**ĐỀ CƯƠNG ÔN TẬP NGUYÊN LÝ HỆ ĐIỀU HÀNH**

# Hệ điều hành là gì? Liệt kê các thành phần, chức năng của hệ điều hành

* Là chương trình trung gian giữa phần cứng máy tính và người sử dụng, có chức năng điều khiển phần cứng và cung cấp các dịch vụ cơ bản cho các ứng dụng.
* Mục tiêu:
  + - Giúp người dùng dễ dàng sử dụng hệ thống.
    - Quản lý và cấp phát tài nguyên hệ thống một cách hiệu quả.
* Các thành phần của hệ điều hành bao gồm:
  + - Bộ lập lịch tiến trình
    - Trình quản lý bộ nhớ
    - Trình quản lý I/O
    - Trình quản lý hệ thống tập tin
    - Trình quản lý giao tiếp tiến trình
    - Hệ thống bảo vệ
    - Hệ thống dịch lệnh
* Các chức năng của hệ điều hành:
  + - Phân chia thời gian xử lý trên CPU (định thời).
    - Phối hợp và đồng bộ hoạt động giữa các quá trình.
    - Quản lý tài nguyên hệ thống hiệu quả.
    - Kiểm soát quá trình truy cập, bảo vệ hệ thống.
    - Duy trì sự nhất quán của hệ thống, kiểm soát lỗi và phục hồi hệ thống khi có lỗi xảy ra.
    - Cung cấp giao diện làm việc thuận tiện cho người dùng.

# Cho biết sự khác nhau giữa Server OS và Desktop OS?

* SOS được thiết kế đặc biệt để chạy trên máy chủ, DOS để chạy trên máy tính cá nhân.
* SOS hỗ trợ nhiều dung lượng bộ nhớ hơn;
* SOS thường có các tính năng cao hơn DOS
* DOS có các tính năng hướng đến người dùng hơn
* SOS sử dụng phần cứng hiệu quả hơn đặc biệt là CPU;
* SOS cho phép nhiều kết nối mạng hơn, DOS giới hạn kết nối mạng từ 10 đến 20 kết nối;
* SOS được cấu hình cho các tác vụ ở chế độ nền, DOS chỉ tập trung cho các tác vụ ở chế độ trực diện.

# Quá trình là gì? Nêu các bước khởi tạo quá trình? Các tác vụ đối với tiến trình?

* Quá trình (process) là một chương trình đang thực thi (executing program). Một quá trình bao gồm:
  + - Trong bộ nhớ: Text Section (Program code), Data Section (khu vực dữ liệu), Heap, Stack.
    - Phần cứng: Program Counter (PC), Process Status Word (PSW), Stack Pointer (SP), Memory Management Registers.
* Các bước OS khởi tạo quá trình:
  + - Gán một định danh duy nhất (unique process identifier).
    - Cấp phát không gian nhớ cho process image.
    - Khởi tạo PCB (process control block).
    - Thiết lập các mối liên hệ cần thiết.
* Các tác vụ đối với tiến trình:
  + - Tạo quá trình.
    - Dừng quá trình.

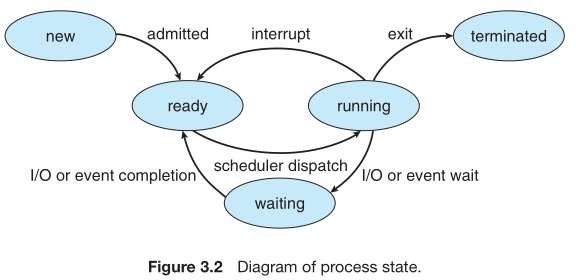
# Quá trình khác chương trình thế nào? Nhiệm vụ chính của bộ phận Process Management?

