đoạn mã chương trình đã được biên dịch bởi compiler.
* **Data Section (khu vực dữ liệu) :** Chứa các biến toàn cục (global variables) và các biến tĩnh (static variables) được khởi tạo trước khi hàm main được gọi.
* **Heap :** Dùng để lưu trữ các bộ nhớ được cấp phát động (như việc gọi new, delete, malloc, calloc, free,…).
* **Stack :** Dùng để lưu trữ các biến cục bộ (local variables). VD như lúc bạn khai báo trong C : int i = 0. Thì biến i sẽ được lưu trong bộ nhớ Stack của Process.
* **Process ID (PID) :** để định danh mã tiến trình

- Trạng thái các tiến trình

+ new: tiến trình vừa được tạo

+ ready: tiến trình đã có đủ tài nguyên, chỉ còn cần CPU để thực thi

+ running: các câu lệnh của tiến trình đang được thực thi

+ waiting: còn được gọi là blocked, nghĩa là tiến trình đang đợi I/O hoàn tất, và ra tín hiệu để tiếp tục các trạng thái khác

+ terminated: tiến trình đã kết thúc

- Khối PCB, định thời tiến trình

+ PCB : là một trong các cấu trúc dữ liệu quan trọng nhất của hệ điều hành

+ Mỗi tiến trình trong hệ thống đều được cấp phát một PCB

+ PCB gồm:

* Trạng thái tiến trình: new, ready, running, …
* Bộ đếm chương trình (PC)
* Các thanh ghi
* Thông tin lập thời biểu CPU: độ ưu tiên,…
* Thông tin quản lý bộ nhớ
* Thông tin, lượng CPU, thời gian sử dụng
* Thông tin trạng thái I/O

+ Bộ định thời các tiến trình sẽ chọn một trong số các tiến trình đang hiện có sẵn cho lần thực thi tiếp theo trong nhân của CPU

+ Mục tiêu: tận dụng tối đa CPU

+ Các hàng đợi định thời:

* Hàng đợi công việc – Job queue
* Hàng đợi sẵn sàng – Ready queue
* Hàng đợi thiết bị - Device queues
* …

- Chuyển ngữ cảnh:

+ Chuyển đổi ngữ cảnh xảy ra khi CPU của máy tính chuyển từ một quy trình hoặc luồng sang một quy trình hoặc luồng khác

+ Chuyển đổi ngữ cảnh cho phép một CPU xử lý nhiều tiến trình hoặc luồng mà không cần bộ xử lý bổ sung. Bất kỳ hệ điều hành nào cho phép đa nhiệm chủ yếu dựa vào việc sử dụng chuyển đổi ngữ cảnh để cho phép các tiến trình khác nhau chạy cùng một lúc. Có ba trường hợp cần chuyển đổi ngữ cảnh, như hình dưới đây.

* Đa nhiệm – Khi CPU cần chuyển đổi các tiến trình trong và ngoài bộ nhớ để nhiều hơn một tiến trình có thể đang chạy.
* Kernel \/User Switch – Khi chuyển đổi giữa chế độ người dùng sang chế độ kernel, nó có thể được sử dụng (nhưng không phải lúc nào cũng cần thiết).
* Ngắt – Khi CPU bị ngắt để trả về dữ liệu từ đĩa đã đọc.

- Đa nhiệm trong thiết bị di động

Một số thiết bị di động chỉ cho phép duy nhất 1 tiến trình chạy một thời điểm, những tiến trình khác phải bị hoãn lại (sớm nhất là iOS)

- Các mô hình giao tiếp giữa các tiến trình:

+ Trao đổi bằng tín hiệu (signal handling)

+ Trao đổi bằng cơ chế đường ống (unnamed pipe & named pipe)

+ Trao đổi thông qua hàng đợi thông điệp (message queue)

+ Trao đổi bằng phân đoạn nhớ chung (Shared memory)

+ Giao tiếp đồng bộ dung semaphore

+ Giao tiếp thông qua socket

- Khái niệm tiến trình cộng tác

+ Trong tiến trình thực thi, các tiến trình có thể cộng tác (cooperate) để hoàn thành các công việc.

