TRƯỜNG ĐẠI HỌC ĐÀ LẠT
Khoa Công nghệ thông tin

TS. Đặng Thanh Hải

Giáo trình tóm tắt

HỆ ĐIỀU HÀNH

(lưu hành nội bộ)

Đà Lạt

LỜI NÓI ĐẦU

Giáo trình " Hệ điều hành" được biên soạn theo chương trình đào tạo hệ thống tín chỉ của trường Đại Học Đà Lạt. Mục đích biên soạn giáo trình nhằm cung cấp cho sinh viên ngành Công Nghệ Thông Tin những kiến thức về hệ điều hành.

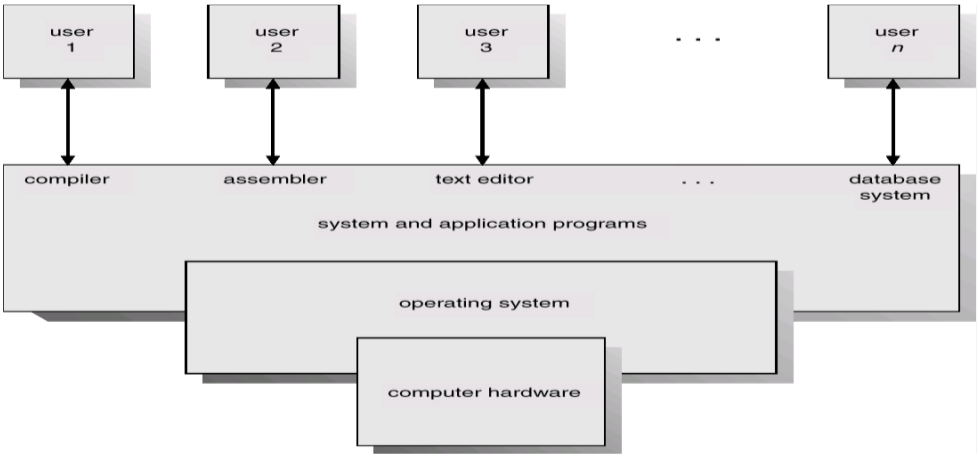
Tuy có rất nhiều cố gắng trong công tác biên soạn nhưng chắc chắn rằng giáo trình này còn nhiều thiếu sót. Chúng tôi xin trân trọng tiếp thu tất cả những ý kiến đóng góp của các bạn sinh viên, cũng như của các đồng nghiệp trong lĩnh vực này để hoàn thiện giáo trình, phục vụ tốt hơn cho việc dạy và học Công nghệ thông tin tại trường Đại Học Đà Lạt.

MỤC LỤC

CHƯƠNG I: TỔNG QUAN HỆ ĐIỀU HÀNH

I. 1 KHÁI NIỆM HỆ ĐIỀU HÀNH

- Hệ điều hành là một chương trình được xem như **trung gian** giữa người sử dụng máy tính và phần cứng máy tính với mục đích **thực hiện** các chương trình giúp cho người dùng sử dụng máy tính **dễ dàng hơn**, sử dụng phần cứng một cách có **hiệu quả.**
- Hệ điều hành là một phần quan trọng của hệ thống máy tính. Một hệ thống máy tính thường bao gồm các phần: phần cứng, hệ điều hành, các chương trình ứng dụng và người sử dụng.
  - **Phần cứng**: Bao gồm tài nguyên cơ bản của máy tính (CPU, memory, I/O devices).
  - **Hệ điều hành**: Điều khiển và kết hợp sử dụng phần cứng trong các ứng dụng khác nhau của nhiều người dùng khác nhau.
  - **Các chương trình ứng dụng**: Sẽ sử dụng tài nguyên hệ thống để giải quyết vấn đề của người sử dụng (Trình biên dịch, hệ thống cơ sở dữ liệu, games, chương trình thương mại).
  - **Người sử dụng**: Người, các máy tính khác.
- Mô hình hệ thống máy tính



- Hệ điều hành cũng có thể được xem là **bộ cấp phát tài nguyên**: Thực hiện quản lý và cấp phát tài nguyên của máy tính cho các chương trình ứng dụng.
- Hệ điều hành **Điều khiển chương trình:** Thực hiện điều khiển thực hiện các chương trình người sử dụng và các hoạt động của thiết bị nhập, xuất.
- **Hệ điều hành còn được gọi là Kernel (nhân):** Đây là các phần cốt lõi của chương trình, thường trú trong bộ nhớ, và thực hiện hầu hết các nhiệm vụ điều hành chính.

I.2 PHÂN LOẠI HỆ ĐIỀU HÀNH

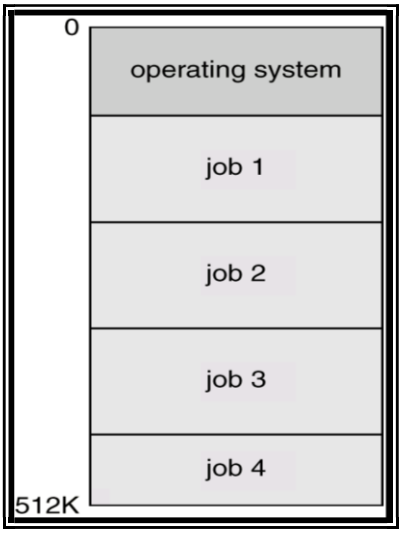
I.2.1 Hệ điều hành xử lý theo lô đơn giản

Khi thực hiện một công việc chấm dứt, hệ thống sẽ thực hiện công việc kế tiếp mà không cần sự can thiệp của người lập trình, do đó thời gian thực hiện sẽ nhanh hơn. Một chương trình gọi là bộ giám sát thường trực được thiết kế để giám sát việc thực hiện dãy công việc một cách tự động, chương trình này luôn thường trú trong bộ nhớ chính.

- Hệ điều hành theo lô thực hiện các công việc lần lượt theo những chỉ thị định trước.

I.2.2 Hệ điều hành xử lý theo lô đa chương

- Đa chương làm gia tăng khai thác CPU bằng cách tổ chức các công việc sao cho CPU luôn luôn phải trong tình trạng làm việc.
- Cách thực hiện là hệ điều hành lưu trữ một phần của các công việc ở nơi lưu trữ trong bộ nhớ. CPU sẽ lần lượt thực hiện các phần công việc này. Khi đang thực hiện, nếu có yêu cầu truy xuất thiết bị thì CPU không nghỉ mà thực hiện tiếp các công việc tiếp theo.
- Mô hình bộ nhớ cho hệ điều hành đa chương:



- Các đặc trương của hệ điều hành đa chương:
  - Việc nhập xuất phải thực hiện thường xuyên bởi hệ thống.
  - Quản lý bộ nhớ – hệ thống phải cấp phát bộ nhớ cho các công việc.
  - Lập lịch CPU – hệ thống phải chọn giữa các công việc nào thật sự được chạy.
  - Cấp phát các thiết bị.

I.2.3 Hệ điều hành đa nhiệm

- Hệ điều hành đa nhiệm là một sự mở rộng logic của hệ điều hành đa chương. Nhiều công việc cùng được thực hiện thông qua cơ chế chuyển đổi CPU như hệ đa chương nhưng thời gian mỗi lần chuyển đổi diễn ra rất nhanh.
- Hệ điều hành đa nhiệm được phát triển để cung cấp việc sử dụng bên trong của một máy tính có giá trị hơn.
- Một chương trình khi thi hành được gọi là tiến trình. Trong khi thi hành một tiến trình nó phải thực hiện các thao tác nhập xuất và trong khoảng thời gian đó CPU sẽ thi hành một tiến trình khác.
- Hệ điều hành đa nhiệm cho phép nhiều người sử dụng chia xẻ máy tính một cách đồng bộ do thời gian chuyển đổi nhanh nên họ có cảm giác là các tiến trình đang chạy được thi hành cùng lúc.
- Hệ điều hành đa nhiệm phức tạp hơn hệ điều hành đa chương và nó phải có thêm các chức năng: quản trị và bảo vệ bộ nhớ, sử dụng bộ nhớ ảo,…
- Hệ điều hành đa nhiệm hiện nay rất thong dụng.

I.2.4 Hệ điều hành tương tác

- Hệ điều hành cung cấp cơ chế truyền thông trực tiếp giữa người sử dụng và hệ thống. Khi hệ điều hành kết thúc thực hiện một lệnh, nó sẽ tìm ra lệnh kế tiếp từ người sử dụng thông qua bàn phím.
- Hệ thống cho phép người sử dụng truy cập dữ liệu và mã chương trình một cách trực tiếp.

I.2.5 Hệ điều hành giao diện bàn giấy (Desktop)

- Hệ điều hành này có cách giao diện với người sử dụng giống như một bàn làm việc, tức trên màn hình trình bày rất nhiều biểu tượng chương trình, công cụ làm việc. Hệ điều hành có đặc điểm là:

- Cài đặt trên *máy tính cá nhân* – hệ thống máy tính được thiết kế cho một người sử dụng đơn lẻ.
- Các thiết bị hỗ trợ đắc lực là thiết bị nhập xuất – bàn phím, mouse, màn hình, máy in.
- Thuận tiện cho người dùng và đáp ứng nhanh.
- Có thể kế thừa kỹ thuật để phát triển hệ điều hành lớn hơn.

- Một số hệ điều hành khác nhau sử dụng bàn giấy hiện nay (Windows, UNIX, Linux)

I.2.6 Hệ thống song song

- Ngoài các hệ thống chỉ có một bộ xử lý còn có các hệ thống có nhiều bộ xử lý cùng chia xẻ hệ thống đường truyền dữ liệu, đồng hồ, bộ nhớ và các thiết bị ngoại vi.
- Thuận lợi của hệ thống xử lý song song:
  - Xử lý nhiều công việc cùng lúc thật sự
  - Tăng độ tin cậy
- Trong hệ thống xử lý song song được thành hai loại:

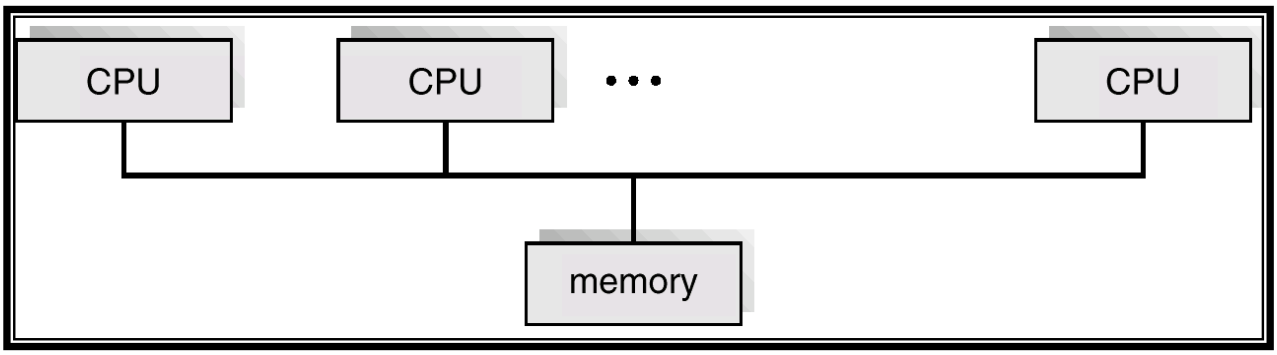
Đa xử lý đối xứng

- Mỗi bộ xử lý chạy một bản sao hệ điều hành.
- Nhiều tiến trình có thể chạy cùng lúc mà không gây hỏng.
- Hầu hết các thế hệ hệ điều hành đều hỗ trợ đa xử lý đối xứng

Đa xử lý không đối xứng

- Mỗi bộ xử lý được gắn vào một công việc cụ thể; Bộ xử lý chủ lập lịch và cấp phát công việc cho bộ xử lý phụ.
- Phổ biến nhiều trong hệ thống cực kỳ lớn.

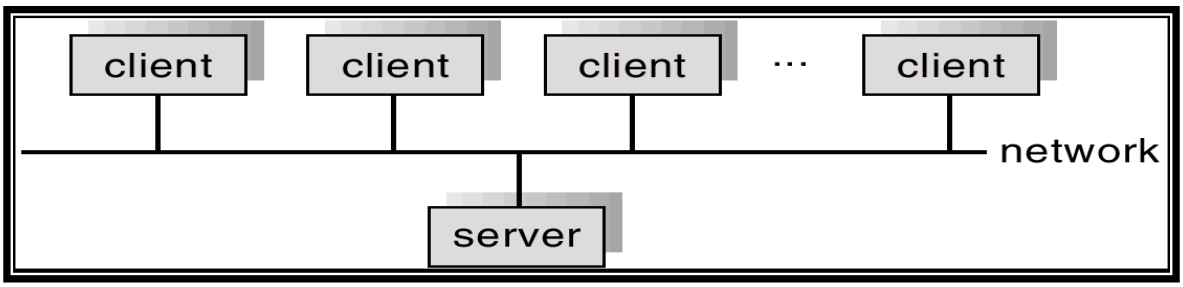
- Kiến trúc hệ thống đa bộ xử lý đối xứng:



I.2.7 Hệ thống phân tán

- Hệ thống thực hiện phân tán việc tính toán giữa các bộ xử lý .
- Mỗi bộ xử lý có vùng nhớ riêng; các bộ xử lý truyền thông với nhau qua hệ thống mạng tốc độ cao.
- Thuận lợi của hệ thống phân tán:
  - Chia xẻ tài nguyên
  - Tăng tốc độ tính toán
  - Đáng tin cậy
  - Ttruyền thông

- Trong hệ thống yêu cầu cơ sở hạ tầng về mạng. Mạng cục bộ (LAN) hoặc mạng diện rộng (WAN), cũng có thể là hệ thống client-server hoặc peer-to-peer.
- Mô hình hệ thống Client- server:



I.2.8 Hệ thống cầm tay

- Máy trợ lý cá nhân kỹ thuật số (PDAs) (personal digital assistant – PDA)
- Vấn đề cần giải quyết :
  - Bộ nhớ bị giới hạn
  - Bộ xử lý chậm
  - Màn hình hiển thị nhỏ

I.3.LỊCH SỬ PHÁT TRIỂN HỆ ĐIỀU HÀNH

- **Thế hệ 1: 1945 – 1955**

  - Năm 1940 Howard Aiken và John Von Neumam đã thành công trong việc xây dựng một máy tính dùng ống chân không.

  -Loại máy này sử dụng khoảng 1000 ống chân không, kích thước lớn nhưng khả năng xử lý chậm

  -Thời kỳ này ngôn ngữ lập trình là ngôn ngữ máy (nhị phân)

  -Việc điều hành máy, thiết kế chương trình đều do một nhóm người.

  -Năm 1950 phiếu đục lỗ ra đời và có thể viết chương trình trên phiếu đục lỗ.

- **Thế hệ 2: 1955 – 1965**
  - Thời kỳ này máy tính được chế tạo bằng thiết bị bán dẫn.
  - Công việc lập trình được thiện trên giấy bằng ngôn ngữ (assembler, fortran) sau đó được đục lỗ trên phiếu và cuối cùng đưa phiếu vào máy.
  - Hệ thống xử lý theo lô ra đời. Các công việc lưu trữ vào băng từ, chuyển điều khiển đến các công việc khác nhau được thực hiện bởi một chương trình thường trú- Đây chính là tiền thân của hệ điều hành
  - Với hệ thống máy tính này đã có sự phân biệt rõ ràng giữa người thiết kế, người xây dựng, vận hành, lập trình và bảo trì máy.

- **Thế hệ 3: 1965 – 1980**
  - Thời kỳ này máy tính được chế tạo bằng IC do đó:
    - Kích thước và giá cả máy tính giảm đáng kể
    - Máy tính trở nên phổ biến hơn
    - Các thiết bị ngoại vi dành cho máy tính càng nhiều
    - Các thao tác điều kiển máy tính ngày càng phức tạp
  - Hệ điều hành ra đời nhằm điều phối, kiểm soát hoạt động và giải quyết các yêu cầu tranh chấp thiết bị.
  - Một số hệ điều hành ra đời: MULTICS, UNIX

- **Thế hệ 4: 1980 -**
  - 1980 IBM cho ra đời máy tính cá nhân PC với hệ điều hành MS-DOS
  - Có nhiều hệ điều hành đa nhiệm, giao diện ngày càng thân thiện với người sử dụng ra đời.
  - Hiện nay hệ điều hành mạng được phát triển mạnh mẽ. (Windows, Linux)

CÂU HỎI

1. Trình bày các khái niệm hệ điều hành?
2. Trình bày khái niệm hệ điều hành đa nhiệm? Sự khác nhau giữa hệ điều hành đa chương và hệ điều hành đa nhiệm?

