**TÀI LIỆU THI GIỮA KÌ HỆ ĐIỀU HÀNH**

## Chương 1: Giới thiệu về OS

### *1. Định nghĩa*

Hệ điều hành là 1 một phần mềm hệ thống. Cung cấp môi trường giao tiếp giữa các phần mềm ứng dụng và phần cứng máy tính để dễ dàng thực thi các phần mềm ứng dụng

### *2. Vai trò của OS*

-**Virtual machine(user view)**: ảo hoá phần cứng máy tính và cung cấp các khái niệm trừu tượng giúp dễ dàng thao tác với phần cứng.

-**Quản lý tài nguyên(system view)**: phân chia tài nguyên (CPU, bộ nhớ) cho các tiến tình một cách **công bằng, hiệu quả và an toàn**. Các hoạt động cấp phát và thu hồi, chia sẻ và bảo vệ.

-**Hỗ trợ giao tiếp qua lời gọi hệ thống(programer view)**: tạo điều kiện và đơn giản hoá việc lập trình ứng dụng.

### *3. Phân loại*

-**HĐH xử lí theo lô**: thực thi tuần tự các tập lệnh. Nhược: CPU sẽ nhàn rỗi trong khi đợi I/O và thiếu tương tác user.

-**HĐH đa chương(Multiprogramming)**: Trong khi tiến trình đợi I/O, CPU sẽ đổi sang cái khác. Tận dụng tối đa CPU nhưng khó lập lịch và bảo vệ bộ nhớ.

-**HĐH chia sẻ thời gian/đa nhiệm(Multitasking)**: liên tục chuyển đổi CPU cho các tiến trình khi chúng dùng hết 1 khoảng thời gian nhỏ. Phản hồi nhanh đến User tạo cảm giác các tiến trình chạy song song nhưng lập lịch CPU phức tạp hơn.

-**HĐH song song/Đa xử lý(Multiprocessing)**: chạy trên máy tính đa CPU, chia chương trình thành các tác vụ để thực thi đồng thời trên các CPU khác nhau. Tăng sức mạnh tính toán, đáng tin cậy nhưng khó trong việc hỗ trợ phần cứng và công nghệ song song.

-**HĐH phân tán(Distributed)**: Quản lí việc chia sẻ tài nguyên, công việc, tăng khả năng tính toán của nhiều hệ thống máy tính được liên kết qua đường mạng.

-**HĐH thời gian thực**: Công việc phải được hoàn thành trong điều kiện hạn chế time. Hard real-time: công việc phải hoàn thành đúng lúc, delay sẽ gây ảnh hưởng: quản lí hệ thống công nghiệp, giao thông. Soft RT: delay có thể chấp nhận được: hệ thống đa phương tiện. HĐH bị giới hạn, phức tạp và tốn tiền.

-**HĐH nhúng(Embedded):** cài đặt trên các thiết bị nhỏ như điện thoại, đồng hồ thông minh. Được thiết kế cho mục đích cụ thể, có thể có/ko có UI. Giới hạn về tài nguyên và thuật toán phức tạp.

### *4. Các thành phần chính*

-**Quản lí tiến trình và tiểu trình:** thao tác với tiến trình, tiểu trình; cơ chế liên lạc IPC; điều phối CPU; đồng bộ hoá.

-**Quản lí bộ nhớ:** cấp phát, thu hồi và bảo vệ vùng nhớ; quản lí vùng nhớ ảo khi cần thêm, các chương trình được load vào vùng nhớ từng phần.

-**Quản lí tập tin và ổ đĩa:** file là 1 khái niệm trừu tượng mà OS cung cấp để lưu trữ dữ liệu. Tổ chức file, cấp phát và thu hồi file, việc lưu trữ file trên ổ đĩa.

-**Quản lí I/O**: trung gian giữa yêu cầu I/O và thiết bị phần cứng. Giao tiếp phần cứng I/O(device controller, DMA, polling, interrupt I/O), phần mềm I/O cho phép truy cập và quản lí hoạt động I/O (device driver, interrupt handler)

-**Lời gọi hệ thống(system call):** Được cung cấp để truy cập vào các dịch vụ của OS, được gọi thông qua API (Win32, POSIX(Unix, linux, Mac), java).

