TỔNG HỢP NHỮNG CÂU HỎI LÝ THUYẾT HỆ ĐIỀU HÀNH

1. **Process là gì? Process khác chương trình ở điểm gì? Cho biết các công việc chính của bộ phận Process Management.**

          – Process là một chương trình đang thực thi. Một process thường chứa text section (program code), data section (chứa các biến toàn cục), bộ đếm chương trình, nội dung các thanh ghi của bộ xử lý.  
          – Chương trình không phải là một process, một chương trình là một thực thể thụ động, như nội dung của các tập tin được lưu trên đĩa, trái lại một quá trình là một thực thể chủ động, với một bộ đếm chương trình xác định chỉ thị lệnh tiếp theo sẽ thực thi và tập hợp tài nguyên có liên quan.  
          – Process Management:  
                    + Tạo và xóa các quá trình người dùng và hệ thống.

+ Tạm dừng và thực thi tiếp quá trình. (suspend/resume)  
                    + Cung cấp các cơ chế đồng bộ hóa quá trình. (synchronization)  
                    + Cung cấp các cơ chế giao tiếp quá trình. (interprocess communication)  
                    + Cung cấp các cơ chế quản lý Deadlock.

1. **Cho biết các công việc chính của bộ phận Memory Management, file management, secondary storage management?**

Memory Management: bộ nhớ chính là trung tâm của các thao tác và xử lý. Để nâng cao hiệu suất sử dụng CPU, hệ điều hành cần quản lý bộ nhớ thích hợp. Các nhiệm vụ của bộ phận Memorry Management:

* + Theo dõi thành phần nào của của bộ nhớ đang được sử dụng và tiến trình nào đang sử dụng
  + Quyết định tiến trình nào hoặc dữ liệu nào sẽ được di chuyển ra khỏi hoặc đưa vào bộ nhớ
  + Cấp phát và thu hồi không gian bộ nhớ khi cần thiết.

File management: bộ phận quản lý tập tin có nhiệm vụ thực hiện các hoạt động trong việc quản lý tập tin. Thực hiện các công việc chính:

* Tạo và xóa tập tin/ thư mục.
* Hỗ trợ các hàm để thao tác trên tập tin và thư mục.
* Ánh xạ các tập tin/ thư mục trên các thiết bị lưu trữ thứ cấp.
* Sao lưu và phục hồi dữ liệu.

Secondary storage management (quản lý hệ thống lưu trữ thứ cấp):

* Quản lý không gian trống trên đĩa. (free space management).
* Cấp phát không gian lưu trữ (storage allocation).
* Định thời hoạt động cho đĩa (disk scheduling).

1. **Cho biết các công việc chính của bộ phận I/O system management và của Hệ thống bảo vệ?**

I/O system management(quản lý hệ thống I/O):

Xây dựng trình quản lý thiết bị nhập/xuất nhằm che giấu sự khác biệt của các thiết bị phần cứng từ người dùng.

Bao gồm:

+ Thành phần quản lý bộ nhớ chứa vùng đệm (buffering), lưu trữ (caching) và spooling(vùng chứa).

+ Cung cấp giao diện chung đến các trình điều khiển thiết bị (device-driver interface).  
+ Bộ điều khiển các thiết bị phần cứng (device driver).

Hệ thống bảo vệ:

* Cung cấp các cơ chế đăng nhập / xuất (login / logout).
* Phân định được sự truy cập tài nguyên hợp pháp hoặc bất hợp pháp(authorized/ unauthorized).
* Phương tiện thi hành các chính sách (enforcement of policies). Chính sách: cần bảo vệ dữ liệu của ai đối với ai.

1. **System call là gì? hãy đưa ra một số ví dụ về system call. System program là gì và nó khác application program như thế nào? liệt kê các system program cơ bản.**

System call: cung cấp giao diện giữa một quá trình và hệ điều hành, ví dụ như các lệnh open, read, write file hoặc Lệnh fork() trong Unix dùng để tạo ra một tiến trình mới.. Thông thường, lời gọi hệ thống ở dạng nhị phân hay hợp ngữ. Trong các ngôn ngữ lập trình cấp cao, các lời gọi thường được định nghĩa trước, được xây dựng thành các thư viện hệ thống. Ba phương pháp thông dụng truyền tham số khi sử dụng system call:  
    + Qua thanh ghi.  
    + Qua một vùng nhớ, địa chỉ của vùng nhớ được gửi đến HĐH qua thanh ghi.  
    + Qua stack.