Ngày nay một hệ điều hành được thiết kế phải là những loại hệ điều hành nào?

CHƯƠNG II: CẤU TRÚC HỆ ĐIỀU HÀNH

II.1 CÁC THÀNH PHẦN CỦA HỆ ĐIỀU HÀNH

Hệ điều hành cung cấp một môi trường làm việc cho các chương trình thi hành. Nó cung cấp các dịch vụ cho người sử dụng, giao tiếp với người sử dụng. Các thành phần bên trong của hệ điều hành

II.1.1 Quản lý tiến trình

- ***Tiến trình:*** là một chương trình đang thực hiện. Một tiến trình cần các tài nguyên bao gồm thờI gian CPU, bộ nhớ, files, và thiết bị nhập xuất, để hoàn tất các công việc của mình.
- Vai trò của việc quản lý tiến trình trong hệ điều hành.
    - Tạo, huỷ tiến trình của người sử dụng và của hệ thống
    - Ngưng và cho phép chạy lại các tiến trình.
    - Cung cấp cơ chế :
        - Đồng bộ hóa tiến trình
        - Truyền thông giữa các tiến trình
        - Kiểm soát deadlock

II.1.2 Quản lý bộ nhớ chính

- Bộ nhớ là một dãy lớn các word hoặc byte, mỗi phần tử có một địa chỉ. Nó là nơi lưu trữ, truy xuất dữ liệu một cách nhanh.
- Bộ nhớ chính là thiết bị lưu trữ có thể thay đổi. Nó sẽ làm mất hết dữ liệu trong trường hợp hệ thống bị hỏng.
- Vai trò quản lý bộ nhớ chính trong hệ điều hành:
    - Lưu trữ thông tin các vùng nhớ hiện được sử dụng bởI ai.
    - Quyết định tiến trình nào được nạp vào bộ nhớ khi bộ nhớ có chỗ trống.
    - Cấp phát và thu hồi bộ nhớ khi cần thiết.

II.1.3 Quản lý tập tin

- Một file là một sự thu thập các thông tin có liên quan được định nghĩa bởI người tạo ra nó. Thường file thể hiện cho chương trình và dữ liệu.
- Vai trò quản lý file trong hệ điều hành:
    - Tạo và xóa file.
    - Tạo và xoá thư mục.
    - Cung cấp các thao tác trên file và thư mục.
    - Ánh xạ file vào hệ thống lưu trữ phụ.
    - Backup tập tin trên các thiết bị lưu trữ

II.1.4 Quản lý hệ thống nhập xuất

- Một trong những mục tiêu của hệ điều hành là che dấu những đặc thù của thiết bị phần cứng đối với người sử dụng thay vào đó là một lớp thân thiện hơn, người sử dụng dễ thao tác hơn.
- Một hệ thống nhập/ xuất bao gồm**:**
    - Hệ thống buffer-caching
    - Giao tiếp thiết bị
    - Bộ điều khiển cho các thiết bị phần cứng

II.1.5 Quản lý hệ thống lưu trữ phụ

- Chính vì bộ nhớ chính thường thay đổi và quá nhỏ lưu trữ tất cả dữ liệu và chương trình một cách lâu dài, hệ thống máy tính cung cấp bộ nhớ phụ để back up từ bộ nhớ chính.
- Hầu hết hệ thống máy tính ngày nay sử dụng đĩa như thành phần cơ bản lưu trữ cả chương trình và dữ liệu.
- Vai trò quản lý đĩa trong hệ điều hành:
    - Quản lý nhớ còn trống
    - Cấp phát lưu trữ
    - Lập lịch đĩa

II.1.6 Hệ thống bảo vệ

- *Bảo vệ* truy cập bởi các chương trình, các tiến trình, hoặc người sử dụng.
- Cơ chế bảo vệ phải là:
    - Phân biệt giữa cho phép hay không được phép.
    - Chỉ rõ điều khiển bị lợi dụng.
    - Cung cấp các biện pháp phải tuân thủ.

II.1.7 Hệ thống dòng lệnh

- Một trong những phần quan trọng của chương trình hệ thống trong một hệ điều hành là cơ chế **dòng lệnh**, đây là sự giao tiếp giữa người sử dụng và hệ điều hành.
- Các lệnh đưa vào hệ điều hành thong qua bộ điều khiển lệnh. Trong các hệ thống đa nhiệm một chương trình có thể đọc và thong dịch các lệnh điều khiển được thực hiện một cách tự động.
- Chức năng hệ thống dòng lệnh là lấy lệnh kế tiếp và thi hành.
- Các lệnh có quan hệ với việc tạo và quản lý các tiến trình, kiểm soát nhập xuất, quản lý bộ lưu trữ phụ, quản lý bộ nhớ chính, truy xuất hệ thống tập tin và cơ chế bảo vệ.

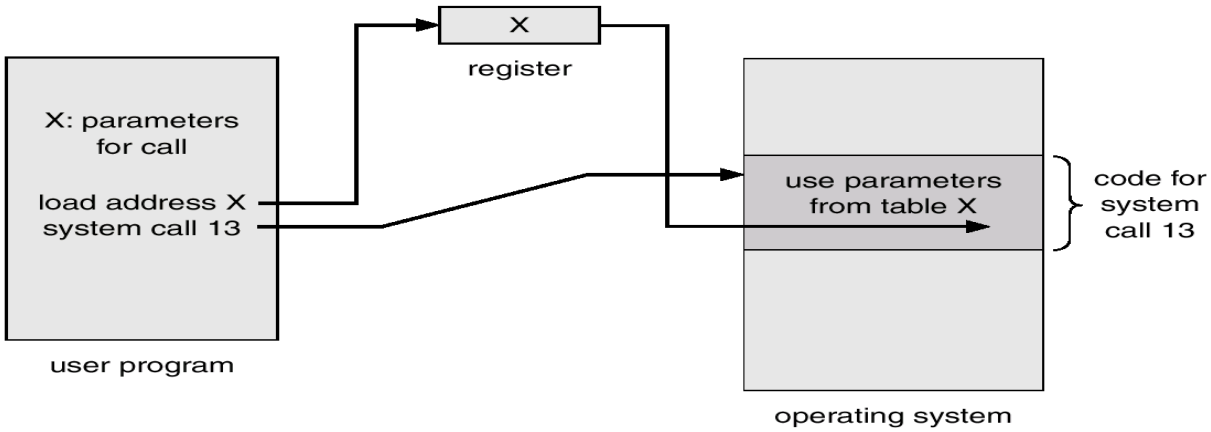
II.2 CÁC DỊCH VỤ HỆ ĐIỀU HÀNH

- Thực hiện chương trình – hệ thống có khả năng nạp một chương trình vào bộ nhớ và thi hành nó.
- Thực hiện nhập xuất – Từ chương trình người dùng không thể thực hiện nhập xuất trực tiếp, hệ điều hành phải cung cấp các cách thức để thực hiện nhập xuất.
- Các thao tác trên hệ thống file – chương trình có khả năng đọc, ghi, tạo và xoá file.
- Truyền thông – Trao đổi thông tin giữa các tiến trình đang thực hiện cùng lúc trên máy tính hay trên các hệ thống trên mạng. Thực hiện bằng cách thông qua bộ nhớ dùng chung hay qua các thông điệp.

- Phát hiện lỗi – bảo đảm phát hiện lỗi trong CPU, bộ nhớ, thiết bị nhập xuất hoặc trong chương trình người sử dụng

II.3 LỜI GỌI HỆ THỐNG

- Lời gọi hệ thống là giao diện giữa chương trình đang chạy và hệ điều hành. Thông thường là các chỉ thị bằng ngôn ngữ assembler.
- Có ba phương pháp được sử dụng truyền tham số giữa chương trình đang chạy và hệ điều hành.
    - Truyền tham số qua các thanh ghi.
    - Lưu trữ các tham số trong một bảng trong bộ nhớ và địa chỉ của bảng được truyền qua tham số vào thanh ghi.
    - Các chương trình thực hiện *Push* các tham số vào stack và được pop bởi hệ điều hành.



- Các loại lời gọi hệ thống:
    - Điều khiển tiến trình
    - Quản lý file
    - Quản lý thiết bị
    - Truyền thông

II.4 CHƯƠNG TRÌNH HỆ THỐNG

Hệ điều hành là một tập hợp các chương trình hệ thống. Chương trình hệ thống cung cấp một môi trường thuận tiện cho việc phát triển và thực hiện chương trình. Chúng được chia thành:

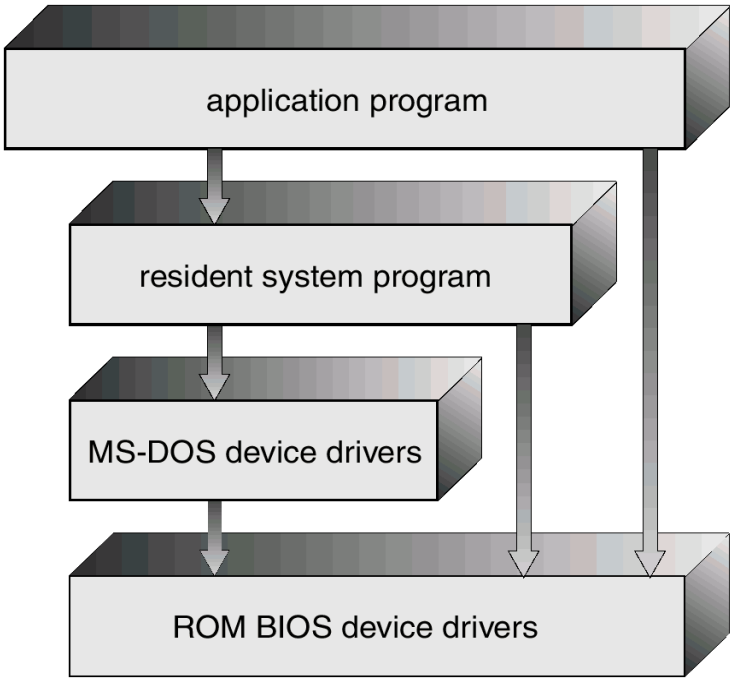
- **Thao tác trên file**: các chương trình này tạo, xoá, sao chép, in, liệt kê và các thao tác tổng quát trên tập tin và thư mục.
- **Thông tin các trạng thái:** Cung cấp các thông tin về ngày, giờ, khối lượng bộ nhớ hoặc dung lượng đĩa, số lượng người dung, hoặc thong tin về trạng thái. Những thông tin này được định dạng và xuất trên các thiết bị xuất như terminal hay tập tin.
- **Mô tả tập tin:** Một số trình soạn thảo văn bản bao gồm việc tạo và mô tả tập tin lưu trên đĩa.
- **Hỗ trợ ngôn ngữ lập trình:** Chương trình dịch, hợp ngữ, và thông dịch cho một số ngôn ngữ lập trình. Các chương trình này có thể được cung cấp chung với hệ điều hành hay là một phần riêng.
- **Nạp và thực hiện chương trình:** Hệ thống cung cấp các bộ nạp, định vị, liên kết ngoài ra còn cung cấp chức năng debug.
- **Truyền thông:** Hệ thống cung cấp cơ chế tạo sự kết nối ảo giữa các tiến trình, người sử dụng, và hệ thống máy tính khác.
- **Chương trình ứng dụng:** Thông thường hệ điều hành kèm theo một số chương trình ứng dụng như định dạng, sao chép đĩa,…

II.5 CẤU TRÚC HỆ THỐNG

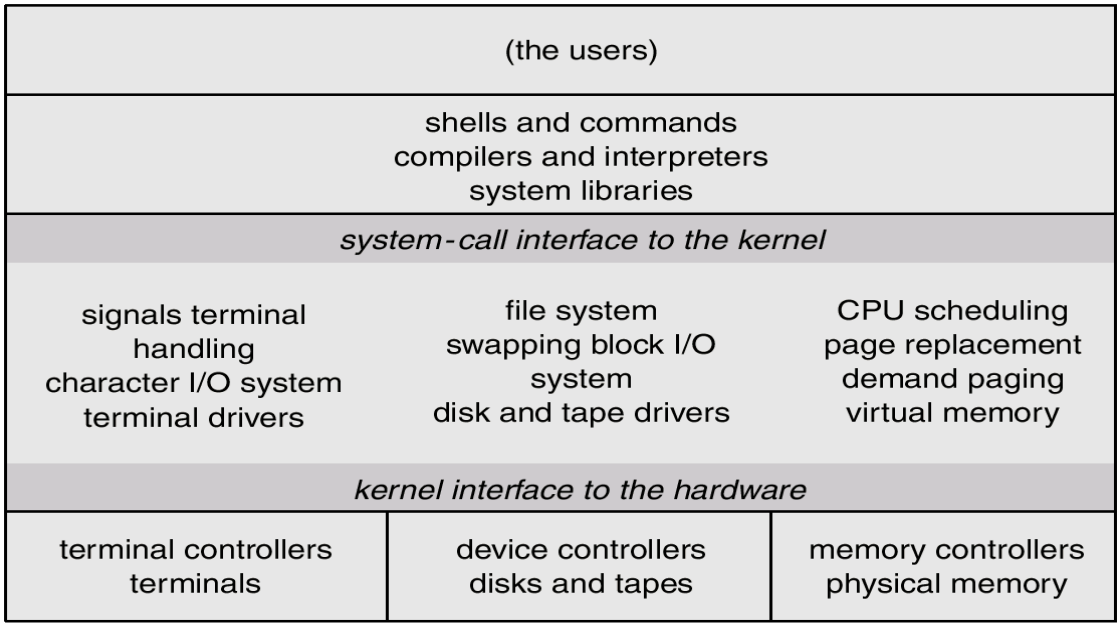
II.5.1 Cấu trúc đơn giản

- Các hệ điều hành đơn giản thường không có cấu trúc được định nghĩa tốt, thường bắt đầu từ một hệ thống nhỏ, đơn giản và có giới hạn.

- MS-DOS là một hệ điều hành có cấu đơn giản, nó cung cấp những chức năng cần thiết nhất trong một không gian nhỏ nhất do sự giới hạn của phần cứng và không chia thành những đơn thể rõ rệt
- Các chương trình ứng dụng có thể truy cập trực tiếp các thủ tục nhập xuất cơ bản và ghi trực tiếp lên màn hình hay bộ điều khiển đĩa.



- Hệ điều hành Unix bao gồm hai phần: hạt nhân và các chương trình hệ thống. Hạt nhân (kernel) được chia thành một chuỗi giao tiếp và driver thiết bị .
- Những gì dưới lời gọi hệ thống và bên trên phần cứng là hạt nhân (kernel)
- Hạt nhân cung cấp hệ thống tập tin, lập lịch CPU, quản trị bộ nhớ và các chức năng của hệ điều hành khác thông qua lời gọi hệ thống.



II.5.2 Cấu trúc theo lớp

- Các phiên bản mới của Unix được thiết kế sử dụng phần cứng phức tạp hơn, do đó hệ điều hành được chia thành nhiều phần nhỏ hơn
- Việc chia hệ thống thành nhiều phần nhỏ nó che dấu thông tin, không cho chương trình của người sử dụng có thể cài đặt những hàm truy xuất cấp thấp, thay vào đó là các lớp giao tiếp bên trong.
- Hệ điều hành chia thành nhiều lớp. Lớp dưới cùng là phần cứng, lớp trên cùng là giao tiếp với người sử dụng.
- Một lớp của hệ điều hành bao gồm một số cấu trúc dữ liệu và các hàm có thể được gọi bởi lớp ở phía trên và bản than nó gọi những chức năng của lớp bên dưới. Mỗi lớp cài đặt chỉ sử dụng những thao tác do lớp dưới cung cấp.