* Quá trình là một chương trình đang thực thi, sở hữu một tập con trỏ lệnh, tập các thanh ghi, các biến và một tập các tài nguyên cần cho hoạt động của nó.
* Chương trình là tập các chỉ thị điều khiển máy tính để tiến hành một hoạt động nào đó, khi được thực thi thì chương trình trở thành quá trình.
* Bộ phận Process Management có nhiệm vụ:
  + - Tạo và hủy tiến trình người dùng và tiến trình hệ thống
    - Hoãn và khôi phục tiến trình
    - Cung cấp cơ chế cho việc đồng bộ hóa tiến trình
    - Cung cấp cơ chế cho việc giao tiếp tiến trình
    - Cung cấp cơ chế quản lý deadlock.

# Trong quá trình thực thi, một process có thể có những trạng thái nào? Vẽ sơ đồ liên hệ giữa các trạng thái?

* Các trạng thái của quá trình:
  + - new: quá trình vừa được tạo
    - ready: quá trình đã có đủ tài nguyên, chỉ còn cần CPU
    - running: các lệnh của quá trình đang được thực thi
    - waiting: hay blocked, quá trình đợi I/O hoàn tất hoặc tín hiệu đồng bộ.
    - terminated: quá trình đã kết thúc.
    - Sơ đồ liên hệ giữa các trạng thái:

# Khối PCB là gì? Nêu các thông tin chứa trong khối PCB?



* Một khối điều khiển tiến trình (Process Control Block - PCB) là một cấu trúc dữ liệu được sử dụng bởi hệ điều hành máy tính để chứa các thông tin cần thiết về một quá trình, nó còn được gọi là một mô tả quá trình.
* Các thông tin chứa trong PCB:
  + - Process state. trạng thái có thể là new, ready, running, waiting, v.v
    - Program Counter. địa chỉ của lệnh tiếp theo sẽ được thực hiện cho quá trình này.
    - CPU registers. phụ thuộc vào kiến trúc máy tính. Có thể kể đến vài loại như accumulators, index registers, stack pointers, general-purpose registers, condition-code information.
    - CPU-scheduling information. gồm độ ưu tiên, con trỏ đến các hàng đợi, và các tham số của việc lập thời biểu.
    - Memory-Management Information. chứa page tables, segment tables, memory limits (giới hạn bộ nhớ).
    - Accounting information. gồm PID, lượng CPU, thời gian thực sử dụng, thời gian giới hạn v.v.
    - I/O status information. gồm danh sách các thiết bị I/O được phân bổ cho quá trình, danh sách các tệp đang mở, v.v.

# Vì sao các process khi hoạt động phải đồng bộ với nhau? Nêu các tính chất mà một lời giải cho bài toán Critical Section phải thỏa mãn.

* Các process chia sẻ tài nguyên chung đồng thời. Nếu không có sự kiểm soát khi truy cập các dữ liệu chia sẻ thì có thể dẫn đến trường hợp không nhất quán và tranh chấp dữ liệu. Vì vậy, hệ thống cần có cơ chế bảo đảm sự thực thi có trật tự của các process đồng thời.
* Một giải pháp cho bài toán Critical Section phải thoả mãn các điều kiện dưới đây:
  + - Mutual Exclusion: tại một thời điểm chỉ có 1 process được thực thi trong vùng tranh chấp.
    - Progress: nếu không có process nào đang thực thi trong vùng tranh chấp và có một số process đang chờ đợi vào vùng tranh chấp thì chỉ những process không đang thực thi trong vùng không tranh chấp mới được là ứng cử viên cho việc chọn process nào được vào vùng tranh chấp kế tiếp. Quá trình chọn lựa này không được trì hoãn vô hạn định.
    - Bounded Waiting: Thời gian chờ của mỗi process để được vào vùng tranh chấp là có hạn, không để xảy ra tình trạng đói tài nguyên (starvation).
* Để sử dụng tài nguyên, tiến trình thực hiện theo các bước sau:
  + - Yêu cầu (request): process phải chờ nếu yêu cầu không được đáp ứng ngay
    - Sử dụng (use): process sử dụng tài nguyên
    - Hoàn trả (release): process hoàn trả tài nguyên
    - Trong đó các tác vụ yêu cầu (request) và hoàn trả (release) đều là system call.
* Lời gọi hệ thống là việc một chương trình máy tính yêu cầu một dịch vụ từ nhân của hệ điều hành mà nó được thực thi. Dùng để giao tiếp và cung cấp giao diện giữa tiến trình và hệ điều hành.