+ Các tiến trình cộng tác với nhau để :

* Chia sẻ dữ liệu (information sharing).
* Tăng tốc độ tính toán (computational speedup).
  + Các mạng lưới máy tính sẽ hợp với nhau để tạo thành các cluster.
  + Nếu hệ thống có nhiều CPU, chia công việc tính toán thành nhiều công việc tính toán nhỏ chạy song song.
* Thực hiện một công việc chung.
  + Xây dựng một phần mềm phức tạp bằng cách chia thành các module/process hợp tác nhau.

- Hiểu các tính chất của đường liên lạc.

- Biết về các loại đường ống trong UNIX/Windows:

+ Unamed pipe: chỉ sử dụng cho việc giao tiếp giữa tiến trình cha và con

+ Named pipe: sử dụng cho việc giao tiếp giữa 2 tiến trình unamed và các tiến trình cha có thể chia sẽ dữ liệu qua các đường ống named pipe và named pipe tồn tại dưới dạng file hệ thống

- Biết về Socket và RPC:

+ Socket:là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều, hay còn gọi là two-way communication để kết nối 2 process trò chuyện với nhau. Điểm cuối (endpoint) của liên kết này được gọi là socket.

+ RPC:  là một mô hình kỹ thuật mạng hay còn được biết đến là cơ chế giao tiếp giữa hai tiến trình. (**Remote Procedure Call**)

- Thao tác tạo và huỷ tiến trình:

+ fork() : tạo tiến trình

+ exit() : để thoát khỏi và kết thúc tiến trình

- Ví dụ về đa tiến trình trong Chrome: Trình duyệt Chrome có thể vẫn hoạt động ngay cả khi cần đóng bất kỳ tab nào. Mỗi tab là mỗi tiến trình  
- Giới thiệu bài toán Producer - Consumer (dùng bounded buffer và shared varibles)  
- Các câu hỏi trong phần "bài tập" sau chương"  
- Các phương pháp giao tiếp giữa các tiến trình.  
- Hiểu các đoạn mã sử dụng lời gọi tạo tiến trình.

**Chương 4 - Tiểu trình và Đồng thời**

- Khái niệm, thuật ngữ, định nghĩa.

+ Tiểu trình : Là một đơn vị cơ bản sử dụng CPU, gồm :

* Thread ID.
* PC (Program Counter).
* Registers.
* Stack.
* Chia sẻ chung code, data, resources (file).

- Những lý do cần tiến trình đa luồng: để cho việc thực thi đồng thời đa các tác vụ với nhau và thay vì xài cấu trúc đa tiến trình để xử lý các tác vụ đơn lẻ thì sử dụng 1 tiến trình với đa luồng sẽ lợi ích hơn bởi vì ở kiến trúc đa luồng các tác vụ sẽ có thể chia sẽ tài nguyên với nhau còn ở tiến trình thì không thể. Nhưng lí do chính là để cải thiện hiệu suất của nó. Hiệu suất có thể được thể hiện theo nhiều cách:

* A web server will utilize multiple threads to simultaneous process requests for data at the same time.
* An image analysis algorithm will spawn multiple threads at a time and segment an image into quadrants to apply filtering to the image.
* A ray-tracing application will launch multiple threads to compute the visual effects while the main GUI thread draws the final results.