Lớp 6: Chương trình của người sử dụng

Lớp 5: Driver thiết bị và bộ lập lịch

Lớp 4: Bộ nhớ ảo

Lớp 3: Kênh nhập xuất

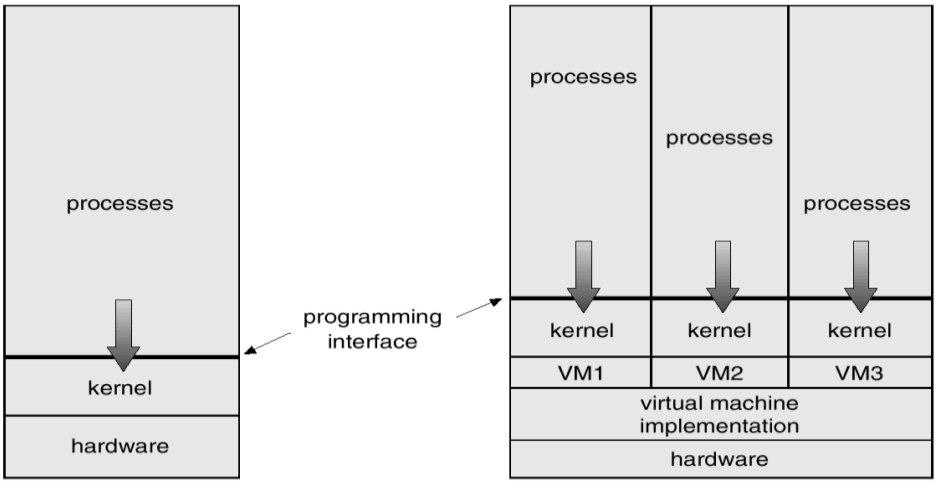
Lớp 2: Lập lịch CPU

Lớp 1: Thông dịch các chỉ thị

Lớp 0: Phần cứng

II.6 MÁY ẢO

- Một máy ảo cung cấp một giao diện giống hệt các lớp phần cứng.
- Hệ điều hành tạo ra các tiến trình ảo, mỗI việc thực hiện trên bộ xử lý vớI bộ nhớ ảo của nó.
- Tài nguyên của máy tính thật được chia xẻ để tạo ra máy ảo.
- Lập lịch CPU cũng được tạo ra như là ngườI sử dụng có bộ xử lý riêng.
- Spooling và hệ thống file được cung cấp một card reader ảo và một line máy in ảo.



- Ưu và khuyết điểm của máy ảo:
  - Khái niện máy ảo đưa ra chế độ bảo vệ tài nguyên hệ thống hoàn chỉnh từ các máy ảo khác. Các máy ảo là độc lập nhau không trực tiếp chia xẻ tài nguyên.

- Một hệ thống máy ảo là một phương tiện hoàn chỉnh để nghiên cứu hệ điều hành và phát triển nó. Các hệ thống được phát triển trên máy ảo thay vì trên máy thật bởi vậy hệ thống trên máy thật không bị phá vỡ.
- Khái niệm máy ảo cũng có khó thực hiện các yêu cầu trong bảng sao chính xác như trên máy thật.

II.7 QUÁ TRÌNH NẠP HỆ ĐIỀU HÀNH

Hiện nay máy tính được sử dụng 2 chuẩn BIOS (Base Input Output System) và UEFI (Unified Extensible Firmware Interface) cho việc quản lý máy tính và quá trình nạp hệ điều hành.

- Theo chuẩn BIOS chương trình quản lý chỉ dùng chức năng xử lý 16-bit và địa hóa bộ nhớ là 1MB. Tương ứng với chuẩn này, MBR (Master Boot Record) được dùng quản lý các phân vùng đĩa cứng. MBR chỉ cho phép quản lý 4 phân vùng đĩa cứng cơ bản và dung lượng đĩa cứng tối đa 2TB.
- Chuẩn UEFI được phát triển bởi Intel và dùng chức năng xử lý 32 bit và 64 bit địa hóa bộ nhớ lớn nên xử lý được nhiều việc hơn, ngoài ra chúng còn dùng cấp phát driver cho các thiết bị độc lập.
- UEFI dùng bảng phân vùng GPT (Globally Unique ID partition Table) quản lý phân vùng đĩa cứng lên tới 128 phân vùng và cho phép quản lý ổ cứng lên tới 9.4 ZB.

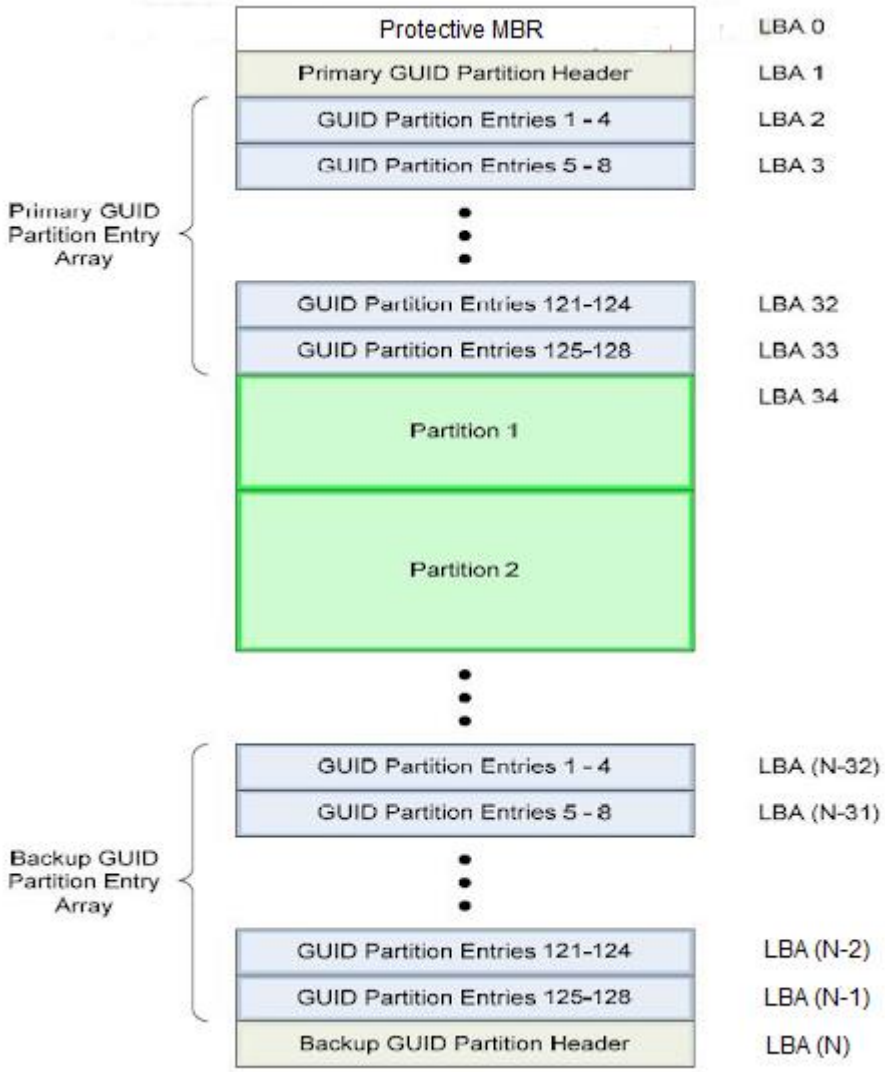
II.7.1Chuẩn BIOS

- Khi bật máy, c*hương trình Bootstrap* – (là đoạn mã lưu trữ trong ROM) được thi hành để kiểm tra các thiết bị máy tính có hoạt động tốt không. Nếu mọi thiết bị đầu sẵn sàng thì chương trình này đọc bootsector (đĩa mềm) hay masterboot (đĩa cứng) vào bộ nhớ tại địa chỉ 0:7C00h và trao quyền điều khiển tại đây.
- Trong bootsector bao gồm bảng tham số đĩa để mô tả tổ chức vật lý và tổ chức logic trên đĩa và một chương trình mồi hệ điều hành.

- Từ đó chương trình mồi hệ điều hành trong bootsector sẽ nạp các phần còn lại của hệ điều hành (kernel) vào bộ nhớ và hệ điều hành bắt đầu hoạt động.
- Đối với đĩa cứng thì có dung lượng lớn, do đó một hệ điều hành có thể không sử dụng hết dung lượng trên đĩa. Một đĩa cứng có thể chứa đồng thời nhiều hệ điều hành, mỗi hệ điều hành được lưu trữ mỗi phân vùng riêng biệt và mỗi phân vùng gọi là partition.
- Masterboot là sector đầu tiên trên đĩa cứng vật lý và bao gồm một bảng mô tả các partition hiện có trên đĩa như là sector bắt đầu, sector kết thúc, thuộc tính của các partition tương ứng. Ngoài ra masterboot còn chứa một đoạn chương trình thực hiện đọc bảng tham số partition để xác định partition active. Từ đó nạp bootsector của partition active đó vào bộ nhớ và chuyển quyền điều khiển cho chương trình mồi hệ điều hành trong bootsector.

II.7.1Chuẩn UEFI

- UEFI là một hệ điều hành tối giản nằm phía trên phần cứng và firmware của máy tính. Thay vì được lưu trong firmware, như là BIOS, chương trình UEFI được lưu trữ ở thư mục /EFI/
- Theo lược đồ miêu tả, có 1 primary GPT ở đầu và 1 secondary GPT ở cuối. Đây là những gì làm cho GPT hữu ích hơn MBR. GPT lưu trữ và backup header và parttion table vào cuối ổ đĩa để nó có thể được phục hồi nếu primary table bị hỏng. Nó cũng thực hiện checksum bằng CRC32 để phát hiện lỗi và những sai hỏng của header và patition table.
- Một ổ đĩa được chia ra làm nhiều LBA (Logical Block Addressing). Thông thường, một LBA có kích thước là 512 byte, tuy nhiên kích thước có thể thay đổi lên đến 1024 byte hoặc 2048 byte.
    - LBA đầu tiên sẽ có cấu trúc giống một ổ đĩa dạng MBR, nhằm giúp các phần mềm dựa trên MBR có thể "hiểu" được GPT nhằm tránh ghi đè.
    - LBA 1 sẽ gồm các header chứa GUID và thông tin về dung lượng, vị trí phân vùng.
    - Các LBA tiếp theo (2-33) chứa các GUID tương ứng với các phân vùng.
    - Một phiên bản của các LBA 1-33 sẽ được sao lưu ở vùng dữ liệu cuối của ổ đĩa.



- Các phân vùng sẽ nằm sau LBA33, số lượng phân vùng trên lý thuyết có thể đạt đến vô hạn. Mỗi phân vùng sẽ được gán một GUID (Globally Unique Identifier) để đảm bảo tính duy nhất của các phân vùng.
- GPT chỉ cho phép dùng **Windows 64 bit** từ Windows 7 trở về sau này. Đối với Windows 7 trở về trước thì thường dùng MBR thay cho GPT

CÂU HỎI

1. Khi xây dựng một hệ điều hành phải bao gồm những thành phần nào? Trình bày các chức năng của các thành phần đó? Liệt kê các thành phần không thể thiếu được của một hệ điều hành đa nhiệm?

2. Các chương trình ứng dụng giao tiếp với hệ điều hành thông qua thành phần nào của hệ điều hành?
3. Lời gọi hệ thống được cài đặt trong hệ điều hành bằng kỹ thuật gì? Việc truyền tham số cho lời gọi hệ thống bằng kỹ thuật nào? Trình bày kỹ thuật đó cho việc truyền tham trị và tham biến?
4. Trình bày quá trình hệ điều hành được khởi động trên một hệ thống máy tính? Nếu sector đầu tiên trên ổ đĩa bị hỏng thì việc cài đặt hệ điều hành lên ổ đĩa đó có được thực hiện không? Giải thích?
5. Trên một ổ đĩa cứng có thể thể cài đặt được nhiều hệ điều hành không? Vì sao?
6. Vì sao có loại virus được gọi là B-Virus? Các biện pháp phòng và diệt loại B-Virus?
7. Cài đặt phần mềm máy ảo *VMware workstation*?

CHƯƠNG III: GIỚI THIỆU MỘT SỐ HỆ ĐIỀU HÀNH

III.1 HỆ ĐIỀU HÀNH MS-DOS

III.1.1 Giới thiệu

- MS-DOS là một hệ điều hành đầu tiên chạy trên máy PC và được thiết kế bởi Microsoft
- MS-DOS là hệ điều hành đơn nhiệm, một người 24ung , và có cấu trúc đơn giản nên yêu cầu cấu hình máy thấp, bộ nhớ chính 640KB, giao diện theo cơ chế dòng lệnh
- MS-DOS có thể được cài đặt trên đĩa mềm hoặc đĩa cứng

III.1.2 Cấu trúc hệ điều hành MS-DOS

- Tổ chức của MS-DOS bao gồm:
  - Chương trình mồi hệ điểu hành được nạp vào bootsertor
  - Chương trình shell: giao tiếp giữa người sử dụng và hệ điều hành.
  - Chương trình chứa các chức năng cơ bản của hệ điều hành
  - Chương trình nhập xuất
  - Hệ thống các chương trình tiện ích.

- Cấu trúc bộ nhớ

<table>
<tr><td rowspan="2">1024KB- 4GB</td><td>Extended memory area</td></tr>
<tr><td>High memory area (64KB)</td></tr>
<tr><td>640-1024KB</td><td>Upper memory area</td></tr>
<tr><td>0-640KB</td><td>Conventional Memory</td></tr>
</table>

- Nội dung hệ điều hành
  - IO.SYS : hệ thống nhập xuất
  - MSDOS.SYS: hệ thống tập tin, giao tiếp dòng lệnh
  - CONFIG.SYS: cài đặt driver thiết bị
  - COMMAND.COM : tập lệnh nội trú (internal)
  - AUTOEXEC.BAT : chứa tập lệnh DOS chạy tự động khi hệ điều hành bắt đầu

III.1.3 Lịch sử phát triển

- Năm 1980 IBM sản xuất máy tính cá nhân đầu tiên với bộ xử lý 8088, 16 bit và yêu cầu Microsoft thiết kế một hệ điều hành. MS-DOS ra đời 1981
- Version 1.0 ra đời 8/1981 bao gồm 4000 dòng hợp ngữ.
  - Quản lý 12K bộ nhớ
  - Được tổ chức thành 3 tập tin : IBMIO.COM- chứa hệ thống nhập xuất, IBMMS-DOS chứa hệ thống tập tin trên đĩa, chương trình giao tiếp, COMMAND.COM chứa các lệnh xử lý.
  - Được cài đặt trên đĩa mềm hai mặt 320KB

- Version 2.0 : ra đời 3/1983 cung cấp hệ thống tập tin cấp bậc
  - Sử dụng đĩa cứng 10MB
  - Cho phép cài đặt các bộ điều khiển thiết bị trong tập tin CONFIG.SYS

- Version 3.0 : ra đời 8/1984 quản lý được đĩa cứng 20MB, đĩa mềm 1.2MB, sử dụng đĩa ảo – lấy RAM làm đĩa ảo
- Version 4.0 : ra đời 7/1988 hỗ trợ đĩa cứng lớn hơn 32MB đến 2GB, sử dụng vùng nhớ mở rộng đĩa ảo

- Version 5.0 : ra đời 4/1991 tận dụng bộ nhớ mở rộng 26ung để lưu trữ các device driver.
- Version 6.0: 1993 có các đặc điểm:
    - Tăng dung lượng đĩa sử dụng DBLSPACE
    - Tạo bộ nhớ cho đĩa với SmartDrv
    - Tối ưu bộ nhớ dùng Memmaker
    - Cứu tập tin bằng tiện ích Undelete
    - Kiểm tra đĩa bằng Scandisk

III.1.4 Cài đặt hệ điều hành

- Khởi động hệ điều hành DOS và thực hiện lệnh Format ổ đĩa: /s
- Hoặc Sys ổ đĩa:

III.1.5 Tập lệnh

- Lệnh thông tin hệ thống
    - DATE: Thiết lập hoặc hiển thị ngày hệ thống
    - TIME: Thiết lập hoặc hiển thị giờ hệ thống
    - PROMPT: Định nghĩa dấu nhắc hệ thống
    - SET: Định nghĩa biến môi trường
    - VER: Hiển thị phiên bản hệ điều hành

- Lệnh làm việc với đĩa
    - DISKCOPY: Sao chép đĩa mềm
    - FORMAT: Định dạng đĩa
    - LABEL: xem, đặt nhã đĩa
    - VOL: Hiển thị nhãn và số serial của đĩa
- Lệnh làm việc với thư mục

- Ổ đĩa:↵ Chuyển ổ đĩa hiện hành
- CHDIR hoặc CD : chuyển thư mục hiện hành
- DIR: hiển thị nội dung thư mục
- MKDIR hoặc MD: tạo thư mục mới
- PATH: Định nghĩa đường dẫn
- RMDIR hoặc RD: xoá thư mục khác rỗng
- TREE: Hiển thị cấu trúc cây thư mục

- **Thiết lập môi trường**
  - BUFFERS: Mô tả số buffer đĩa
  - DEVICE: Cài đặt device driver
  - FILES: Số tập tin tối đa được mô tả
  - LASTDRIVER: Số ổ đĩa tối đa