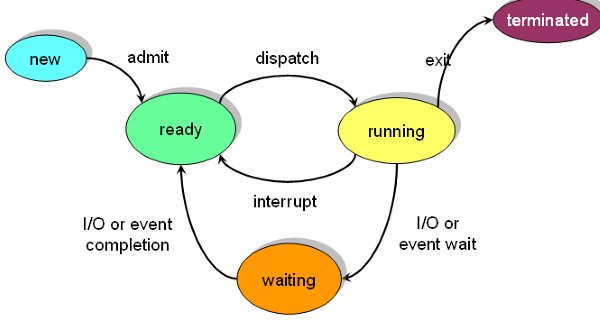
System Program là các chương trình hệ thống cung cấp môi trường thuận lợi cho việc phát triển và thực thi chương trình. Bao gồm:

+ Quản lý hệ thống file: như các tác vụ create, delete, rename, copy…  
         + Thông tin trạng thái: date, time, dung lượng bộ nhớ trống, số lượng người dùng…  
         + Soạn thảo file  
         + Hỗ trợ cho các ngôn ngữ lập trình: compiler, assembler, interpreter…  
         + Nạp và thực thi, giúp tìm lỗi chương trình: loader, debugger..  
         + Giao tiếp: như email, talk, web browser..  
 Aplication program (chương trình ứng dụng): là bất kỳ chương trình được thiết kế để thực hiện một chức năng cụ thể trực tiếp cho người sử dụng, hoặc trong một số trường hợp, cho một chương trình ứng dụng khác. Nó khác với system program ở chỗ system program cung cấp nền, môi trường cho việc phát triển và thực thi aplication program.

Ví dụ như Web Browser, Microsoft Office Word.

1. **Kernel mode là gì ? User mode là gì ? Cho biết sự khác nhau giữa chúng ?Các trạng thái của một quá trình? vẽ sơ đồ và giải thích sự chuyển đổi giữa các trạng thái?**

– Kernel mode: tác vụ được thực thi thay mặt cho hệ điều hành.  
          – User mode: là tác vụ được thực thi thay mặt cho người dùng.  
          – Khác biệt: Khi CPU ở trong Kernel mode, nó có thể thực thi bất cứ chỉ thị nào và tham chiếu đến bất kỳ địa chỉ nhớ nào. Khi CPU ở trong User mode, nó không thể truy nhập đến phần cứng hoặc tham chiếu đến bộ nhớ một cách trực tiếp. Các đoạn mã trong User mode phải được chuyển đổi thành các hàm API của hệ thống để có thể truy nhập tới bộ nhớ và phần cứng.  
          – Trạng thái của quá trình:  
                    + new: quá trình vừa được tạo ra.  
                    + ready: quá trình đã có đủ tài nguyên, chỉ còn cần CPU.  
                    + running: các lệnh của quá trình đang được thực thi  
                    + waiting (blocked): quá trình đợi I/O hoàn tất  
                    + terminated: quá trình đã kết thúc.

[](https://ry3btg.blu.livefilestore.com/y1mqd5LGjsoapMewq41vUAsWcwmhz7LBffUWKUPXl-0ghYqBVPr7TwOpBwJRi_ehxyNqEMBe8JS1qGZFHks7J3-uhErdXC7e7FImmM3kgh1ZRNszCfdP260vaUH93RBKuaVPipUEH5CbaTCceZawJU5EQ/15.bmp)

*giải thích sự chuyển đổi giữa các trạng thái????*

**« GIẢI THÍCH SƠ ĐỒ NÀY NỮA NHEN !**kiếm dùm t chỗ này nữa

Tại một thời điểm, chỉ có một tiến trình có thể nhận trạng thái *Running* trên một bộ xử lý bất kỳ. Trong khi đó, nhiều tiến trình có thể ở trạng thái Waiting hay *Ready*.