# Bộ định thời là gì? Kể tên và nêu chức của các bộ định thời?

* Bộ định thời (Scheduler) sẽ lựa chọn 1 trong các tiến trình hiện có để thực thi trên CPU. Nguyên nhân là do, trong một thời điểm nhất định, chỉ duy nhất có một tiến trình được quyền ở trạng thái running mà thôi, có 3 bộ định thời bao gồm:
* Short-Term Scheduling (hay còn gọi là Dispatcher) : Dùng để định thời cho CPU
  + - Xác định process nào trong ready queue sẽ được chiếm CPU để thực thi kế tiếp.
    - Bộ định thời Short-Term sẽ được gọi mỗi khi có một trong các sự kiện/interrupt sau xảy ra :
      * Ngắt thời gian (clock interrupt).
      * Ngắt ngoại vi (I/O interrupt).
      * Lời gọi hệ thống (Operating System Call).
      * Signal.
* Medium-Term Scheduling : Dùng để định thời Swaping.
  + - Process nào được đưa vào (swap-in), đưa ra khỏi (swap-out) bộ nhớ chính.
    - Được thực hiện bởi phần quản lý bộ nhớ.
* Long-Term Scheduling (hay còn gọi là Job Scheduler) :
  + - Xác định chương trình nào được chấp nhận nạp vào hệ thống để thực thi.
    - Điều khiển mức độ multiprogramming của hệ thống.
    - Long-Term Scheduling thường cố gắng duy trì xen lẫn CPU-Bound và I/O Bound Process.

# Định thời là gì? Vì sao phải định thời? Nêu các tiêu chí định thời?

* Định thời CPU là lựa chọn process thực thi (từ ready queue) tại mỗi thời điểm trên CPU.
* Trong hệ thống tại mỗi thời điểm, chỉ có một quá trình được thực thi. Do vậy cần phải giải quyết vấn đề phân chia lựa chọn process thực thi sao cho hiệu quả nhất, đó là lý do phải định thời.
* Các tiêu chí định thời:
  + - Mức độ sử dụng CPU (CPU Utilization) : Khoảng thời gian CPU bận, từ 0% đến 100%.
    - Thông lượng (Throughput) : Số lượng process hoàn tất trong một đơn vị thời gian.
    - Thời gian hoàn thành (Turnaround time) : Thời gian để một process hoàn tất, kể từ lúc vào hệ thống (submission) đến lúc kết thúc (termination).
    - Thời gian đợi (Waiting time) : Thời gian process chờ trong hàng đợi ready.
    - Thời gian đáp ứng (Response Time) : Thời gian từ lúc tiến trình xuất hiện cho đến khi thực hiện tiến trình đó lần đầu tiên.

# Hai yếu tố của giải thuật định thời là gì ?

* Hàm chọn lựa (selection function) : Dùng để chọn process nào trong ready queue được thực thi (tuỳ thuật toán định thời sẽ có cách chọn khác nhau).
* Chế độ quyết định (decision mode) : Chọn thời điểm thực thi hàm chọn lựa để định thời. Bao gồm 2 chế độ quyết định :
  + - Non-Preemptive (Không ưu tiên) : Khi ở trạng thái running, process sẽ thực thi cho đến khi kết thúc hoặc bị blocked do yêu cầu I/O.
    - Preemptive (Ưu tiên) :
      * Process đang thực thi (running) có thể bị ngắt và chuyển về trạng thái ready
      * Chi phí cao hơn non-preemptive. Đổi lại, ta có được thời gian đáp ứng tốt hơn, vì không có trường hợp một process độc chiếm CPU quá lâu.