- **Sử dụng tập tin Batch**
  - CALL : Gọi tập tin batch khác với tham số
  - ECHO: Hiển thị thông điệp lên màn hình
  - FOR: Thực hiện lệnh DOS nhiều lần
  - GOTO: Nhảy đến một vị trí thi hành lệnh khác trong tập tin batch
  - IF: Kiểm tra điều kiện thi hành lệnh batch
  - PAUSE: Dừng lệnh bacth
  - REM: Không thi hành lệnh bacth

- **Ví dụ tạo ra đĩa khởi động cài đặt driver CDROM**

- Copy các tập tin himem.sys, oakcdrom.sys, aspi2dos.sys, mscdex.exe, guest.exe, smartdrv.exe vào đĩa mềm.
- Tạo tập tin config.sys

[memu]

menuitem=NONE, Khoi dong khong co CDROM

Menuitem=CDROM, Khoi dong co CDROM

Menuitem=USB, Khoi dong co USB

[common]

Device =himem.sys /testm:off

Lastdrive=z

[CDROM]

Device=oakcdrom.sys /d:mscd001

[USB]

Device=aspi2dos.sys /int/all

[NONE]

- **Tạo tập tin autoexec.bat**

@echo off

Goto %config%

:CDROM

Mscdex.exe /d:mscd001

Goto end

:USB

Guest.exe

Goto end

:NONE

:END

Smartdrv.exe

III.2 HỆ ĐIỀU HÀNH WINDOWS

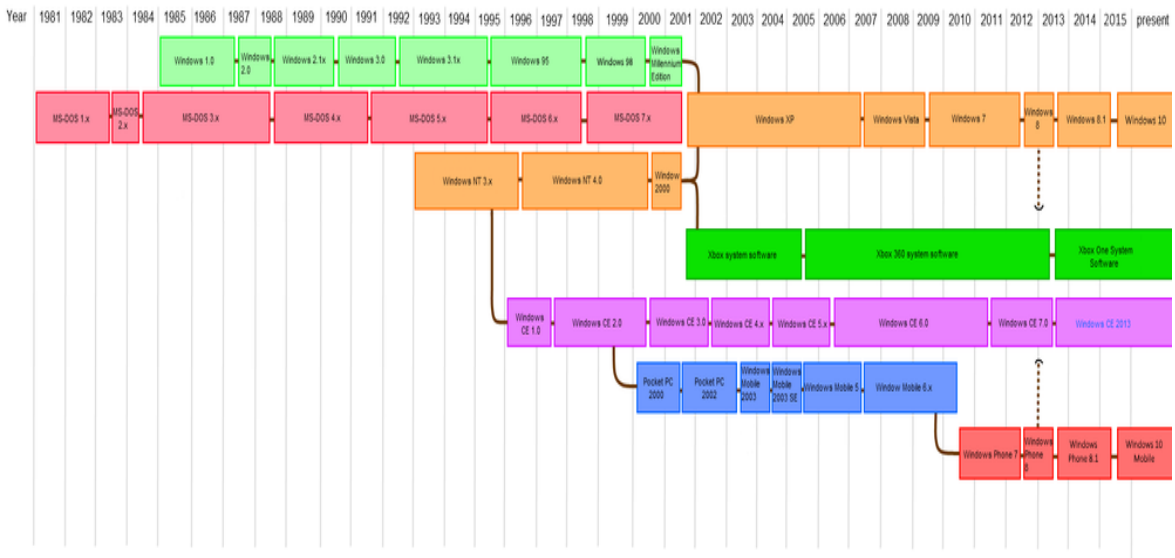
III.2.1 Giới thiệu

- Windows được thiết kế bởi Microsoft.
- Windows là hệ điều hành đa nhiệm, nhiều người 29ung.
- Giao diện người dung thân thiện, chế độ đồ họa
- Cài đặt và thay đổi cấu hình hệ thống dễ dàng, khái niệm Plug and play
- Có tính ổn định cao, nếu có tiến trình nào bị hỏng thì hệ thống huỷ bỏ tiến trình đó mà không ảnh hưởng đến toàn bộ hệ thống.
- Có tính bảo mật cao.

III.2.2 Lịch sử phát triển

- 11/1983 Microsoft tuyên bố sự ra đời hệ điều hành Windows
- 11/1987 version 2.0 ra đời có sự thay đổi về giao diện, cửa sổ có thể chồng lên nhau, menu, dialog box
- 5/1990 version 3.0 cho phép truy cập đến bộ nhớ 16Mb bộ nhớ. Có Program Manager, Task Manager, File Manager.
- 4/1992 version 3.1 hỗ trợ Multimedia(sound& music). Windows trở thành HĐH chiến lược của Microsoft.
- 9/1995 Windows 95. Độc lập thiết bị, hỗ trợ CD-Rom, kỹ thuật Plug and play, hỗ trợ mạng cục bộ, truy xuất từ xa. Cải tiến giao diện đồ họa.

- 1998 Windows 98 ra đời với hệ thống tập tin FAT32, hỗ trợ tên tập tin dài. Tiếp theo WindowsMe được tích hợp rất nhiều driver thiết bị
- 2000 Windows 2K ra đời là sự lai ghép Windows 98 và Windows NT hỗ 2 hệ thống tập tin FAT32 và NTFS. Hỗ trợ tính bảo mật trên hệ thống tập tin NTFS. Hệ điều hành mạng Client/server. Có 2 phiên bản Windows 2K server và professional
- 2001 : Windows Xp ra đời với cải tiến giao diện. Hỗ trợ mạng Internet.
- Các phiên bản hệ điều hành Windows:



III.2.3 Các tiện ích của Windows

- Quản lý các driver thiết bị gắn vào máy tính: **Device Manager**
- Xem cấu hình máy tính: **dxdiag**
- Xem ,cài đặt các thông số máy tính: **control panel**
- Khởi động các dịch vụ HĐH: **Administrator tool – Service**
- Cài đặt thêm các thành phần HĐH, gỡ bỏ các chương trình: **Administrator Tool- Add or remove programs**

- Quản lý đĩa: **Administrator Tool – Computer Management – Disk management**
- Tạo tài khoản người 31ung: **Administrator Tool – Computer Management-Local users and Groups**
- Xem huỷ các tiến trình đang chạy : **Task manager**
- Bảo mật folder trên hệ thống NTFS
- Chia xẻ folder

III.3 HỆ ĐIỀU HÀNH LINUX

III.3.1 Đặc điểm

- Multi: Tasking, Threading, User
- Multi-platform: Chạy trên nhiều nền tảng phần cứng (khác Intel).
- Open Source: Bao gồm cả kernel, drivers, công cụ phát triển
- Hỗ trợ nhiều hệ thống File: Minix-1, Xenix, System V , MS-DOS, VFAT, FAT-32, ISO 9660 (CD-ROMs). EXT, và EXT2
- Multiple Networking Protocols: Các giao thức nền tảng được hỗ trợ bởi Kernel như: TCP, Ipv4, Ipv6, AX.25, X.25, IPX, v.v…
- Multiprocessor Simultaneous Multiprocessing (SMP)
- Virtual Memory PagingMemory Protection: Hệ thống và các quá trình được bảo vệ lẫn nhau do đó không quá trình nào có thể làm cho toàn hệ thống sụp đổ.
- TCP/IP Networking: bao gồm ftp, telnet,…
- Client and Server Support: Bao gồm Netware, và Windows (SMB)
- Ký hiệu Linux Kernel

    - Các phiên bản của Linux. Các phiên bản của HDH Linux được xác định bởi hệ thống số dạng X.YY.ZZ. Nếu YY là số chẵn => phiên bản ổn định. YY là số lẻ => phiên bản thử nghiệm

- Ví dụ:
- Kernel 2.4.2
- 2 là Số chính
- .4 là số phụ , phiên bản ổn định
- .2 Patch Level, phiên bản ổn định (nếu số lẻ là phiên bản đang thử nghiệm)

III.3.2 Lịch sử phát triển

- 1969 Kend Thompson thiết kế môi trường nghiên cứu và phát triển chương trình đây chính là tiền 32ung HĐH Unix. Unix được viết bằng hợp ngữ
- 1973 Unix được viết lại bằng ngôn ngữ C, dễ hiểu hơn

1975 Mã nguồn của Unix được cung cấp cho các trường đại học. Unix được nhiều công ty, tổ chức phát triển

- Năm 1991 Linus Torvalds, sinh viên của đại học tổng hợp Helsinki, Phần lan, bắt đầu xem xét Minix, một phiên bản của Unix làm ra với mục đích nghiên cứu cách tạo ra một hệ điều hành Unix chạy trên máy PC với bộ vi xử lý Intel 80386
- Ngày 25/8/1991, Linus cho ra version 0.01 và thông báo trên comp.os.minix của Internet về dự định của mình về Linux
- 1/1992, Linus cho ra version 0.12 với shell và C compiler. Linus không cần Minix nữa để recompile HDH của mình. Linus đặt tên HDH của mình là Linux
- 1994, phiên bản chính thức 1.0 được phát hành

- Các phiên bản hệ điều hành Linux:

phân phối	Phiên bản mới nhất	Trang web chính thức	Các bản dẫn xuất
Ubuntu	17.10	http://www.ubuntu.com/	Kubuntu, Xubuntu, Edubuntu, Ubuntu Studio, Lubuntu, Macbuntu,Ubuntu Kylin, Ubuntu MATE
Debian GNU/Linux	9.2	http://www.debian.org/	
Elementary OS	0.4.1 Loki	http://www.elementaryos.org/	
Ultimate Edition	5.7	http://ultimateedition.info/	
Red Hat Enterprise Linux	7.4	http://www.redhat.com/rhel/	
Chrome Linux	2.4.1290	http://getchrome.eu/	
Chrome OS	62.0.3202.62	http://google.com/intl/en/chrome	
Fedora	27	http://www.fedoraproject.org/	
SUSE Linux Enterprise Desktop	13.2	http://vi.opensuse.org/	OpenSUSE 11.4, Mono 2.10.4

Linux Mint	18.2 sonya	http://linuxmint.com/	
Knoppix	8.1	http://www.knoppix.org/	
PCLinux OS	2017	http://www.pclinuxos.com/	
Mandrake	2011	http://www.mandriva.com	Mandriva
CentOS	7	http://www.centos.org/	
Gentoo	20170907	http://www.gentoo.org/	
Slackware	14.2	http://www.slackware.com/	
SLAX	7.0.8	http://www.slax.org/	
Sabayon	17.11	http://www.sabayon.org/	
Dreamlinux	5	http://www.dreamlinux.info/	
OpenSolaris	11.1	http://www.opensolaris.org/	
Hồng kỳ linux	6.0 SP3	http://www.redflag-linux.com/	
Puppy linux	6.3	http://puppylinux.org/	
Hacao Linux	2011	http://www.hacao.com/	
Asianux	4.5	http://www.asianux.vn/	Asianux Server
SliTaz	5.0 Rolling	http://www.slitaz.org/	GNU/Linux

Linpus	2.2	http://www.linpus.com/	Linpus Linux
Back Track	5r3	http://www.backtrack-linux.org/	Back Track - Linux, Kali Linux
BOSS Linux	6.0 (Anoop)	www.bosslinux.in	
Kali linux	2017.2	http://www.kali.org/	Kali - Linux, Back Track
Nova	4.0 (2013)	www.nova.cu	
Backbox	5	http://www.backbox.org	Backbox, linux
Canaima GNU/Linux	5.1	http://canaima.softwarelibre.gob.ve/	
Super Ubuntu	11.10	http://superubuntu.linuxfreedom.com/download.html	Ubuntu, Zorin OS, Linux Mint,
Zorin OS	12.2	http://zorin-os.com/	Ubuntu, Super Ubuntu, Linux Mint

III.3.3 Cài đặt hệ điều hành

- Các thao tác chuẩn bị cài đặt
  - Kiểm tra phần cứng máy tính
    - ✓ CPU, RAM, HDD: Tùy thuộc vào phiên bản Linux mà ta sẽ cài đặt.
    - ✓ Ví dụ: Để cài đặt RedHat Linux 6.0 ta cần cấu hình phần cứng tối thiểu như sau: CPU Intel 133MHz, 16MB RAM, 2Gb HDD Kiểm tra phần mềm
  - Phiên bản Linux sẽ cài đặt.

- Ví dụ: RedHat Linux 7.3 (phiên bản này yêu cầu CPU 233MHz ↑, 64MB RAM, 4Gb HDD

- Phân hoạch đĩa cứng
  - Đối với hệ điều hành Linux nó đòi hỏi phải có ít nhất 2 partition của đĩa cứng để có thể cài đặt thành công.
  - Partition thứ nhất: 36ung để chứa hđh. Dung lượng cho partition này tuỳ theo các package mà bạn cài đặt, thông thường khoảng 2Gb là đủ.
    - ✓ Swap Partition: Dung lượng cho parttion chỉ cần bằng dung lượng của RAM là vừa đủ
    - ✓ Đặc biệt đối với các hệ thống Linux mà sau này muốn cài đặt hệ quản trị CSDL Oracle lên thì ta phải cho swap space lớn hơn hoặc bằng 500MB
- Các chế độ cài đặt
  - Server
  - Workstation
  - Custom
  - Upgrade

III.3.4 Tập lệnh

- Quá trình khởi động RedHat Linux

  Tập tin đầu tiên mà hệ điều hành xem xét đến là /etc/inittab

  # Default runlevel. The runlevels used by RHS are:

  #0 – halt (Do NOT set initdefault to this)

  #1 – Single user mode

  #2 – Multiuser, without NFS

  #3 – Full multiuser mode

#4 – unused

#5 – X11

#6 – reboot (Do NOT set initdefault to this) id:3:initdefault:

- Tập lệnh cơ bản
  - Login : Đăng ký sử dụng
  - Who : cho biết thông tin về người sử dụng
  - Date: thông báo về thời gian
  - Exit, logout : chấm dứt phiên làm việc
  - Passwd: thay đổi password
  - Man: giúp đỡ
  - Ls : Liệt kê nội dung thư mục
  - Cp: sao chép tập tin
  - Mv: Đổi tên một tập tin
  - Rm: xóa tập tin
  - Cat: Hiển thị tập tin
  - Pwd: Cho biết thư mục hiện hành
  - Cd: Thay đổi thư mục hiện hành
  - Rmdir: xóa thư mục
  - Chmod: thay đổi quyền tập tin
  - Chgrp: thay đổi nhóm
  - Chown: thay đổi người sở hữu
  - df,du: thông hệ thống tập tin
  - Useradd: Tạo người 37ung
  - Usergroup:Tạo nhóm người 37ung
  - Su: chuyển đổi người 37ung
  - Biên dịch chương trình c: gcc

gcc -o output input.c

- chown: change owner. Thay đổi quyền sở hữu tập tin cho user khác. Chỉ được chạy bởi root

  chown user1 hello.txt

- chgrp: change group. Thay đổi quyền sở hữu tập tin cho nhóm khác . Chỉ được chạy bởi root

  chgrp users hello.txt

- chmod: change mode: Thay đổi quyền của file, file của ai người đó mới được thay đổi (ngoại trừ root, có quyền trên tất cả các file

  chmod g+r hello.txt

  chmod o-x hello.txt

  chmod u+xwr hello.txt

CÂU HỎI VÀ BÀI TẬP

1. Trình bày các đặc điểm của hệ điều hành DOS, Windows, Linux? Sực khác nhau giữa các hệ điều hành trên.
2. Cài đặt và sử dụng tập lệnh hệ điều hành MS-DOS?
3. Tạo đĩa CDRom có khả năng khởi động hệ điều hành DOS nhận diện được CDROM và USB ?
4. Cài đặt và sử dụng tập lệnh hệ điều hành Windows?
5. Cài đặt và sử dụng tập lệnh hệ điều hành Linux?
6. Cài đặt nhiều hệ điều hành trên một máy tính?