- Tất cả các tiến trình mới tạo lập đều được chuyển sang trạng thái NEW –trạng thái đầu tiên , sau đó đưa vào hàng đợi job queue, hệ điều hành sẽ sử dụng điều phối chậm (long-term scheduler) và không tiếm quyền để lựa chọn một tiến trình thích hợp nhất trong hàng đợi job queue để chuyển nó sang trang thái Ready và đưa tiến trình này vào hàng đợi Ready Queue.  
- Những tiến trình đang ở trạng thái sẵn sang Ready được sếp vào hàng đợi Ready Queue, ở trạng thái này hệ điều hành sẽ sử dụng trình điều phối nhanh(Short- Term scheduler) và có tiếm quyền để lựa chọn tiến trình thích hợp giao CPU cho nó và chuyển nó sang trạng thái Runing. Tiến trình sau khi giao CPU thì rời khỏi hàng đợi Ready queue (tiến trình đó vẫn có thể quay trở lại hàng đợi này nếu nó chưa thục thi  
xong).  
- Tiến trình được giao CPU nếu đã thực thi hoàn tất thì kết thúc và chuyển sang trạng thái terminated sau đó trả lại CPU cho hệ thống , việc chuyển trạng thái này được hệ điều hành sử dụng trình điều phối chậm không tiềm quyền .  
- Tiến trình được giao CPU nhưng chưa thực thi xong công việc và bị tiếm quyền thì sẽ bị chuyển từ trạng tháiRuning sang trạng thái Ready và quay trở về cuối hàng đợi Ready queue để đợi đến lượt giao CPU kế tiếp , tiến trình chuyển trạng thái xong trả lại CPU cho hệ điều hành .  
- Tiến trình được giao CPU nhưng chưa hoàn tất công việc của mình vì phải chờ đợi sự xuất hiện của sự kiện hay IOnào đó thì sẽ được hệ điều hành chuyển sang trạng thái waiting và chuyển nó vào hàng đợi Waiting queue , việc chuyển sang trạng thái này được hệ điều hành sử dụng trình điều phối chậm không tiếm quyền .  
- Những tiến trình ở trạng thái Waiting(nằm trong hàng đợi Waiting queue) sau khi sự kiện hay IO mà nó chờ xuất hiện thì lập tức nó được hệ điều hành chuyển nó sang trạng thái sẵn sàng Ready và đưa nó vào hàng đợi Ready queue để sẵn sàng cho lần cấp CPU kế tiếp .

1. **Tại sao các hệ điều hành hiện đại hỗ trợ môi trường đa nhiệm ? Phân biệt multitasking, multiprogramming và multiprocessing.** 
   * + - Các hệ điều hành hiện đại hỗ trợ môi trường đa nhiệm **vì**: tăng hiệu suất sử dụng CPU và tăng tốc độ xử lý.
       - Multitasking (đa nhiệm): thực hiện nhiều nhiệm vụ cùng một lúc, CPU sẽ luân phiên xử lý các tiến trình trong một khoảng thời gian rồi chuyển sang cho tiến trình khác. VD: nghe nhạc và duyệt web đồng thời.
       - Multiprogramming (đa chương): có nhiều hơn một chương trình đang nằm trong bộ nhớ để được lựa chọn chuyển giao cho CPU thực thi. Ví dụ: mở Word, Excel, Powerpoint cùng lúc, trong khi chúng ta sử dụng Word thì Excel và Powerpoint vẫn thường trực trong bộ nhớ mặc dù chúng không thực hiện việc gì cả.
       - Multiprocessing (đa tiến trình): sử dụng nhiều hơn 1 CPU để có thể xử lý song song cùng lúc nhiều tiến trình.

Như bạn đã biết, user apps chuyển từ user mode đến kernel mode khi chúng tạo ra system service call. Cho ví dụ, Windows ReadFile thậm chí cần gọi internal windows routine  để thực sự handles read data từ file. routine đó, bởi vì nó access đến các cấu trúc dữ liệu bên trong hệ thống, phải được run trong kernel mode. việc chuyển từ user mode sang kernel mode được hoàn thành bởi việc sử dụng một special processor instruction, làm cho processor switch đến kernel mode. OS traps instruction đó, chú ý răng một system service được requested, hợp lệ hóa các arguments thread passed đến system function, và sau đó thực thi internal function. Trước khi trả lại điều khiển cho user thread, processor mode được switched trở lại user mode. Theo cách này OS bảo vệ được dữ liệu của chính nó từ việc quan sát chăm chú và sự chỉnh sửa từ phía user processes.

1. **Khái niệm tiến trình được xây dựng nhằm mục đích gì ? Sự khác biệt, mối quan hệ giữa tiến trình và tiểu trình ?**

Để hỗ trợ sự đa chương, máy tính phải có khả năng thực hiện nhiều tác vụ đồng thời. Nhưng việc điều khiển nhiều hoạt động song song ở cấp độ phần cứng là rất khó khăn. Vì thế các nhà thiết kế hệ điều hành đề xuất một mô hình song song gỉa lặp bằng cách chuyển đổi bộ xử lý qua lại giữa các chương trình để duy trì hoạt động của nhiều chương trình cùng lúc, điều này tạo cảm giác có nhiều hoạt động được thực hiện đồng thời. Trong mô hình này, tất cả các phần mềm trong hệ thống được tổ chức thành một số những tiến trình (process).