# Deadlock là gì? Cho ví dụ? Điều kiện tồn tại Deadlock? Cách giải quyết?

* + Deadlock là trạng thái của hệ thống mà ở đó có hai hoặc nhiều process đang cạnh tranh tài nguyên, mỗi process yêu cầu tài nguyên được giữ bởi process khác. Do vậy chúng không thể tiếp tục thực thi.
  + Ví dụ hệ thống có 2 file trên đĩa (A và B).
    - P1 đang mở (đang khoá) file A và yêu cầu file B.
    - P2 đang mở (đang khoá) file B và yêu cầu file A.
  + Điều kiện tồn tại deadlock:
    - Mutual exclusion: tại một thời điểm, mỗi tài nguyên chỉ được sử dụng bởi một process.
    - Hold and wait: một process đang giữ ít nhất một tài nguyên và đợi thêm tài nguyên do quá trình khác đang giữ.
    - No preemption: không lấy lại tài nguyên đã cấp phát cho process, trừ khi process tự hoàn trả nó.
    - Circular wait: tồn tại một chu kỳ đóng các yêu cầu tài nguyên.
  + Cách giải quyết:
    - Bảo đảm rằng hệ thống không rơi vào tình trạng deadlock bằng cách (preventing) hoặc (avoiding) deadlock.
      * Ngăn deadlock: không cho phép (ít nhất) một trong 4 điều kiện cần cho deadlock.
      * Tránh deadlock: các quá trình cần cung cấp thông tin về tài nguyên nó cần để hệ thống cấp phát tài nguyên một cách thích hợp.
    - Cho phép hệ thống vào trạng thái deadlock, nhưng sau đó phát hiện deadlock và phục hồi hệ thống.
    - Bỏ qua mọi vấn đề, xem như deadlock không bao giờ xảy ra trong hệ thống.

# Giải thích hiện tượng đói tài nguyên (starvation) trong hệ thống? Khi một process rơi vào tình trạng này thì nó có thể thực thi tiếp được không? Giải thích?

* + Đói tài nguyên (starvation) là vấn đề gặp phải trong tính toán đồng thời (concurrent computing) và là một vấn đề chính yếu của các giải thuật sử dụng độ ưu tiên, trong đó nhiệm vụ ưu tiên thấp có thể chờ đợi mãi vì luôn có một số công việc khác có mức độ ưu tiên cao hơn chiếm dụng tài nguyên mà nó cần để thực thi.
  + Khi một process rơi vào tình trạng đói tài nguyên thì nó vẫn có thể thực thi tiếp nếu hệ thống nâng mức độ ưu tiên của nó tăng dần theo thời gian đợi của nó, theo cách này, một process ưu tiên thấp cuối cùng sẽ được ưu tiên nâng lên đủ cao để nó được thực thi.
  + Một process bị đói tài nguyên mà không được xử lý thì kết quả thường thấy là nó sẽ bị mất khi hệ thống bị tắt hoặc treo. Đôi khi nó sẽ được chạy vào cuối cùng khi hệ thống rảnh rỗi.

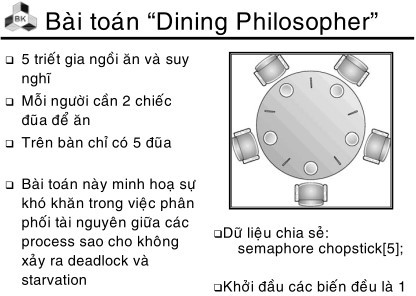
# Phân biệt Deadlock với Starvation?