CHƯƠNG IV: HỆ THỐNG QUẢN LÝ TẬP TIN

IV.1 KHÁI NIỆM TẬP TIN – THƯ MỤC

- Tập tin
  - Tập tin là đơn vị lưu trữ thông tin của bộ nhớ ngoài.
  - Các tiến trình có thể đọc hay tạo mới tập tin.
  - Các thông tin trong tập tin là bền vững không bị ảnh hưởng bởi các xử lý ngoại trừ người sử dụng muốn xóa.
  - Tập tin được quản lý bởi hệ điều hành.
- Thư mục
  - Thư mục lưu trữ các tập tin theo một qui định.
  - Một số hệ thống coi thư mục cũng như là tập tin.
  - Hệ thống quản lý tập tin
  - Các tập tin được quản lý bởi hệ điều hành với một cơ chế riêng gọi là hệ thống quản lý tập tin.
- Hệ thống quản lý tập tin bao gồm: cách hiển thị , các yếu tố cấu thành tập tin, cách truy xuất, các thao tác trên tập tin và thư mục,…

IV.2 MÔ HÌNH QUẢN LÝ VÀ TỔ CHỨC TẬP TIN

- Tập tin
  - Tên tập tin:
    - Mỗi tập tin được quản lý bằng một tên.
    - Cách đặt tên tập tin mỗi hệ diều hành là khác nhau.
  - Cấu trúc tập tin:
    - Dãy các byte không có cấu trúc.
    - Dãy các record có chiều dài cố định
  - Kiểu tập tin: Các hệ điều hành hỗ trợ cho nhiều loại tập tin khác nhau

- Tập tin thường: là tập tin text hay nhị phân chứa các thông tin của người sử dụng.
- Thư mục: là một loại tập tin hệ thống dùng lưu trữ cấu trúc hệ thống tập tin
- Tập tin có ký tự đặc biệt: liên quan đến nhập xất thông qua các thiết bị nhập xuất

▪ Thuộc tính của tập tin: Các thông tin về tập tin gọi là thuộc tính tập tin. Thuộc tính tập tin thường là các thông in:

- Bảo vệ: bảo vệ việc truy xuất từ người sử dụng
- Mật khẩu: Mật khẩu cần thiết khi truy xuất.
- Người tạo: Chỉ danh người tập tin.
- Người sở hữu: Chỉ danh người sở hữu hiện tại.
- Chỉ đọc: 0: đọc ghi, 1: chỉ đọc
- Ẩn: 0 bình thường, 1 không hiển thị khi liệt kế
- hệ thống: 0 bình thường, 1 tập tin hệ thống .
- Khoá: 0 không khóa, 1 bị khóa
- Độ dài record: số byte trong một record
- Thời gian tạo: ngày , giờ tạo tập tin
- Thời gian truy xuất sau cùng: ngày , giờ truy xuất gần nhất
- Thời gian thay đổi cuối cùng: ngày, giờ thay đổi tập tin
- Kích thước hiện thời: Số byte tập tin

• Thư mục

▪ Hệ thống thư mục cấp bậc:

- Một thư mục thường chứa các entry. Mỗi entry thể hiện cho một tập tin(chứa các thuộc tính tập tin)
- Số lượng thư mục trên mỗi hệ điều hành là khác nhau.

- Một số hệ thống chỉ có một thư mục duy nhất, nhưng một số hệ thống có thư mục gốc, trong thư mục gốc có các thư mục con và trong các thư mục con lại có các thư mục con nữa.

▪ Đường dẫn

- Trong hệ thống tổ chức thư mục cấp bậc theo hình cây có hai cách để xác định một tập tin: đường dẫn thuyệt đối, đường dẫn tương đối.
- Trong hầu hết các hệ điều hành tổ chức thư mục "." và ".." để chỉ ra thư mục hiện hành và thư mục cha.

IV.3 CÁC CHỨC NĂNG HỆ THỐNG TẬP TIN

Một hệ thống tập tin thông thường bao gồm các chức năng sau:

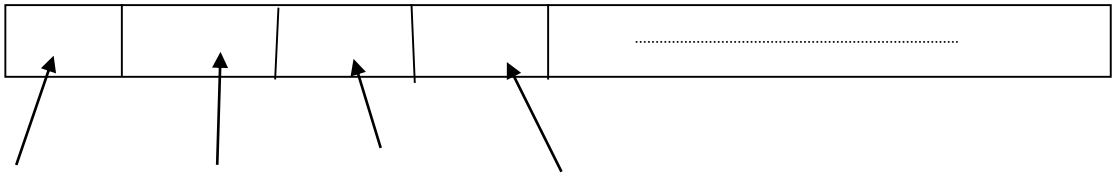
- Tạo tập tin
- Xoá tập tin
- Mở tập tin
- Đọc, ghi tập tin
- Tìm kiếm tập tin
- Đổi tên tập tin
- Tạo thư mục
- Xóa thư mục
- Đổi tên thư mục

IV.4 CÀI ĐẶT HỆ THỐNG TẬP TIN

- **Cấu trúc tập tin**
  - Mỗi tập tin hay thư mục (tập tin đặc biệt) được thể hiện bằng một Entry tập tin. Cấu trúc của Entry tập tin lưu trữ thông tin về tập tin tương ứng.

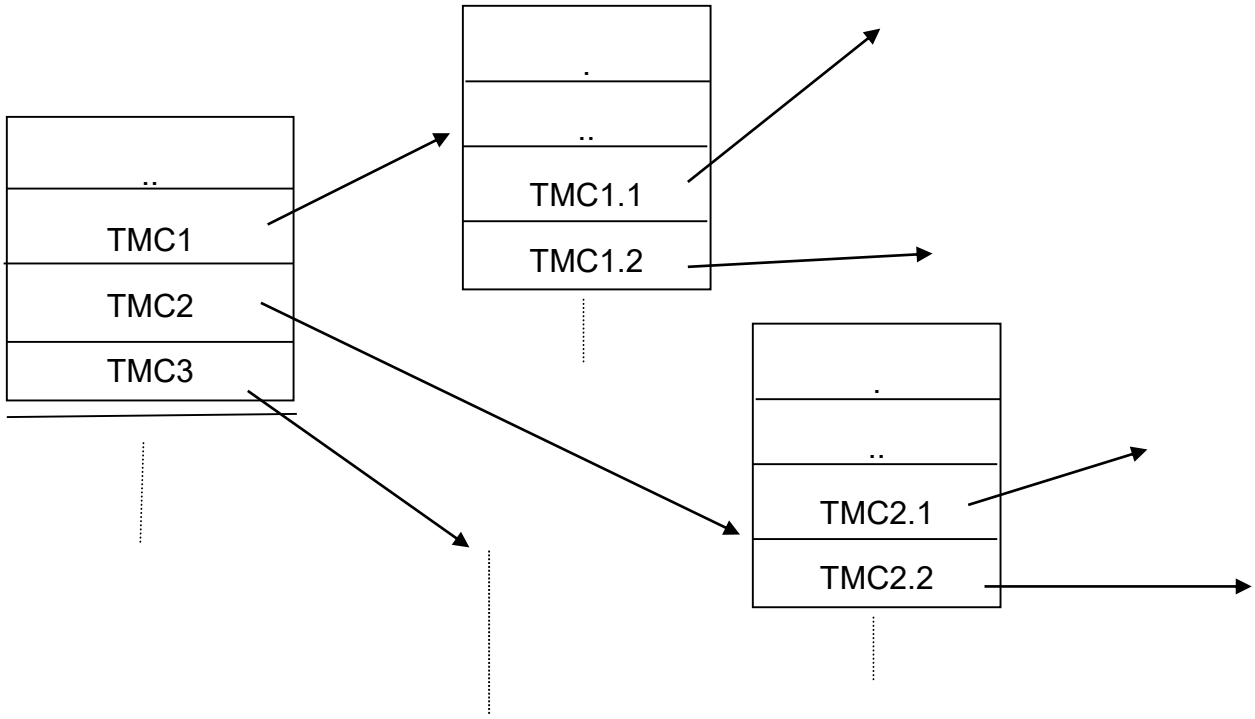
- Trước khi tập tin được đọc, ghi hệ thống phải biết đường dẫn do người sử dụng cung cấp từ đó định vị được cấu trúc entry của tập tin

- Cấu trúc một entry tập tin:



Mã NSD Tên TT PMR kích thước , …..

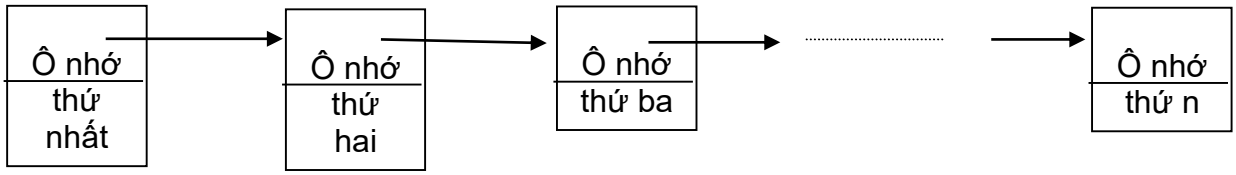
- **Cấu trúc thư mục**
  - Thư mục chứa các tập tin do đó chứa các entry của tập tin, kích thước mỗi Entry tuỳ thuộc mỗi hệ thống
  - Trong hệ thống quản lý tập tin cần phải định vị cấu trúc thư mục gốc, và các thư mục con



- **Quản lý vùng nhớ còn trống trên đĩa**

Hệ thống quản lý tập tin còn phải thực hiện lưu trữ các thông tin về các ô nhớ đã sử dụng chưa? Các ô nhớ đã sử dụng thuộc về tập tin, thư mục nào?

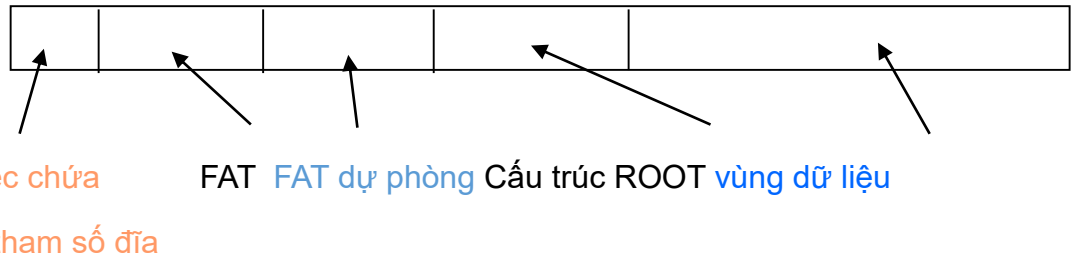
  - Phương pháp định vị liên tiếp: lưu trữ nội dung tập tin trên một dãy các khối liên tiếp nhau.
    - Ưu điểm : dễ cài đặt, dễ thao tác trên các tập tin
    - Khuyết điểm: Xử lý phức tạp khi kích thước tập tin thay đổi, sự phân mảnh trên đĩa gây lãng phí
  - Phương pháp định vị bằng danh sách liên kết: Không có sự phân mảnh vì các ô nhớ được cấp phát hết, truy xuất chậm vì các ô nhớ chứa nội dung tập tin nằm rải rác.



IV.5 HỆ THỐNG TẬP TIN MS-DOS

Cấu trúc tổng quát hệ thống tập tin MS-DOS bao gồm các cấu trúc:

- **Bảng tham số đĩa**: lưu trữ các thông tin về đĩa và các thông tin cần thiết cho hệ thống tập tin.
- Cấu trúc **FAT**(File Allocation Table): Lưu trữ các thông tin địa chỉ các ô nhớ còn trống, các ô nhớ nào thuộc về mỗi tập tin,…
- Cấu trúc **thư mục gốc**(ROOT):Cấu trúc thư mục gốc bao gồm các Entry của tập tin, thư mục con.
- Vùng **lưu trữ dữ liệu**(Data): Lưu trữ nội dung tập tin và các cấu trúc thư mục con. Cấu trúc thư mục con tương tự cấu trúc thư mục gốc.



- **Bảng tham số đĩa(BPB):** nằm trong bootsec bắt đầu tại địa chỉ offset 0 và chứa các thông tin:

Offset	**Kích thước(B)**	**Tên, ý nghĩa**	
+0h	3	JMP	Lệnh nhảy đến đoạn mồi HĐH
0h+3	8	Version	Tên Cty, Version của HĐH
+0Bh	2	SecSiz	Số byte trong một sector
+0Dh	1	ClustSiz	Số sector của cluster
+0Eh	2	ResSec	Số Sector trước bảng FAT
+10h	1	FatCnt	Số bảng FAT cài đặt
+11h	2	RootSiz	Số Entry tối đa của ROOT
+13h	2	TotSecs	Tổng số sector đĩa <32MB
+15h	1	Media	Byte chỉ danh đĩa
+16h	2	FatSiz	Số sector trong bảng FAT
+18h	2	TrkSecs	Số sector trên mỗi Track
+1Ah	2	HeadCnt	Số đầu đọc ghi
……			
+20h	4	TotSec	Tổng số sector đĩa >32MB

…….			
+3eh			Bắt đầu đoạn CTmồi HĐH

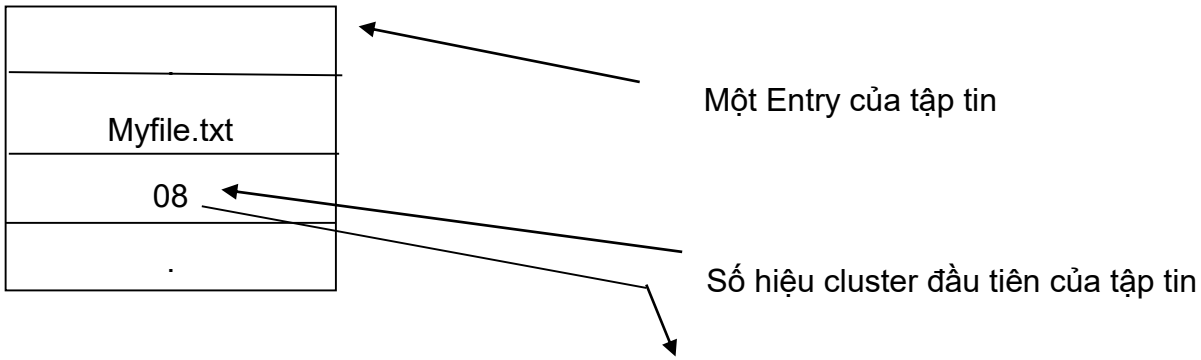
- Đối với đĩa cứng sector đầu tiên là masterboot chứa bảng tham số partition. Địa chỉ offset bảng partition bắt đầu tại 01BEh . Với một đĩa cứng có thể chi thành 4 partition, mỗi partition được thể hiện bằng một Entry(16byte) có cấu trúc như sau:

Offset	**Kích thước(B)**	**Tên, ý nghĩa**	
+0h	1	Boot	=0 không Active, =80h Active
+1h	1	HdBeg	Số mặt bắt đầu
+2h	2	SecCylBeg	Số cylinder bắt đầu(10bit) và sector bắt đầu (6bit)
+4h	1	Sys	=0Unknown, =1: Dos FAT 12bit, ; =4: Dos FAT 16bit
+5h	1	HdEnd	Số mặt kết thúc
+6h	2	SecCylEnd	Số cylinder kết thúc(10bit) và sector kết thúc (6bit)
+8h	4	SecLogic	Sector bắt đầu tương đối
+0Ch	4	TotSecs	Tổng số sector của partition

- **Cấu trúc Fat(file allocation table)**
  - Khái niệm Cluster: Khi đĩa được format đơn vị nhỏ nhất trên đĩa là Sector. Đối đĩa cứng lớn có nhiều sector mà Dos không thể quản lý được. Trong trường hợp này để giảm số sector cần quản lý bằng cách định nghĩa cluster là tập hợp các sector. Lúc này Dos chỉ quản lý Cluster thay vì sector.
  - Số hiệu các entry trong bảng Fat thể hiện cho các cluster có số hiệu tương ứng trong vùng dữ liệu.
  - Fat thể hiện thông tin về các cluster còn trống hay không? Và các cluster đã sử dụng thuộc về tập tin nào.
  - Do Trong Fat 2 phần tử đầu tiên dành riêng không sử dụng nên số hiệu các entry được đánh số từ 2 trở đi. Chính vì vậy số hiệu cluster cũng được đánh số từ 2 trở đi.
  - Fat có 3 loại : fat 12bit, fat 16bit, fat 32bit. Đối với Fat 12bit thì kích thước mỗi Entry trong Fat là 12 bit và quản lý được số cluster tối đa là (212 -2). Tương tự đối với Fat 16bit và Fat 32bit.
  - Fat 12bit dùng cho đĩa <32MB, Fat 16Bit dùng đĩa <2GB và Fat 32Bit dùng cho đĩa cứng > 2GB(HĐH Windows)
  - **Nội dung của Fat**
    - Dos quản lý tập tin bằng cách giá trị entry của cluster này chứa giá trị là số tứ tự entry tiếp theo nó, cứ thế các cluster của một tập tin tạo thành một chuỗi cho đến khi gặp dấu hiệu kết thúc tập tin.
    - Tuỳ thuộc vào loại fat 12bit hay 16bit các entry có giá trị như sau:
      - 0(000) : Cluster tương ứng số hiệu entry còn trống
      - (0)002- (F)FEF : Cluster đang chứa dữ liệu của một tập tin nào đó, giá trị là của nó là số hiệu cluster kế tiếp
      - (F)FF0 – (F)FF6 : Dành riêng không sử dụng