- Những lợi ích của đa luồng:

+ Chia sẻ tài nguyên

+ Phản ứng (reponsiveness)

+ Tối ưu hóa kiến trúc đa tiến trình

+ Kinh tế và chi phí sản xuất

- Những khó khăn của lập trình đa luồng:

+ Dev cần phải kỹ lưỡng về việc đồng bộ giữa các threads

+ Lập trình đa luồng có thể ngốn rất nhiều blocked thread

+ Khi gặp lỗi rất khó để sửa

+ Rất khó để xử lí tính đồng thời giữa các threads

- Phân biệt đồng thời và song song:

+ Đồng thời : thường là về việc xử lí nhiều thứ cùng một lúc

+ Song song: thì thường không nói về việc xử lí nhiều việc cùng 1 lúc

- Hiểu về song song dữ liệu và song song tác vụ

+ Song song dữ liệu có nghĩa là việc thực thi đồng thời cùng 1 tác vụ trên mỗi nhân tính toán

+ Song song tác vụ có nghĩa là việc thực thi đồng thời các tác vụ khác nhau trên nhiều nhân tính toán

- Phân biệt và chỉ ra sự khác nhau của tiến trình đơn và đa luồng.

+ Sự khác nhau chính giữa tiến trình đơn và đa luồng là tiến trình đơn luồng thì chỉ duy nhất 1 luồng thực thi nhiều tác vụ của 1 tiến trình trong khi tiến trình đa luồng, thì nhiều luồng cùng xử lí nhiều tác vụ của 1 tiến trình

- Ví dụ đa luồng trong kiến trúc server.

+ Trong một hệ thống server của ngân hàng thì khi khách hàng gửi yêu cầu để sử dụng dịch vụ thì 1 thread sẽ được tạo ra để khách hàng có thể giao tiếp với server. Và đó là khi chỉ có 1 khách hàng, còn trong trường hợp nhiều khách hàng gửi yêu cầu thì chúng ta phải tạo ra nhiều luồng để đáp ứng yêu cầu của nhiều khách hàng ở cùng thời điểm

- Hiểu và tính được Luật Amdahl.  
- Hiểu luồng mức người dùng và luồng mức nhân cùng 3 mô hình ánh xạ giữa chúng.  
- Biết về khái niệm signal và signal handler.

- Các câu hỏi trong phần "bài tập" sau chương"  
- Hiểu các đoạn mã sử dụng lời gọi tạo tiểu trình và đồng bộ chúng

**CHƯƠNG 5B: ĐỊNH THỜI CPU NÂNG CAO**

- Khái niệm lập lịch 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.

- Lập lịch thời gian thực mềm (Soft real-time) / Lập lịch thời gian thực cứng (Hard real-time

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

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

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

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

- Bộ lặp lịch Windows có những đặc trưng gì.

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

**CHƯƠNG 6: ĐỒNG BỘ HOÁ**

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

+ Entry section: là một phần của tiến trình quyết định sự xâm nhập của một tiến trình cụ thể , đoạn mã này được sử dụng để kiểm soát quá trình đồng bộ

+ Critical section : phần mã này cho phép 1 tiến trình xâm nhập vào làm thay đổi các biến chung (toàn cục) dễ xảy ra lỗi

+ Exit section : cho phép các tiến trình khác đang chờ trong phần Entry section để vào phần Critical. Nó cũng kiểm tra xem 1 tiến trình đã hoàn thành quá trình thực thi nếu xong thì loại bỏ ngay.

+ Remainer section: Phần còn lại của đoạn code (đoạn mã này có thể chạy cùng lúc mà không gây ra sai sót dữ liệu)

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

* Critical section vì đây là một phân đoạn mã có thể được truy cập chỉ bằng một quy trình tại một thời điểm. Phần quan trọng chứa các biến được chia sẻ cần được đồng bộ hóa để duy trì tính nhất quán của các biến dữ liệu. Trong phần nhập, tiến trình yêu cầu nhập vào Critical section
* Bằng cách sử dụng: Peterson’s solution với 2 biến là boolean flag và int turn để kiểm soát 3 điều kiện sau:

1. Mutual exclusion được đảm bảo rằng là chỉ duy nhất 1 tiến trình có thể truy cập vào Critical section tại 1 thời điểm
2. Tiến trình cũng được đảm bảo, khi một tiến trình bên ngoài phần quan trọng không chặn các quy trình khác vào phần quan trọng
3. Bounded Waiting được bảo tồn vì mọi quy trình đều có cơ hội công bằng

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

+ Các thao tác lên biến số này tuần tự được thực thi trong CPU

+ Lệnh đơn nguyên là có nghĩa là lệnh đó không có khả năng chia ra các phần nhỏ hơn

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

+ Barrier là 1 loại phương pháp đồng bộ. Rào cản được áp dụng cho 1 số thread hay process trong source code có nghĩa là bất kì luồng/tiến trình nào cũng phải dừng lại tại thời điểm này và không thể tiến hành cho đến khi tất cả các luồng/tiến trình khác đạt đến rào cản này.

+ Các câu lệnh thay đổi biến số chia sẻ cần được nhìn thấy bởi tất cả các tiến trình khác

- Mutex lock là gì? Cách sử dụng?

+ Mutex lock là một cấu trúc dữ liệu, được Linux kernel xây dựng theo nguyên tắc mutual exclusion, dùng để ngăn chặn race condition xảy ra trên các cấu trúc dữ liệu khác. Nói nôm na, mutex lock đảm bảo rằng: tại một thời điểm bất kì, chỉ có tối đa một thread truy cập vào critical resource.

- Test\_and\_set() hoạt động ra sao?

- Semaphore là gì? Cách sử dụng?

Là một biến số nguyên, ngoài việc khởi tạo, chỉ được truy cập thông qua hai phép toán nguyên tử tiêu chuẩn: wait() và signal().

- Monitor là gì? Cách nó hoạt động?

1. Monitor là một công cụ đồng bộ, có chức năng tương tự như semaphore nhưng dễ điều khiển hơn.

2. Monitor, là một kiểu dữ liệu trừu tượng, lưu lại các biến chia sẻ và các phương thức dùng để thao tác lên các biến chia sẻ đó. Các biến chia sẻ trong monitor chỉ có thể được truy cập bởi các phương thức được định nghĩa trong monitor.

3. Một monitor chỉ có 1 process hoạt động, tại một thời điểm bất kỳ đang xét.

🡪 Bằng cách này, monitor sẽ đảm bảo mutual exclusion và dữ liệu chia sẻ của bạn sẽ được an toàn.

- Starvation nói đến tình trạng nào trong hệ thống?

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

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

Nhiều tiến trình truy cập và thao tác đồng thời một dữ liệu và kết quả của thực thi phụ thuộc vào thứ tự cụ thể mà việc truy cập diễn ra.

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

1. Mutual exclusion: Các tiến trình ko thể thực thi trong critical sections của tiến trình khác.
2. Progress: Chỉ những tiến trình trong remainer sections mới được chọn critical sections trước.
3. Bounded waiting: Giới hạn số lần các tiến trình vào critical sections của chúng trước khi tiến trình khác được vào critical sections.

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

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

- Xem lại toàn bộ code của slide CH06.

**CHƯƠNG 7: CÁC VÍ DỤ ĐỒNG BỘ**

- Xem lại bài toán Triết gia ăn tối.

- Xem lại bài Toán Bộ ghi - Bộ đọc.

- Xem lại bài toán Producer - Consumer.

**CHƯƠNG 8: TẮC NGHẼN**

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

- Đọc thông tin của đồ thị RAG.

- 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?

- Giải thuật nhà băng.

+ Thuật toán sắp xếp việc cấp phát tài nguyên dựa vào đồ thị RAG không thể áp dụng được đối với các hệ thống có nhiều thực thể (instance) của cùng một loại tài nguyên (ví dụ R có 2 thực thể), lúc đó ta phải dùng đến một giải thuật tránh deadlock có tên là giải thuật Banker. Được gọi là giải thuật Banker là vì thuật toán có thể được sử dụng trong các hệ thống ngân hàng để đảm bảo rằng ngân hàng sẽ không chi tiền đến mức mà nó không thể đáp ứng được nhu cầu của tất cả các khách hàng.