Tiến trình có không gian địa chỉ và chỉ có một dòng xử lý. Các tiểu trình thì chia sẻ một không gian địa chỉ, và các dòng xử lý này hoạt động song song tương tự như các tiến trình phân biệt.

Một tiến trình có thể có nhiều tiểu trình chạy đồng thời.

Tiến trình là một bộ phận của chương trình đang thực hiện. Tiến trình là đơn vị làm việc cơ bản của hệ thống, trong hệ thống có thể tồn tại nhiều tiến trình cùng hoạt động, trong đó có cả tiến trình của hệ điều hành và tiến trình của chương trình người sử dụng. Các tiến trình này có thể hoạt động đồng thời với nhau.

Để một tiến trình đi vào trạng thái hoạt động thì hệ thống phải cung cấp đầy đủ tài nguyên cho tiến trình. Hệ thống cũng phải duy trì đủ tài nguyên cho tiến trình trong suốt quá trình hoạt động của tiến trình.   
Tiểu trình cũng là đơn vị xử lý cơ bản trong hệ thống, nó cũng xử lý tuần tự đoạn code của nó, nó cũng sở hữu một con trỏ lệnh, một tập các thanh ghi và một vùng nhớ stack riêng và các tiểu trình cũng chia sẻ thời gian xử lý của processor như các tiến trình.   
Các tiểu trình trong một tiến trình chia sẻ một không gian địa chỉ chung, điều này có nghĩa các tiểu trình có thể chia sẻ các biến toàn cục của tiến trình, có thể truy xuất đến stack của tiểu trình khác trong cùng tiến trình. Như vậy với mô hình tiểu trình, trong hệ thống có thể tồn tại nhiều dòng xử lý cùng chia sẻ một không gian địa chỉ bộ nhớ, các dòng xử lý này hoạt động song song với nhau.

1. a) Hãy đưa ra giải thích ngắn gọn cho các khái niệm sau :
   * 1. *Process, Thread, I/O bound process, CPU bound process*
   1. b) Giải thích sự khác biệt giữa Preemptive và non-preemptive scheduling. Để tối ưu hệ thống nếu chỉ quan tâm đến thông lượng (throughput) thì ta lựa chọn chế độ điều phối nào? Giải thích sự lựa chọn.

Quá trình (process) là trạng thái tức thời của một chương trình đang chạy trên máy tính. Nó bao gồm bộ nhớ cần thiết để chạy chương trình (không gian địa chỉ của quá trình) và khả năng kiểm soát hiện trạng của bộ xử lý trong quá trình thực thi chương trình (tiến trình điều khiển của quá trình). Luồng (thread) tương tự như quá trình nhưng chỉ bao gồm tiến trình điều khiển. Nhiều luồng sử dụng không gian địa chỉ của một quá trình.  
Quá trình và luồng có chung một mục đích: buộc máy tính phải làm nhiều việc hơn tại một thời điểm. Để làm điều đó, bộ xử lý (hay các bộ xử lý) phải chuyển đổi một cách trơn tru giữa các tác vụ, điều này đòi hỏi chương trình ứng dụng phải được thiết kế để chia sẻ tài nguyên máy tính.

**tính hướng xuất / nhập của tiến trình ( I/O-boundedness):**

Khi một tiến trình nhận được CPU, chủ yếu nó chỉ sử dụng CPU đến khi phát sinh một yêu cầu nhập xuất ? Hoạt động của các tiến trình như thế thường bao gồm nhiều lượt sử dụng CPU , mỗi lượt trong một thời gian khá ngắn.

**Tính hướng xử lý của tiến trình ( CPU-boundedness):**

Khi một tiến trình nhận được CPU, nó có khuynh hướng sử dụng CPU đến khi hết thời gian dành cho nó ? Hoạt động của các tiến trình như thế thường bao gồm một số ít lượt sử dụng CPU , nhưng mỗi lượt trong một thời gian đủ dài

b)Lập lịch không dừng **Non- preemtive**:

- Một khi tiến trình được cấp CPU, CPU sẽ bị chiếm dụng cho đến khi tiến trình trả lại CPU do tiến trình đó đã kết thúc, hoặc chuyển sang trạng thái đợi.