- (F)FF7 : Cluster hỏng
- (F)FF8- (F)FFF : Cluster cuối cùng của tập tin

Một ví dụ nội dung của Fat:



	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00			03	04	05	Ff	00	00	09	0a	0b	15	00	00	00	00
01	00	00	00	00	00	16	17	19	f7	1a	1b	ff	00	00	00	00
02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Tập tin myfile.txt dài 10 cluster: 8,9,0a,0b,15,16,17,19,1a,1b

Cluster 18h : đánh dấu hỏng

Cluster 2,3,4,5: thuộc một tập tin nào đó

Các cluster khác còn trống

Tìm các cluster của một tập tin

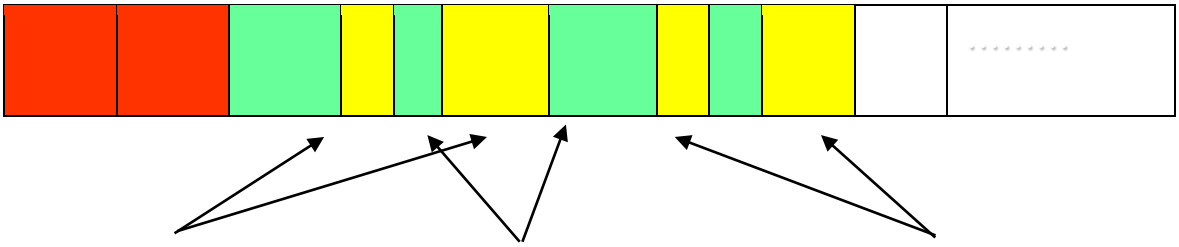
PointType ReadEntryFile(unsigned int ClustBegin)

```
{
        PointType ListClust, last;
        ListClust=last=NULL;
        unsigned clust;
        clust=ClustBegin;
        while(1)
        {
                InsertLast(ListClust, clust);
                clust=NextEntry(clust)
                if (clust==0x0fff)
                        break;

        }
        return ListClust;
}
```

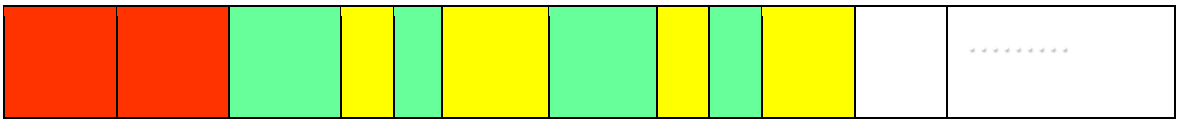
Đọc nội dung các Entry trong Fat:

- Nếu Fat 16 bit thì địa chỉ entry kế tiếp bằng địa chỉ entry hiện thời +2
- Nếu Fat 12 bit : Xét 3 byte một : số hiệu entry chẵn là 12 bit thấp và số hiệu entry lẻ là 12 bit cao



Số hiệu Entry 2 Số hiệu Entry 3 Số hiệu Entry 4

Tương ứng Cluster 2 Tương ứng Cluster 3 Tương ứng Cluster 4

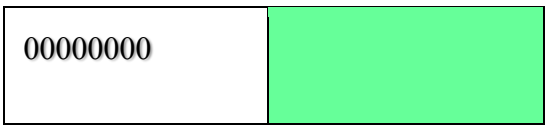


Các biến dùng chung:

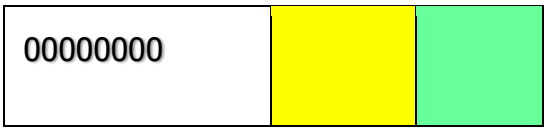
unsigned X, X1, Addr;

Trường hợp số hiệu Entry Chẵn

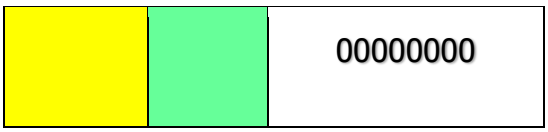
X = FAT[Addr];



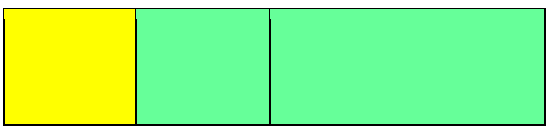
X1= FAT[Addr+1];



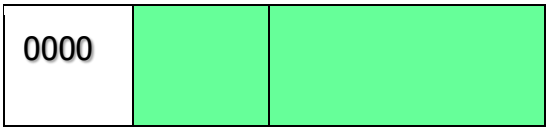
X1= X1 <<8;



X = X + X1;

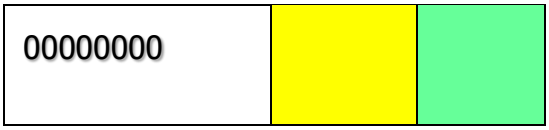


X = X & 0x0FFF;

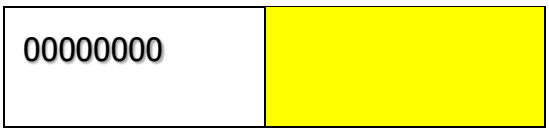


Trường hợp số hiệu Entry lẻ

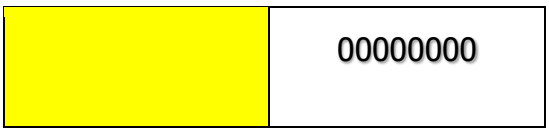
X = FAT[Addr];



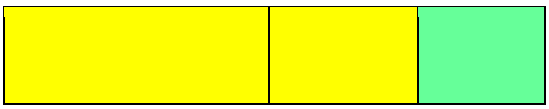
X1= FAT[Addr+1];



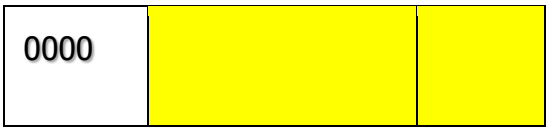
X1= X1 <<8;



X = X + X1;



X = X >>4;



Hàm đọc nội dung Entry Fat 12bit

unsigned NextEntry (unsigned Index)

{ unsigned X, X1, Addr;

*Addr= (Index *3)/ 2;*

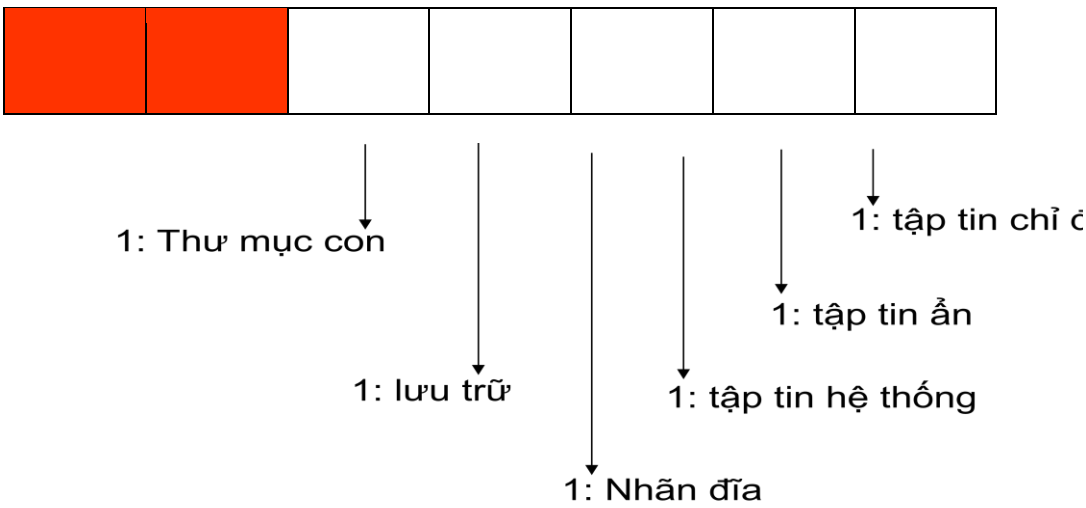
```
        X = FAT[Addr];
        X1= FAT[Addr+1];
        X1= X1 <<8;
        X = X + X1;
        if ((Index %2)==0)
                X= X & 0X0FFF;
        else
                X = X >>4;
        return X;
}
```

- **Cấu trúc một Entry Trong thư mục gốc hay thư mục con:** Mỗi Entry thể hiện cho một tập tin hay thư mục con và lưu trữ các thông tin cần thiết. Kích thước Entry là 32 byte

Offset	**Kích thước**	**Nội dung**
+0h	8	Tên tập tin hay thư mục
+8h	3	Phần mở rộng tên tập tin
+0bh	1	Thuộc tính tập tin
+0ch	0ah	Dành riêng không sử dụng

+16h	2	Thời gian tạo
+18h	2	Ngày tạo
+1ah	2	Số hiệu cluster đầu tiên
+1ch	4	Kích thước tập tin (bytes)

- Byte đầu tiên của entry thể hiệc các thông tin sau:
  - 0: Entry này còn trống
  - . : Thư mục cha
  - 0E5 : Entry của tập tin này tạm thời bị xóa
  - Ký tự bất kỳ : tên của một tập tin
- Diễn dải byte thuộc tính:



5bit: giờ	6bit: phút	5bit: giây

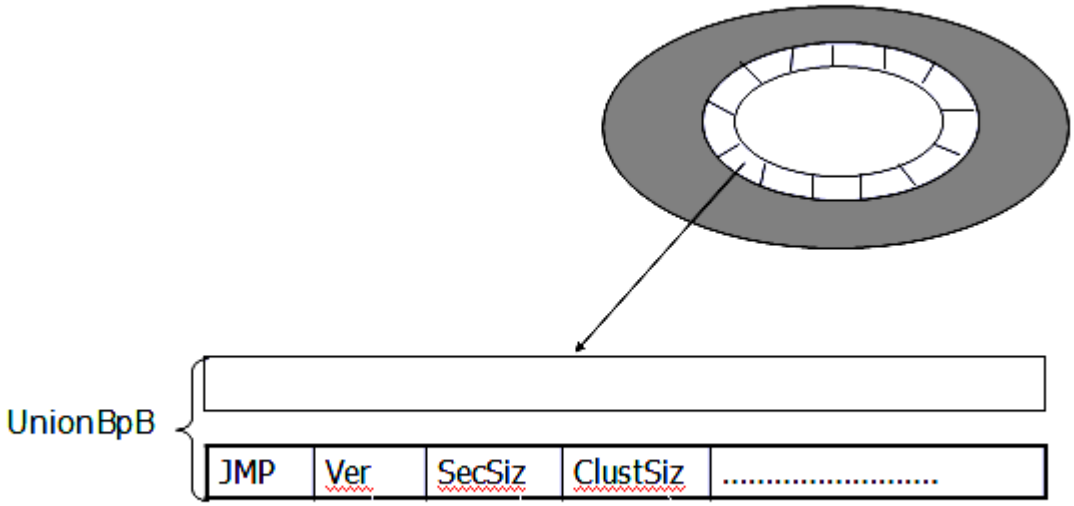
- **Cấu trúc dữ liệu cài đặt:**
  - **Khai báo bảng tham số đĩa**

```
Typedef struct {
        unsigned char    JMP[3];
        unsigned char    Ver[8];
        unsigned         SecSiz;
        unsigned char    ClustSiz;
        unsigned         ResSec;
        unsigned char    FatCnt;
        unsigned         RootSiz;
        unsigned         TotSec;
        unsigned char    Media;
        unsigned         FatSiz;
        unsigned         TrkSec;
        unsigned         HeadCnt;
        unsigned         HidSec;
       }EntryBPB;

typedef union {
        unsigned char    Sector[512];
```

EntryBPB Entry;

}UnionBPB;



Ví dụ

```
#include <iostream.h>
#include<dos.h>
Typedef struct {.....
}EntryBPB
Typedef Union {.....}UnionBPB
Int main()
{
        UnionBPB  bpb;
```

```
        ReadDisk (0,1, bpb.sector);
        cout<<bpb.Entry.secsiz;
        cout <<bpb.Entry.Clustsiz;
        ……..
        Return 1;
}
```

- **Khai báo kiểu thời gian tạo tập tin, thư mục**

```
typedef struct {
        unsigned    S:5 ;
        unsigned    M: 6 ;
        unsigned    H:5;
        }Time;
typedef union {
        unsigned    intTime;
        Time        T ;
        }UnionTime;
```

- **Khai báo kiểu ngày tháng năm tạo tập tin, thư mục**

```
typedef struct {
```

```
                unsigned    D: 5 ;
                unsigned    M: 6 ;
                unsigned    Y:7;
                }Date;
    typedef union {
                unsigned    intDate;
                Date        Day ;
                }UnionDate;
```

- **Khai báo cấu trúc byte thuộc tính tập tin**

```
typedef struct {
            unsigned char    ReadOnly : 1 ;
            unsigned char    Hidden   : 1 ;
            unsigned char    System   : 1 ;
            unsigned char    Volume   : 1 ;
            unsigned char    SubDir   : 1 ;
            unsigned char    Archive  : 1 ;
            unsigned char    DR       : 2 ;
    }Attrib;
typedef union {
            unsigned char    charAtt;
```

```
        Attrib      Attr ;
        }UnionAttrib;
```

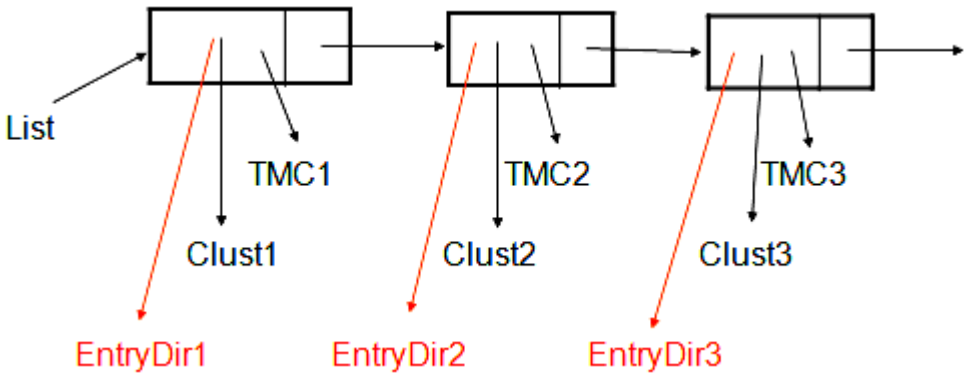
- **Khai báo cấu trúc một Entry tập tin**

```
typedef struct {
        unsigned char     FileName[8] ;
        unsigned char     Ext[3] ;
        UnionAttrib       Attribute ;
        unsigned char     DR[10] ;
        UnionTime         CreateTime ;
        UnionDate         CreateDate ;
        unsigned          ClustBegin ;
        long              FileSize;
    }EntryDir;

typedef union {
        unsigned char     Entry[32];
        EntryDir          EntDir ;
    }UnionDir;
```

- **Danh sách liên kết**

```
typedef struct  Node{
        void  *Data;
        Node *Next;
    }NodeType;
typedef  NodeType *PointerType;
```



- **Hàm InsertLast**

```
int  InsertLast (PointerType &List, PointerType &Last,  void* Item)
{
        PointerType  Temp;
        Temp= new  NodeType;
        if( !Temp)
            return 0;
```

```
        Temp->Data = Item;
        Temp->Next = NULL;
        if(List==NULL)
                List=Temp;
        else Last->Next=Temp;
        Last = Temp;
        return 1;
}
```

- **Hàm đọc Sector từ đĩa vớI số hiệu theo kiểu vật lý**

```
int ReadDiskBios(unsigned char *Buff,unsigned Side, unsigned TrackSec, unsigned Sector, unsigned SecNum)
{
        union  REGS  u, v;
        struct  SREGS  s;
        int k, i=0;
        v.x.cflag=1;
        while(i<2) && (v.x.cflag!=0))
        {
                u.h.ah= 0x2;// Đọc đĩa
                u.h.dl=0 ; ổ đĩa mềm A
                u,h.dh=Side;
```

```
                u.x.cx=TrackSec;  // Track và Sector bắt đầu
                u.h.al=SecNum;
                s.es=FP_SEG(Buff);
                u.x.bx=FP_OFF(Buff);
                int86x (0x13, &u, &v, &s);
                i++;
            }
            k=v.h.ah;
        return (!v.x.cflag);
    }
```