+ Trạng thái an toàn là trạng thái mà số lượng tài nguyên còn lại (Available) của hệ thống có thể đáp ứng được tất cả các process và tạo ra 1 chuỗi an toàn.

+ Chuỗi an toàn là một thứ tự thực hiện việc cung cấp tài nguyên cho các process, với giả định rằng các process sẽ trả lại tài nguyên mà nó đang giữ sau khi thực hiện xong.

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

- 4 điều kiện xảy ra tắc nghẽn là gì?

1. Mutual exclusion: Các tiến trình dùng chung tài nguyên.

2. Hold and wait: Tiến trình giữ ít nhất một tài nguyên và đợi các tài nguyên khác giải phóng.

3. No preemption: Không ưu tiên giải phóng trước 1 phần tài nguyên tiến trình đang dùng.

4. Circular wait: Tiến trình phụ thuộc vào tài nguyên của nhau theo vòng. (Có trắc nghiệm chọn ra trường hợp có thể xảy ra deadlock)

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

+ Có 3 cách loại trừ deadlock:

1. Sử dụng giao thức để ngăn chặn hoặc tránh deadlock.
2. Cho phép hệ thông xảy ra deadlock, phát hiện, và sửa lại.
3. Bỏ qua hoàn toàn các vấn để và giả sử deadlock sẽ không xảy ra.

+ Cách thứ 3 được sử dụng nhiều nhất trong các hệ điều hành. Các phần mềm xử lí deadlock sẽ được viết sau.

- Các cách phục hồi một hệ thống tắc nghẽn.

Có 2 cách:

1. Hủy bỏ 1 hoặc nhiều tiến trình để thoát Circular wait
2. Đặt ưu tiên tài nguyên cho các tiến trình.

**CHƯƠNG 9: BỘ NHỚ CHÍNH**

- Khái niệm Cache / Memory / Register.

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

* Chia bộ nhớ logic thành các khối có cùng kích thước được gọi là trang(page).
* Bảng trang(page table) chứa địa chỉ cơ sở của mỗi trang(page) trong bộ nhớ vật lý.
* Phá vỡ bộ nhớ vật lý thành các khối có kích thước cố định được gọi là khung(frame).

- Địa chỉ luận lý, địa chỉ vật lý.

- Thanh ghi BASE, LIMIT có vai trò gì.

* Thanh gì LIMIT chỉ đinh kích thước của phạm vi.
* Thanh ghi BASE giữ địa chỉ bộ nhớ vật lý hợp lệ nhỏ nhất.

- 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.

1. Compile time:
   * + Biết vị trí tiến trình nằm trong compile time -> tạo mã tuyệt đối.
     + Nếu sau đó vị trí thay đổi -> biên dịch lại mã này.
2. Load time:
   * + Không biết vị trí tiến trình nằm trong compile time -> tạo mã có thể di dời và đợi đến load time.
     + Nếu sau đó địa chỉ thay đổi -> tải lại mã người dùng để kết hợp giá trị đã thay đổi này.
3. Excution time:
   * + Nếu tiến trình có thể được di chuyển trong quá trình thực thi của nó từ phân đoạn bộ nhớ này sang phân đoạn bộ nhớ khác -> đợi đến run time.

🡪 Hầu hết các hệ điều hành có mục đích chung sử dụng phương pháp Excution time.

- Phân trang trong Windows.

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

- Hiện tượng phân ngoại nội xảy ra khi nào? Cách khắc phục?

- Các cách chọn lổ trống khi cấp phát liên tục.

- Chia sẻ trang.