-**Hệ thống cơ chế dòng lệnh:** giao diện text để tương tác với OS

-**Hệ thống bảo vệ và bảo mật:** User/chương trình phải có quyền để thao tác trên các đối tượng hay thành phần hệ thống

### *5. Kiến trúc hệ điều hành*

-**Đơn giản:** các thành phần OS không được phân chia tốt. Ứng dụng có thể truy cập phần cứng trực tiếp. Đơn giản và dễ mở rộng nhưng dễ hư hại.

-**Một khối(Monolithic)**: các thành phần được kết lại thành 1 module(vd: UNIX, MS-DOS). Mỗi thủ tục(hàm) có thể gọi những thủ tục khác. Đơn giả, nhanh nhưng khó cài đặt và bảo trì.

-**Phân tầng:** Các thành phần đc chia thành các tầng(vd: MULTICS). Mỗi tầng sử dụng dịch vụ được cung cấp bởi tầng thấp hơn liền kề. Dễ cài đặt và bảo trì hơn 1 khối. Khó ở việc gom nhóm các dịch vụ thành layer và hiệu năng khi lời gọi được truyền qua nhiều tầng.

-**Vi nhân(Microkernel)**: Kernel(nhân HĐH) chỉ chứa các thành phần quan trọng (vd: Mach). Các thành phần khác được cài đặt như chương trình user. Dễ quản lí, mở rộng và tạo mới kiến trúc, bảo mật và đáng tin cậy nhưng có vấn đề về hiệu năng.

-**Lai(Hybrid)**: sự kết hợp giữa nhiều kiến trúc để tăng hiệu năng, bảo mật và khả năng sử dụng. Được dùng trong các OS hiện đại( Linux, Mac OS X, Windows, Android)

-**Khác:** Client-Server, Virtual machine, Exokernel.

## Chương 2: Tiến trình và giao tiếp giữa các tiến trình

### *1. Khái niệm về tiến trình(Process)*

-Một tiến trình là 1 chương trình đang được thực thi

### *2. Các đặc điểm của tiến trình*

-**Trạng thái** của triến trình: New(Được tạo lập), **Ready(**nằm trong hàng đợi chờ được thực thi)**, Running(**đang được thực thi**), Blocked(**Khi đang running nhưng phải đợi tài nguyên khác, hoàn tất thì quay về Ready**)**, Exit(thu hồi tài nguyên)

-**Không gian địa chỉ** của tiến trình: Mỗi tiến trình được cấp 1 vùng nhớ trong bộ nhớ: **Stack** chứa các biến local, tham số, return values được tạo và thu hồi tự động; **Heap**: chứa biến cấp phát động; **Data**: chứa các biến Global hay Static; **Text** chứa tập lệnh.

-**Khối quản lí tiến trình(PCB):** cấu trúc dữ liệu chứa thông tin về process, nằm trong Process Table trong vùng nhớ của OS và được truy cấp trong Kernel mode. Bao gồm: **Process identifier**: số id nhận diện, **Process state**: trạng thái (running,...), **Program Counter**: địa chỉ lệnh tiếp theo sau khi được khôi phục, **CPU registers**: thông tin để reload đúng, **CPU scheduling information**: độ ưu tiên, con trỏ đến hàng đợi, **Memory management information**: định vị vùng nhớ process, **Accounting information**: thống kê và hiệu suất hệ thống, **I/O status information**: về thiết bị I/O, files,...

-**Chuyển đổi ngữ cảnh:** Quá trình OS chuyển đổi CPU giữa các tiến trình, khi chuyển đổi sẽ lưu thông tin PCB process hiện tại và load PCB của process tiếp theo.

### *3. Thao tác với các tiến trình:*

-**Tạo lập tiến trình:** Tiến trình cha có thể tạo lập các tiến trình con (fork (Unix), CreateProcess (Win)). Tiến trình con có thể là bản sao của cha(cùng vùng nhớ) hoặc load 1 chương trình khác để thực thi(exec(Unix)). Khi 1 tiến trình mới được tạo thì nó sẽ được cấp Process ID và các tài nguyên khác, 1 PCB cũng được tạo.

-**Kết thúc tiến trình:** 1 tiến trình có thể kết thúc khi nó kết thúc thực thi hoặc bị kết thúc bởi cha nó hoặc tiến trình OS (exit, ExitProcess).