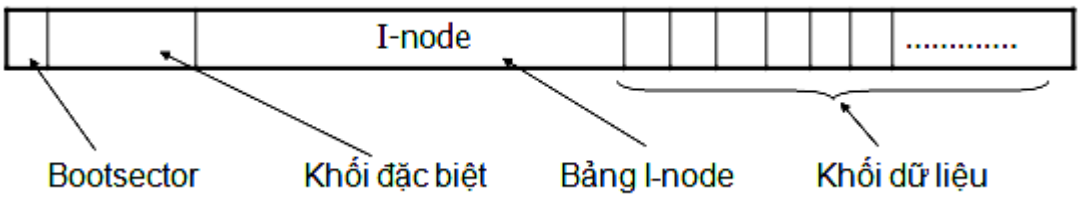
- **Hàm đọc đĩa với đầu vào là sector logic**

```
int ReadDisk(char *Buff, long SectorBegin, int SecNum)
{
        unsigned Side, Track, Sector, X;
        unsigned char X1;
    Sector=(unsigned)(1+(SectorBegin)%Bpb.TrkSec);
    Side=(unsigned)(((SectorBegin/Bpb.TrkSec)%Bpb.HeadCnt);
    Track=(unsigned)((SectorBegin)/(Bpb.TrkSec*Bpb.HeadCnt));
    X=Track;
    X=X & 0XFF00;
```

```
        X=X>>2;
        X=X & 0X00FF;
        X=X | Sector;
        Track = Track <<8;
        TrackSec=Track | X; //Track ( 10), Sector(6)
        if ( ReadDiskBios(Buff, Side, TrackSec, Sector, SecNum))
                return 1;
        else
                return 0;
}
```

IV.5 HỆ THỐNG TẬP TIN UNIX

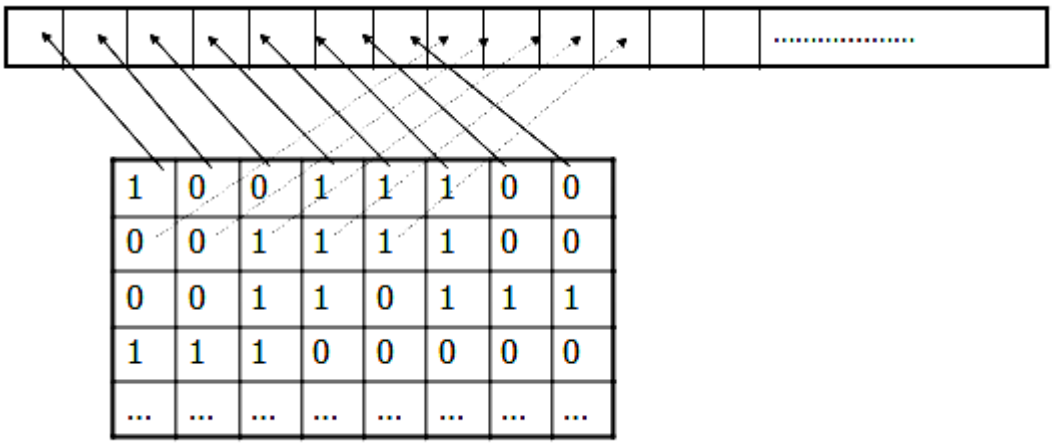
- **Cấu trúc tổng quát**



- BootSector : chứa chương trình mồi hệ điều hành
- Khối đặc biệt: lưu trữ các thông tin quan trọng về toàn bộ hệ thống tập tin, (số I-node, số khối đĩa, dãy các ô nhớ còn trống trên đĩa,…)
- Sau khối đặc biệt là bảng I-Node được đánh số từ 1 đến tối đa.
- Khối dữ liệu là vùng nhớ lưu trữ nội dung tập tin, thư mục.

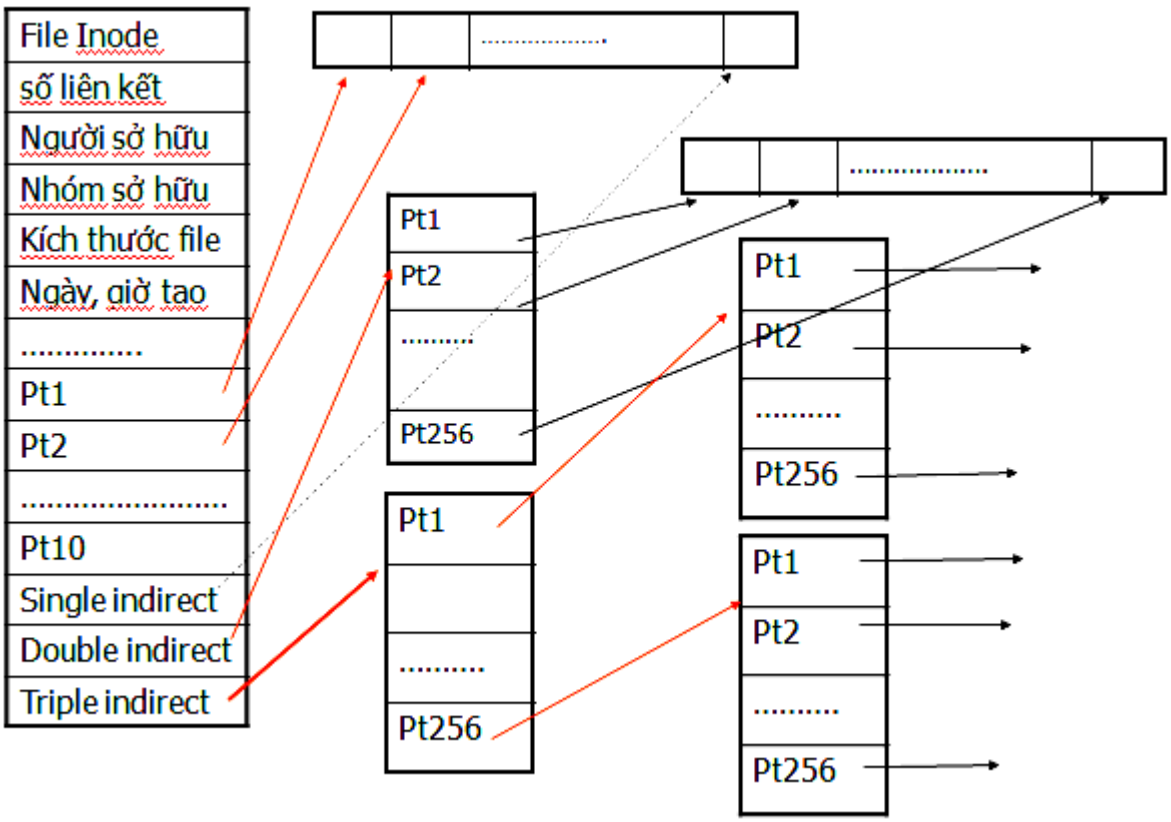
- **Cách quản lý các ô nhớ còn trống trong khối dữ liệu:**

Dùng phương pháp Bitmap: Với đĩa có n ô nhớ sẽ được ánh xạ thành n bit với giá trị 1 là còn trống, giá trị 0 là đã chứa dữ liệu. Như vậy một đĩa 20MB cần 20000 bit để lưu trữ thông tin. Chiếm khoảng 3 ô nhớ.



- Tổ chức lưu trữ tập tin:
  - Mỗi file trong unix tương ứng với một I-Node. Một I-Node có kích thước 64byte bao gồm các thông tin về file: file Node, quyền sở hữu file của người sử dụng, quyền sở hữu nhóm, kích thước file, thời điểm tạo,thời điểm truy cập sau cùng, thời điểm thay đổi sau cùng, địa các ô nhớ chứa nội dung,…
  - Phần đánh địa chỉ các ô nhớ nội dung tập tin được chia thành 2 phần: phần đầu gồm 10 pt chứa được 10 địa chỉ ô nhớ. Phần thứ 2 gồm 3 con trỏ gián tiếp: Single indirect, double indirect, triple indirect.
  - Đối với tập tin có kích thước nhỏ hơn 10 ô nhớ dữ liệu thì các con trỏ gián tiếp không được sử dụng để ghi địa chỉ.

- Khi một file có kích thước lớn hơn 10 ô nhớ dữ liệu thì con trỏ single indirect được sử dụng chỉ đến một ô nhớ dành riêng, ô nhớ này lại chứa 256 địa chỉ của ô nhớ dữ liệu.
- Tương tự con trỏ double chỉ đến một ô nhớ chứa 256 địa chỉ ô nhớ và mỗi ô nhớ này lại chứa 256 địa chỉ ô nhớ dữ liệu.
- Một file lớn nhất sử dụng cả 3 con trỏ gián tiếp là 16 GB.



- **Cấu trúc thư mục**

Cấu trúc thư mục được sử dụng trong Unix vô cùng đơn giản, mỗi entry bao gồm tên tập tin, số hiệu I-Node của file. Mỗi Entry có kích thước 16 byte:

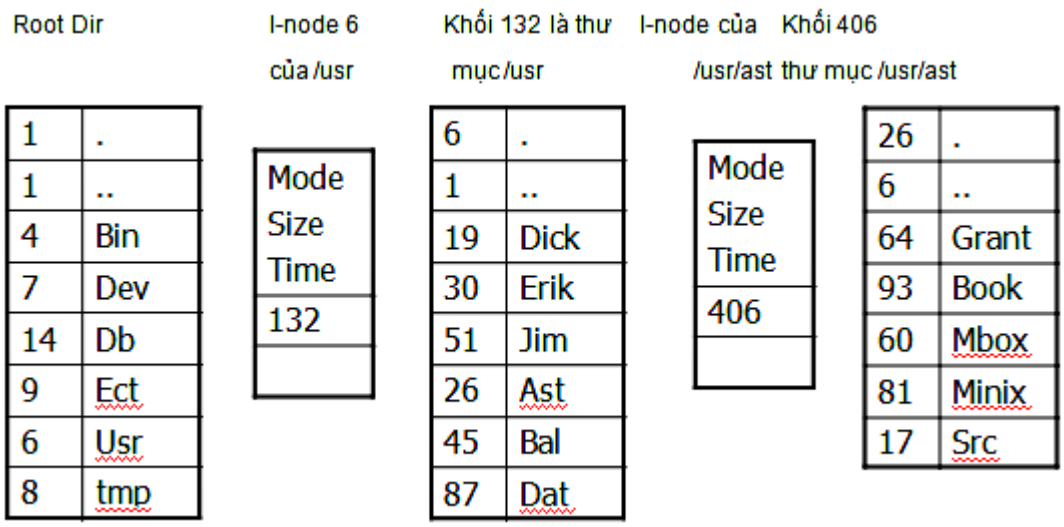
I-Node	Tên tập tin

Khi một file được mở , hệ thống file phải xác định vị trí khối dữ liệu trên đia nhờ vào đường dẫn được cung cấp.

Ví dụ: cách truy tìm file dựa vào đường dẫn được cung cấp :

/usr/ast/mbox

- Trước hết đọc I-Node của thư mục gốc – là I-Node đầu tiên trong bảng I-Node
- Đọc từng entry trong thư mục gốc so sánh với thư mục usr từ đó tím ra I-node của thư mục usr.
- Từ I-node này tiếp tục xác định các entry trong thư mục và so sánh với ast khi đó tìm được I-node của /usr/ast. Tiếp tục tương tự sẽ tìm được I-node của /usr/ast/mbox



CÂU HỎI VÀ BÀI TẬP

1. Trình bày các thành phần cấu trúc hệ thống tập tin?
2. Trình các chức năng hệ thống tập tin?
3. Trình bày cấu trúc tổ chức hệ thống tập tin MS-DOS
4. Trình bày cấu trúc tổ chức hệ thống tập tin Unix
5. Trình bày sự khác biệt giữa 2 cấu trúc tổ chức hệ thống tập tin MS-DOS và Unix?
6. Viết chương trình đọc và xuất ra màn hình bảng tham số đĩa mềm 1.44mb?
7. Viết chương trình xem nội dung thư mục gốc ?
8. Viết chương trình xem nội dung thư mục bất kỳ ?
9. Viết chương trình xem nội dung tập tin ?
10. Viết chương trình xoá tập tin , phục hồi tập tin ?
11. Viết chương trình copy tập tin ?
12. Viết chương trình sắp xếp đĩa (dồn clust)?
13. Viết chương trình xoá các clust mồ côi ?
14. Viết chương trình tạo thư mục?

CHƯƠNG V: HỆ THỐNG QUẢN LÝ TIẾN TRÌNH

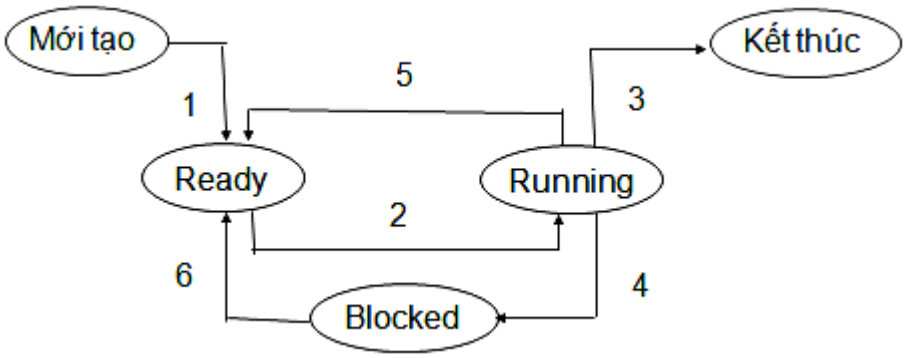
V.1 KHÁI NIỆM TIẾN TRÌNH

- Trong hệ thống đa chương có thể thể thực hiện nhiều tác vụ đồng thời.
- Việc thực hiện đồng thời này được hiện bằng cách chuyển đổi CPU qua lại giữa các chương trình. Điều này tạo cảm giác có nhiều chương trình thực hiện đồng thời.
- Trong hệ thống như vậy tất cả phần mềm được tổ chức thành một số tiến trình.
- Một tiến trình là một chương trình đang được xử lý, sở hữu con trỏ lệnh , tập các thanh ghi, biến và để hoàn thành nhiệm vụ của mình một tiến trình phải sử dụng các tài nguyên máy tính như CPU, bộ nhớ chính, các tập tin và thiết bị nhập xuất.
- Ý tưởng là có thể xem như mỗi tiến trình sở hữu một CPU ảo cho riêng mình, nhưng trong thực tế chỉ có một bộ xử lý thật sự được chuyển đổi qua lại giữa các tiến trình.
- Hệ điều hành chịu trách nhiệm sử dụng một thuật toán điều phối để quyết định thời điểm cần dừng một tiến trình để thực hiện một tiến trình khác

V.2 CÁC TRẠNG THÁI CỦA TIẾN TRÌNH

- Trạng thái của một tiến trình tại một thời điểm được xác định bằng hoạt động hiện thời của tiến trính đó.
- Tại một thời điểm một tiến trình có thể nhận một trong các trạng thái sau đây:
    - Mới tạo: Tiến trình đang được tạo lập.
    - Running: các chỉ thị của tiến trình đang được xử lý.
    - Blocked: Tiến trình chờ được cấp phát một tài nguyên hay chờ một sự kiện nào đó xảy ra.
    - Ready: Tiến trình chờ cấp phát CPU để xử lý.
    - Kết thúc : Tiến trình hoàn tất xử lý.

- Mô hình chuyển đổi giữa các trạng thái:



(1) Tiến trình mới tạo được đưa vào hệ thống.

(2) Bộ lập lịch cấp phát cho tiến trình một khoảng thời gian sử dụng CPU

(3) Tiến trình kết thúc

(4)Tiến trình yêu cầu một tài nguyên nhưng chưa được đáp ứng hoặc phải chờ thao tác nhập xuất.

(5) Bộ lập lịch thu hồi CPU và cấp phát cho tiến trình khác

(6) Tài nguyên mà tiến trình yêu cầu đã được cấp phát hay thao tác nhập xuất đã hoàn tất.

V.3 CÀI ĐẶT TIẾN TRÌNH

- Hệ điều hành quản lý các tiến trình trong hệ thống thông qua khối quản lý tiến trình (Process Control Block- PCB).
- PCB là một vùng nhớ lưu trữ các thông tin mô tả cho tiến trình như sau:
    - Chỉ danh của tiến trình: Để phân biệt các tiến trình
    - Trạng thái tiến trình: Xác định hoạt động hiện hành của tiến trình

- ✓ Ngữ cảnh của tiến trình: quản lý các tài nguyên của tiến trình:
- ✓ Trạng thái CPU : nội dung các thanh ghi.
- ✓ Bộ nhớ chính: Danh sách cácô nhớ được cấp phát cho tiến trình.
- ✓ Tài nguyên sử dụng: Danh sách các tài nguyên hệ thống mà tiến trình đang sử dụng.
- ✓ Tài nguyên tạo lập: Danh sách tài nguyên do tiến trình tạo lập.