Chia sẻ mã và dữ liệu giữa các tiến trình. Thường sử dụng phân trang hoặc phân đoạn để cung cấp các gói thông tin nhỏ. Tóm lại: Chạy nhiều tiến trình với lượng bộ nhớ hạn chế.

- Bộ MMU.

- Số trang, số khung, độ dịch / offset.

- TLB là gì?

- Bảng phân trang nhiều mức.

- Bảng phân trang băm, bảng phân trang đảo ngược.

- Khái niệm về hoán đổi (Swapping).

Một tiến trình phải ở trong bộ nhớ để được thực thi. Tuy nhiên, một tiến trình có thể được hoán đổi tạm thời trong bộ nhớ sang một kho dự phòng và sau đó được đưa trở lại bộ nhớ để tiếp tục thực thi. Việc hoán đổi làm cho tổng không gian địa chỉ vật lý của tất cả các tiến trình có thể vượt quá bộ nhớ vật lý thực của hệ thống, do đó làm tăng mức độ đa chương trình trong một hệ thống.

**CHƯƠNG 10: BỘ NHỚ ẢO**

- Khái niệm bộ nhớ ảo.

Là kỹ thuật được sử dụng trong hệ điều hành của máy tính. Bộ nhớ ảo tận dụng cả phần cứng và phần mềm của máy tính để máy tính bù cho tình trạng thiếu hụt bộ nhớ vật lý, tạm thời chuyển dữ liệu từ RAM sang lưu trữ trên phần mở rộng của ổ cứng gọi là bộ nhớ ảo.

- Lỗi trang, các bước xử lý của hệ thống.

+ Lỗi trang xảy ra khi người dùng truy cập tới một trang không hợp lệ (nghĩa là trang không ở trong không gian địa chỉ của tiến trình, cụ thể là không trong bộ nhớ chính (RAM)) hoặc ta đang truy cập tới một trang hợp lệ nhưng đang trong bộ nhớ phụ (swap space)

+ Xử lý của OS : việc truy xuất 1 trang “không hợp lệ” gây ra một trap(bẫy) lỗi trang (page - fault). Phần cứng phân trang sẽ thông báo rắng là bit không hợp lệ đã được đặt vào, gây ra một trap tới hệ điều hành. Trap này kích khởi động page-fault service routine (PFSR)

+ Các bước xử lý của PFSR:

1. Chuyển process phát ra tham chiếu về trạnh thái blocked
2. Phát ra 1 yêu cầu đọc đĩa để nạp trang được tham chiếu vào 1 frame trống; trong khi đợi I/O, một proces khác được cấp CPU để thực thi.
3. Sau khi I/O hoàn tất, đĩa gây ra 1 ngắt đến hệ điều hành; PFSR cập nhật page table và chuyển process về trạng thái ready

+ Tại bước 2 của PFSR, giả sử phải thay trang vì không tìm được frame trống, PFSR được bổ sung như sau:

1. Xác định vị trí trên đĩa của trang
2. Tìm 1 frame trống:
   * + - * Nếu có fram trống thì dùng 1 giải thuật thay thế để chọn 1 page hi sinh (victim page)
         * Ghi victim page lên đĩa, cập nhật page table và frame table tương ứng
3. Đọc trang cần vào frame trống, cập nhật page table và frame table tương ứng

- Tính địa phương của tham khảo.

Những dữ liệu trong tương lai gần sẽ có thể được truy cập

- Phân trang theo yêu cầu.

- Quy ước sao chép khi ghi (Copy on Write).

- Các giải thuật chọn frame nạn nhân: FIFO, Optimal, LRU, Second Chance.

- Thay thế toàn cục / Thay thế địa phương.

- Thrashing.

**CHƯƠNG 11: LƯU TRỮ THỨ CẤP**

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

- Đặc trưng của SSD.

- Các thiết bị NVM.

- Định thời đĩa: FCFS, SCAN, C-SCAN.

- Các cấp độ RAID.