**Tiến trình Zombie:** con bị kết thúc nhưng PCB chưa được giải phóng vì cha không nhận được tín hiệu -> Cha phải đợi đến khi con hoàn thành (wait, WaitForSingleObject). **Tiến trình mồ côi:** Cha bị kết thúc nhưg con vẫn tồn tại -> được gán lên root process hoặc buộc phải kết thúc.

### *4. Giao tiếp giữa các tiến trình (IPC)*

Cơ chế liên lạc cung cấp cách để các tiến tiền trình giao tiếp, trao đổi thông tin và chia sẻ với nhau.

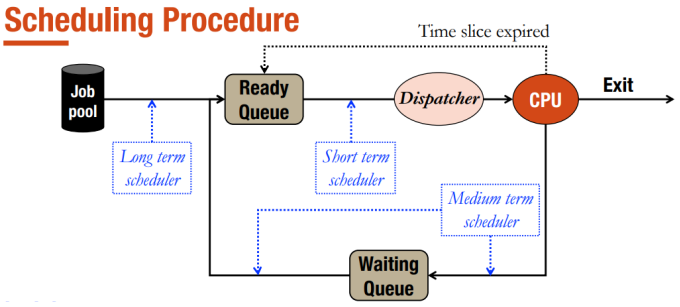
-**Pipe:** truyền dữ liệu 1 chiều giữa 2 process là writer và reader. 2 loại: Pipe ẩn danh (Ordinary) thì 2 process phải có quan hệ cha-con. Named pipe thì 2 process ko cần cha-con.

-**Vùng nhớ chia sẻ:** các tiến trình giao tiếp thông qua 1 vùng nhớ. Vùng nhớ này nằm trong vùng nhớ của tiến trình đã tạo ra nó và yêu cầu phải có cơ chế đồng bộ khi chia sẻ.
-**Socket:** Giao tiếp giữa các máy tính xa nhau dựa trên mô hình client-server. 1 socket đc định danh bởi địa chỉ IP và Port number. **UDP** (Socket không kết nối): ko yêu cầu kết nối, truyền dữ liệu bằng datagrams, có thể mất data, nhanh, ít tin cậy hơn TCP. **TCP**(socket hướng kết nối): yêu cầu kết nối, data truyền dưới dạng streams of bytes, có cơ chế truyền lại data bị mất, chậm hơn UDP và đáng tin cậy.
-**Khác:** Signal handling, Remote Control Protocol, Message Passing.

## Chương 3: Điều phối và lập lịch CPU

### 1. *Khái niệm cơ bản*

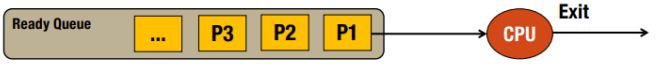
-**CPU - I/O Burst Cycle:** bao gồm CPU burst thực hiện việc tính toán và I/O Burst thực hiện thao tác nhập xuất.
-**Thủ tục lập lịch:**



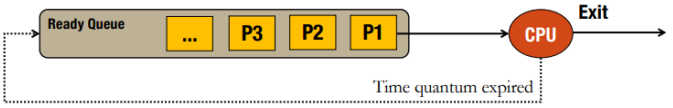

**Scheduler(Bộ định thời)**: là thành phần của OS thực hiện các hoạt động điều phối. Bao gồm:
+Long term scheduler(Job scheduler): chọn các công việc từ Job pool và tải chúng vào ready queue.
+Short term scheduler(CPU scheduler) chọn 1 process để thực thì từ ready queue.
+Medium term scheduler quản lí lựa chọn tiến trình nào được giao tài nguyên trong waiting Queue, việc vào ra WQ của các tiến trình
**Dispatcher(Bộ điều phối)** cung cấp CPU cho process được chọn bởi Short term scheduler.
-**Tiêu chí của lập lịch**: Công bằng và hiệu quả, Tối đa hoá sử dụng CPU, Cực đại hoá thông lượng(số lượng tiến trình đc hoàn thành trên 1 đvtg), Giảm đi thời gian lưu trú(Turnaround), đợi(waiting: ở trong ready queue) và hồi đáp(response: từ lúc vào đến phản hồi đầu tiên)
-**Khi nào cần lập lịch**: Tạo mới tiến trình(cha hay con được thực thi), Kết thúc tiến trình, Tiến trình bị blocked để đợi 1 thao tác I/O hay sự kiện, Ngắt(Interrupt) xảy ra hoàn thành I/O hoặc hết time slice.
-**Các loại điều phối: Điều phối độc quyền/Ko thương lượng** (Nonpreemptive), tiến trình đang chạy sẽ giữ CPU cho đến khi nó kết thúc hoặc chuyển đổi sang trạng thái waiting. **Điều phối không độc quyền/Có thương lượng** (Preemptive), tiến trình đang chạy sẽ bị ngắt bởi OS khi có tiến trình có độ ưu tiên cao hơn đến hoặc hết time slice.