* + Một tiến trình gọi là tắc nghẽn (deadlock) nếu nó đang đợi một sự kiện sẽ không bao giờ xảy ra.
    - Thông thường, có >=1 process liên quan đến việc gây ra deadlock.
    - VD : Process A yêu cầu một tài nguyên đang bị nắm giữ bởi một process B, mà B lại đang yêu cầu một tài nguyên đang bị nắm giữ bởi process A.
  + Một tiến trình gọi là trì hoãn vô hạn định (starvation) nếu nó bị trì hoãn một khoảng thời gian dài lặp đi lặp lại trong khi hệ thống đáp ứng cho những tiến trình khác. Nguyên nhân có thể do Scheduling tệ, khiến cho process có độ ưu tiên thấp không bao giờ được xử lý.
    - Thông thường trì hoãn vô hạn định nếu có xảy ra chỉ liên quan đến một (hoặc một số ít) process.
    - VD : Một process đang ở trạng thái Ready. Chuẩn bị chuyển trạng thái vào Running nhưng lại không bao giờ nhận được CPU.

# Một chương trình như thế nào gọi là bị treo? Vì sao có tình trạng trên? Nêu cách giải quyết?

* + Một chương trình bị treo khi nó ngừng đáp ứng đầu vào (không hoạt động, ngừng phản hồi, đóng băng). Trong trường hợp treo phần mềm, máy tính không bị treo mà chỉ ngừng xử lý các lệnh.
  + Có rất nhiều nguyên nhân khiến một chương trình bị treo, bao gồm các lỗi phần mềm hoặc phần cứng, chẳng hạn như tắc nghẽn (deadlock), đói tài nguyên hay trì hoãn vô hạn định (starvation), phần cứng hoạt động kém, các sự kiện bên ngoài như mạng máy tính chậm, cấu hình sai, và các vấn đề tương thích.
  + Chương trình bị treo tạm thời nếu hệ thống có thể tự khắc phục được, hoặc là vĩnh viễn và cần can thiệp thủ công bằng cách buộc chấm dứt (Force Quit) một hay nhiều chương trình. Trong trường hợp treo nghiêm trọng hơn ảnh hưởng đến toàn bộ hệ thống, giải pháp duy nhất có thể là khởi động lại máy, thường là khởi động lại máy với nút tắt/bật hoặc đặt lại.

# Nêu và đưa ra một giải pháp để giải bài toán “Dining Philosophers Problem” và phân tích ưu nhược điểm của giải pháp đưa ra?