Lập lịch dừng **Preemptive:**

- Một tiến trình đang thực hiện có thể bị ngắt do yêu cầu được thực hiện của 1 tiến trình khác. Khi đó, CPU phải chịu những phí tổn do việc chuyển giữa các tiến trình gây ra – CS (CS – Context Switch chính là công việc xảy ra khi CPU chuyển giữa các tiến trình với nhau). CS thực sự là một chi phí về thời gian của hệ thống (hệ thống sẽ không làm gì ong khoảng thời gian của CS) và phụ thuộc nhiều vào sự hỗ trợ của phần cứng.

**Tại sao lập lịch không dừng không được ưu thích lập lịch cho máy tính trung tâm?**

Có lẽ là nếu sử dụng Non-Preemptive thì tổng thời gian của hệ thống sẽ lớn.:-P

Ví dụ: Tiến trình thứ nhất đang thực hiện dở, vào thời điểm i, tiến trình thứ hai xuất hiện với CPU burst nhỏ hơn CPU burst còn lại của tiến trình thứ nhất tại thời điểm đó. Nếu sử dụng lập lịch không dừng - tức là phải để tiến trình thứ nhất thực hiện xong - thì tổng thời gian để thực hiện tiến trình thứ hai sẽ lớn hơn nhiều nếu cho tiền trình thứ hai thực hiện ngay. Điều này cũng đúng với nhiều tiến trình.

=> Tổng thời gian thực hiện các tiến trình của hệ thống sẽ rất lớn.

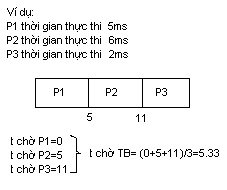
**các giải thuật định thời thông dụng:**

**a. Bộ điều phối tiến trình của hệ điều hành làm nhiệm vụ:** áp dụng một giải thuật điều phối thích hợp để chọn tiến trình thích hợp được sử dụng CPU, và bộ phân phối sẽ chuyển giao CPU cho tiến trình này.

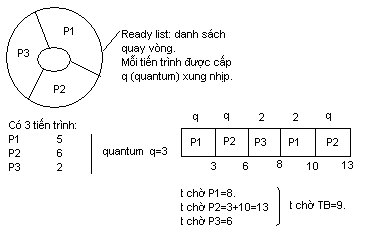
**b. Các giải thuật điều phối thông dụng:**

- FIFO (First in First out): Ready list là 1 queue.

+ Nhược điểm: nếu tiến trình chiếm ít thời gian CPU vào sau thì chờ lâu Thời gian lưu trong hệ thống không cực tiểu.



- RR (Round Robin- quay vòng):



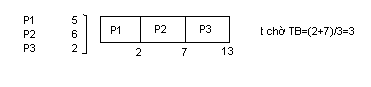
- Độ ưu tiên:

+ Mỗi tiến trình sẽ được gán 1 độ ưu tiên và tiến trình được chọn trong Ready list là tiến trình có độ ưu tiên cao.

+ Tiến trình có độ ưu tiên thấp sẽ phải chờ cho đến hết tiến trình có độ ưu tiên hơn nó Tiến trình đó gọi là hiện tượng đói CPU.

- SJF (Sorted- Job –First):

Trong giải thuật này độ ưu tiên cao nhất thì được gán cho tiến trình có thời gian thực thi ngắn nhất.



Thời gian chờ là cực tiểu trong giải thuật này nhưng thời gian xác định không rõ ràng.

- SJF và RR sau 1 q (quantum) đánh giá lại thời gian thực hiện các tiến trình trong Ready list.

- Nhiều độ ưu tiên: Do đặc điểm hoạt động của tiến trình.

+ Tiến trình hệ thống.

+ Tiến trình thao tác.

+ Tiến trình theo lô.

Các lớp cấp độ ưu tiên, ở mỗi cấp độ ưu tiên là 1 danh sách FIFO.

Áp dụng thêm giải thuật aging (“Già hoá”).

Mỗi khi trao đổi tiến trình thì 1 tiến trình ở cấp độ ưu tiên phía dưới sẽ được tăng cấp độ ưu tiên lên 1.

**KL:** Trong thực tế thì hệ điều hành sẽ tổ chức phối hợp hợp lý giữa các chiến lược chứ không phải dùng 1 cái.