### 2. *Các thuật toán điều phối*

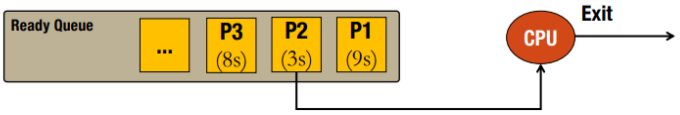
-**First-Come, First-Served (FCFS)**: độc quyền, chọn tiến trình đứng đầu tiên trong ready queue. Đơn giản nhưng hiệu năng thấp và không xem xét đến các tiến trình ưu tiên.



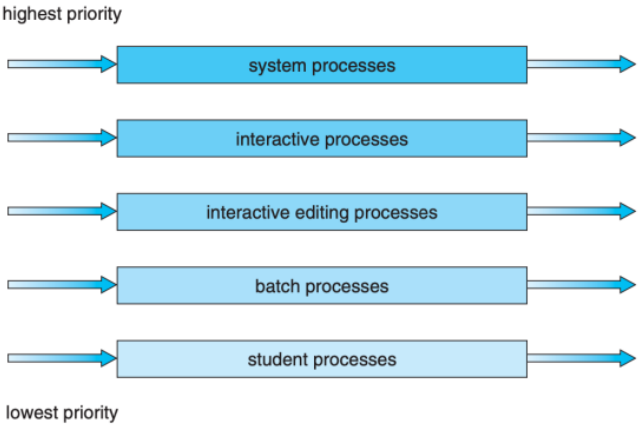

-**Round-Robin (RR)**: Không độc quyền, được dùng trong HĐH chia sẻ thời gian. Chọn tiến trình đầu tiên trong ready queue để thực thi trong 1 khoảng thời gian nhỏ q(time slice). Sau khoảng thời gian đó nếu tiến trình chưa thực hiện xong thì sẽ được chuyển cuối ready queue. Công bằng, giảm thời gian đợi cho các tiến trình ngắn nhưng không xem đến sự ưu tiên và khó trong việc chọn q cho hợp lí.




-**Shortest-Job-First (SJF)**: độc quyền
-**Shortest-Remaining-Time-Next (SRTN)**: không độc quyền
Chọn tiến trình có CPU-Burst thấp nhất. Khi có 1 tiến trình có CPU burst thấp hơn vào ready queue thì SJF sẽ tiếp tục chạy tiến trình hiện tài trong khi SRTN sẽ đổi cho tiến trình mới. Đơn giản, giảm TT và WT cho tiến trình ngắn, tăng thông lượng nhưng khó trong việc ước lượng CPU-burst cho tất cả process. Vấn đề đói CPU khi process có CPU_burst cao hơn luôn không được thực thi bởi các tiến thình nhỏ hơn.




-**Priority**: có 2 phiên bản độc quyền và không độc quyền, cũng được dùng trong hđh chia sẻ thời gian. Mỗi process được gắn với 1 độ ưu tiên. Chọn process có độ ưu tiên cao nhất trong ready queue. Tuỳ vào độc quyền hay không thì tiến trình có độ ưu tiên cao hơn đến sau có được thực thi liền hay không. Vấn đề đói CPU với các tiền trình có độ ưu tiên thấp -> giải pháp là thêm cơ chế tăng độ ưu tiên với các tiến trình ở lâu trong ready queue.
-**Multilevel Priority Queue**: Điều phối với hàng đợi đa cấp, chia ready queue thành nhiều hàng đợi với độ ưu tiên khác nhau. Mỗi queue sẽ sử dụng 1 thuật toán phù hợp. Mỗi tiến trình được gắn với 1 queue. Chọn process ở hàng đợi có độ ưu tiên cao hơn trước và việc chọn process nào trong queue phụ thuộc vào thuật toán của queue.




- **Khác:** Guaranteed scheduling, Lottery scheduling, Fair-Share scheduling