1. **Mô tả và nêu ưu điểm, nhược điểm của từng giải thuật định thời ? FCFS, SJF, SRTF, RR.**
   * FCFS :
     + Cơ chế thực thi :
       - Tiến trình nào yêu cầu CPU trước sẽ được cấp phát trước.
       - Tiến trình sẽ thực thi đến khi kết thúc hoặc bị blocked do I/O.
     + Chế độ quyết định : Non-Preemptive. (độc quyền)
     + Hiện thực : Sử dụng hàng đợi FIFO.
       - Tiến trình đi vào được thêm vào cuối hàng đợi.
       - Tiến trình được lựa chọn để xử lý được lấy từ đầu của queue.
     + Ưu điểm :
       - Sẽ không bị starvation.
       - Thuật toán này dễ cài đặt. Code đơn giản.
     + Nhược điểm :
       - Thời gian chờ trung bình của FCFS thường khá dài (VD : Một process có burst-time rất dài đến trước, khi đó các process có burst-time nhỏ sẽ phải chờ 1 khoảng thời gian rất lâu mới đến lượt thực thi).
       - Lãng phí thời gian do thời gian phần cứng trống khá nhiều (convoy effect).
       - Non-preemptive. Sẽ không hoạt động tốt trong các hệ thống chia sẻ thời gian (time-sharing system) khi các user đều mong muốn được sử dụng CPU trong một khoảng thời gian và không muốn delay quá lâu.
   * SJF :
     + Cơ chế thực thi :
       - Định thời công việc ngắn nhất trước (Burst-time nhỏ nhất).
       - Khi CPU được tự do, nó sẽ cấp phát cho tiến trình nào yêu cầu ít thời gian nhất để kết thúc (burst-time nhỏ nhất).
       - Burst-time có được từ việc dự đoán, dựa vào các lần chạy trước của tiến trình.
       - Nếu có 2 tiến trình cùng Burst-time, tiến trình nào vào hàng đợi trước sẽ được chạy trước (không xét độ ưu tiên).
     + Chế độ quyết định : Non-Preemptive. (độc quyền)
     + Ưu điểm : Tối ưu. Cho thời gian chờ đợi trung bình tối thiểu với một tập tiến trình cho trước.
     + Nhược điểm :
       - Cần phải ước lượng thời gian cần CPU tiếp theo của process (Burst time).
       - Có thể xảy ra starvation nếu số lượng process có burst time nhỏ cần được thực thi quá nhiều.
   * SRTF :
     + Cơ chế thực thi :
       - (Tương tự SJF).
       - Nếu một tiến trình mới được đưa vào danh sách với chiều dài sử dụng CPU cho lần tiếp theo nhỏ hơn (lưu ý, chỉ nhỏ hơn, nếu burst-time bằng thì không preempt) thời gian còn lại của tiến trình đang xử lý, nó sẽ dừng hoạt động tiến trình hiện hành (preempt).
     + Chế độ quyết định : Preemptive. (không độc quyền)
     + Ưu điểm :
       - Preemptive. Thời gian đáp ứng nhanh cho các tác vụ nhỏ.
       - Tránh việc một tác vụ lớn độc chiếm CPU.
       - Thời gian chờ đợi trung bình thường sẽ nhỏ hơn SJF.
     + Nhược điểm :
       - (Các nhược điểm của SJF).
       - Tăng thời gian hoàn thành trung bình.
   * Round Robin :
     + Cơ chế hoạt động :
       - Mỗi tiến trình nhận được một đơn vị nhỏ thời gian CPU (time-slice, quantum time), thông thường từ 10-100 msec để thực thi.
       - CPU Schedulers sẽ chọn 1 tiến trình từ ready queue và “lên dây cót” một quantum cho tiến trình, sau đó cho tiến trình chạy. Lúc này, sẽ có 2 khả năng có thể xảy ra :
         * Thời gian chạy > Quantum : Khi đó, tiến trình sẽ bị interrupt và CPU Schedulers sẽ chọn tiếp tiến trình tiếp theo.
         * Thời gian chạy < Quantum : Tiến trình tiếp theo sẽ ngay lập tức được thực thi tiếp (không cần chờ hết quantum time của tiến trình trước), và tiến trình tiếp theo đó cũng được gán 1 quantum time.
       - Phụ thuộc nhiều vào quantum time :
         * Quantum time ngắn thì đáp ứng nhanh, tuy nhiên overhead lớn do chuyển ngữ cảnh nhiều. Quantum time phải > thời gian chuyển ngữ cảnh (context switch).
       - Quantum time dài thì đáp ứng chậm, tuy nhiên thông lượng (throughput) sẽ cao. Và khi quantum time quá lớn RR->FCFS (Quantum time lớn -> Không bao giờ bị ngắt -> Ai vào trước làm trước -> FCFS).
       - Khi cả tiến trình vừa thực thi xong và tiến trình mới cũng arrive vào cùng một thời điểm, thì tiến trình mới sẽ vào hàng đợi trước rồi mới đến tiến trình cũ.
       - Các tiến trình đều có độ ưu tiên giống nhau.
       - D
     + Chế độ quyết định : Preemptive.( không độc quyền)
     + Ưu điểm :
       - Thời gian đáp ứng trung bình thường thấp -> Thích hợp cho các hệ thống time-sharing.
       - Không xảy ra tình trạng starvation.
     + Nhược điểm :
       - Thời gian chờ đợi trung bình thường khá lớn.
       - Chuyển ngữ cảnh nhiều -> Hao phí cao.
       - Hiệu suất thuật toán phụ thuộc nhiều vào việc chọn quantum time.
       - Không thể sử dụng thuật toán nếu muốn các ứng dụng có độ ưu tiên khác nhau.