- Thông tin giao tiếp: Phản ánh các thông tin về quan hệ của tiến trình với các tiến trình khác trong hệ thống:
  - ✓ Tiến trình cha: Tiến trình tạo lập tiến trình này.
  - ✓ Tiến trình con: Các tiến trình do tiến trình này tạo lập.
  - ✓ Độ ưu tiên: Giúp bộ lập lịch lựa chọn tiến trình được cấp pháp CPU.
- Thông tin thống kê: thống kê về hoạt động của tiến trình: thời gian sử dụng CPU, thời gian chờ.

V.4 TIỂU TRÌNH

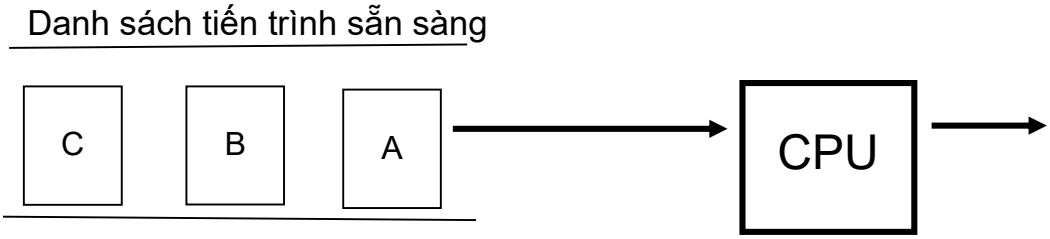
- Trong hệ điều hành mỗi tiến trình có không gian địa chỉ và có một dòng xử lý, nhưng đôi khi người sử dụng mốn có nhiều dòng xử lý cùng chia xẻ trong cùng không gian địa chỉ và các dòng xử lý này hoạt động song song tương tư như các tiến trình phân biệt khác.
- Mỗi dòng xử lý phân biệt này gọi là một tiểu trình.
- Mỗi tiểu trình xử lý tuần tự đoạn mã của minh và sở hữu con trỏ lệnh tập các thanh ghi, stack riêng. Các tiểu trình chia sẻ CPU như các tiến trình độc lập.
- Một tiến trình có thể sở hữu nhiều tiểu trình .
- Các tiểu trình trong một tiến trình có thể chia sẻ tài nguyên của tiến trình cha (các biến toàn cục)

V.5 LẬP LỊCH TIẾN TRÌNH

- Trong hệ thống đa nhiệm tại một thời điểm có thể nhiều tiến trình đồng thời sẵn sàng để xử lý. Mục tiêu là chuyển đổi CPU qua lại các tiến trình thường xuyên.
- Để thực hiện điều này hệ điều hành phải lựa chọn tiến trình kếtiếp để xử lý. Bộ lập lịch sẽ sử dụng thuật toán để thực hiện.
- Mục tiêu của bộ lập lịch: Hệ điều hành xây dựng nhiều chiến lược khác nhau thực hiện lập lịch nhưng phải đạt các mục tiêu như sau:
    - Sự công bằng: Các tiến trình chia sẻ CPU một cách công bằng. Không tiến trình nào chờ vô hạn mới được cấp pháp CPU
    - Tính hiệu quả: Hệ thống phải tận dụng CPU 100% thời gian
    - Thời gain đáp ứng hợp lý: Cực tiểu hóa thời gian hồi đáp cho cac tương tác của người sử dụng.
    - Thời gian lưu lại hệ thống : Cực tiểu hóa thời gian hoàn tất các tác vụ xử lý theo lô.
    - Thông lượng tối đa: Cực đại hóa số công việc được xử lý trong một đơn vị thời gian
- Tất cả mục tiêu trên thường không thỏa hết vì chính bản thân chúng có sự mâu thuẫn với nhau.
- Lập lịch tiến trình:
    - Hệ điều hành tổ chức một danh sách chứa các tiến trình đang sẵng sàng
    - Hệ điều hành sẽ chọn một tiến trình trong danh sách sẵng sàng để cấp phát CPU.
    - Các chiến lược lập lịch tiến trình

V.5.1 Chiến lược lập lịch tiến trình FIFO

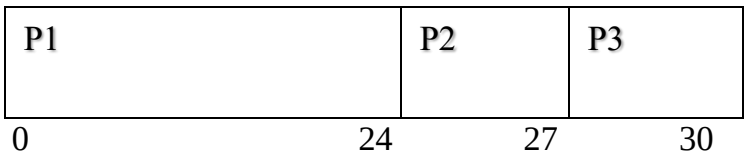
CPU được cấp phát cho tiến trình đầu tiên trong danh sách sẵn sàng- là tiến trình được đưa vào hệ thống sớm nhất.



Ví dụ:

Tiến trình	**Thời điểm vào**	**Thời gian xử lý**
P1	0	24
P2	1	3
P3	2	3

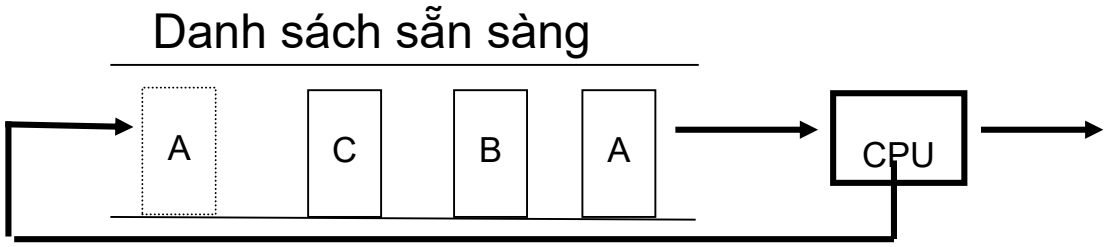
Thứ tự cấp phát CPU cho các tiến trình:



- Thời gian chờ được xử lý của P1 : 0
- Thời gian chờ được xử lý của P2 : 24-1=23
- Thời gian chờ được xử lý của P3 : 24+3 -2 =25
- Thời gian chờ trung bình là : (0+ 23+ 25)/3 =16 milisecondes
- Thời gian chờ trung bình không đạt cực tiểu và xảy ra hiện tượng tích luỹ thời gian tất cả tiến trình phải chờ một tiến trình có yêu cầu thời gian dài kết thúc.

V.5.2 Chiến lược Round Robin

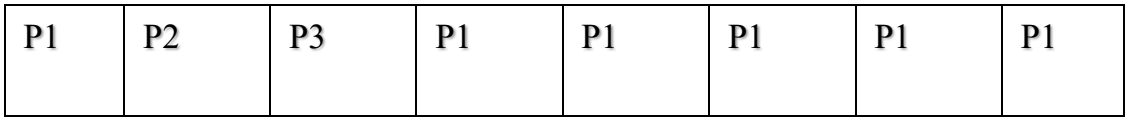
- Trong chiến lược này danh sách sẵn sàng được sử dụng như danh sách vòng. Bộ điều lập lịch lần lượt cấp phát cho từng tiến trình trong danh sách một khoảng thời gian sử dụng CPU gọi là Quantum
- Khi một tiến trình sử dụng hết thời gian Quantum dành cho nó thì hệ điều hành thu hồi CPU cấp cho tiến trình khác trong danh sách.
- Nếu tiến trình bị Blocked hoặc kết thúc trước khi hết Quantum thì hệ điều hành cũng hu hồi CPU.
- Nếu một tiến trình sử dụng hết Quantum mà chưa xử lý xong sẽ được đưa vào cuối danh sách sẵng sàng để chờ cấp phát CPU lần sau.



Ví dụ:

Tiến trình	**Thời điểm vào**	**Thời gian xử lý**
P1	0	24
P2	1	3
P3	4	3

Với Quantum = 4 thứ tự cấp phát CPU như sau:



0 4 7 10 14 18 22 26 30

- Thời gian chở xử lý P1: 0
- Thời gian chờ xử lý P2 : 4-1=3
- Thời gian chờ xử lý P3: 7-2 = 5
- Thời gian chờ xử lý P1 lần sau: 10-4=6
- Thời gian chờ trung bình : (0+3+5+6)/3 =4.66 Milisecondes

- Thời gian của Q quá bé thì chuyển đổi CPU giữa các tiến trình quá nhiều khiến việc sử dụng CPU không hiệu quả.
- Nếu Q quá lớn thì tăng thời gian hồi đáp và giảm khả năng tương tác của hệ thống

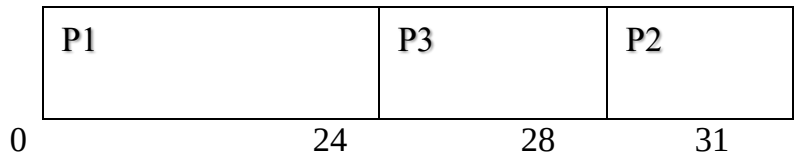
V.5.3 Chiến lược gán độ ưu tiên

- Mỗi tiến trình được gán một độ ưu tiên tương ứng, tiến trình nào có độ ưu tiên cao hơn sẽ được chọn cấp phát CPU đầu tiên.
- Độ ưu tiên được định nghĩa trong nội tại hoặc từ bên ngoài.
- Chiến lược độ ưu tiên không độc quyền: Khi một tiến trình được đưa vào danh sách sẵn sàng, độ ưu tiên của nó được so sánh với độ ưu tiên của tiến trình đang xử lý. Bộ lập lịch sẽ thu hồi CPU từ tiến trình hiện hành để cấp phát cho tiến trình mới nếu độ ưu tiên của nó cao hơn độ ưu tiên của tiến trình hiện hành
- Chiến lược độ ưu tiên độc quyền: CPU vẫn được cấp phát cho tiến trình hiện hành mặc dù tiến trình mới vào có độ ưu tiên cao hơn độ ưu tiên của tiến trình hiện hành.

Ví dụ : Chiến lược độ ưu tiên độc quyền

Tiến trình	**Thời điểm vào**	**Độ ưu tiên**	**Thời gian xử lý**
P1	0	3	24
P2	1	2	3
P3	2	1	4

Thứ tự cấp phát CPU như sau:

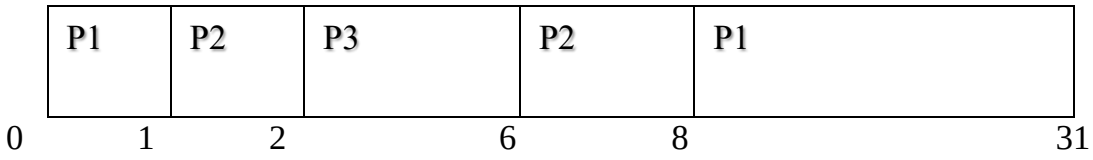


Ví dụ : Chiến lược độ ưu tiên không độc quyền

Tiến trình	**Thời điểm vào**	**Độ ưu tiên**	**Thời gian xử lý**
P1	0	3	24
P2	1	2	3

P3	2	3	4

Thứ tự cấp phát CPU như sau:



- Với chiến lược này tiến trình có độ ưu tiên thấp sẽ đợi CPU vô hạn. Để tránh trường hợp này thì bộ lập lịch phải giảm dần độ ưu tiên của các tiến trình sau một chu kỳ thời gian.

V.5.4 Chiến lược công việc ngắn nhất được thực hiện trước:

- Đây là thuật giải dành cho hệ thống xử lý theo lô, khi mà thời gian chạy của mỗi công việc được biết trước.
- Giả sử a, b, c, d lần lượt là thời gian của 4 công việc. Nếu cho 4 công việc này chạy theo thứ tự đó thì thời gian chạy trung bình là :

  (4a+3b+2c+d) /4.

- Dễ dàng thấy là nếu chọn công việc ngắn cho chạy trước thì giá trị thời gian chạy trung bình là nhỏ nhất.

V.6 ĐỒNG BỘ HÓA TIẾN TRÌNH

- Sự liên lạc giữa các tiến trình
  - Trong môi trường đa nhiệm các tiến trình không chạy độc lập mà thường xuyên có nhu cầu trao đổi thông tin với nhau.

- Nguyên tắc chung trao đổi thông tin giữa các tiến trình: Sử dụng vùng nhớ được chia sẻ, Trao đổi thông điệp.
- Vấn đề nảy sinh : xảy ra hiện tượng đua nhau sử dụng vùng nhớ chia xẻ dẫn đến kết quả không chính xác - Cần phải đồng bộ hóa tiến trình.

- Điều kiện đua: Nếu có nhiều tiến trình đọc , ghi dữ liệu vào vùng nhớ dùng chung và kết quả cuối cùng phụ thuộc thời điểm tiến trình nào chạy thật sự gọi là điều kiện đua.
- Vùng găng:
  - Để tránh điều kiện đua, nếu hệ điều hành không cho phép có nhiều tiến trình đọc hoặc ghi vào vùng nhớ lưu trữ chung tại cùng một thời điểm cần phải đạt sự loại trừ lẫn nhau
  - Một phần nào đó của chương trình mà tại đó thực hiện truy cập đến vùng nhớ dùng chung gọi là vùng găng

- Để tránh điều kiện đua thì hệ điều hành phải được thiết kế sao cho không cho phép 2 hay nhiều hơn tiến trình đồng thời trong vùng găng.
- Bốn điều kiện cần đảm bảo để thiết kế hệ điều hành cho phép nhiều tiến trình sử dụng vùng nhớ dùng chung một cách đúng đắn và hiệu quả:
  - (1) Không cho phép có nhiều hơn một tiến trình đồng thời trong vùng găng.
  - (2) Khi lập trình các tiến trình không được phép có bất kỳ giả định nào về tốc độ CPU và số lượng CPU.
  - (3) Không cho phép một tiến trình ở ngoài vùng găng của mình làm Blocked một tiến trình khác.
  - (4) Không cho phép bất kỳ một tiến trình nào đợi thời gian quá lâu mới có thể vào vùng găng của mình.

V.6.1 Các phương pháp thực hiện loại trừ nhau vào vùng găng

V.6.1.1 Dùng biến khóa

Hệ điều hành sử dụng một biến dùng chung, gọi là biến khóa lock được khởi tạo =0. Khi một tiến trình muốn vào vùng găng, nó thực hiện kiểm tra biến lock

```
int lock =0;


if (lock==0) (*)
        { lock =1;
         tiến trình trong vùng găng;
        }
else
        { tiến trình đợi cho đến khi lock =0
        }
```

Với phương pháp này vi phạm điều kiện (1) khi bộ lập lịch thu hồi CPU của một tiến trình tại (*), tiến trình này đã ghi nhận biến lock =0 và chuẩn bị vào găng. Lúc này bộ lập lịch cấp phát CPU cho tiến trình khác, tiến trình này kiểm tra biến lock thấy vẫn =0 và được phép vào găng. Nếu bộ lập lịch lại cấp phát CPU cho tiến trình trước thì tiến trình này tiếp tục chạy và thực hiện vào găng. Vậy tại một thời điểm có 2 tiến trình vào găng.

V.6.1.2 Luân phiên ngặt

Ý tưởng: Dùng một biến dùng chung turn=0; và mỗi tiến trình có đoạn mã sau:

```
whie   (TRUE)                          while(TRUE)
{                                  {
        while (turn !=0);                  while(turn !=1);
        Tronggăng();                       TrongGăng();
        turn=1;                                turn =0;
        ngoàigăng();                       NgoàiGăng();
}                                  }
```

Phương pháp này vi phạm điều kiện (4) vì theo phương pháp này hai tiến trình phải thay phiên ngặt vào ra vùng găng. Nếu một tiến trình vào ra vùng găng không luân phiên sẽ gây ra tiến trình còn lại đợi quá lâu không vào vùng găng.

V.6.1.3 Giải pháp Peterson

```
#define  FALSE  0
#define  TRUE   1
#define  N      2
int turn;
int interested[N]; /* khởi gán bằng FALSE*/

void Vàogăng(int Process)
{
```

```
        int other;
        other = 1- Process;
        interested[Process]= TRUE;
        turn = Process;
        while (turn ==Process && interested[other] ==TRUE);
}

void RaGăng(int Process)
{
        interested[Process]=FALSE;
}
```

Khi một tiến trình nào đó muốn vào vùng găng thi gọi hàm **Vaogang()** và truyền tham số là số hiệu tiến trình.

Khi tiến trình muốn ra khỏi vùng găng nó gọi hàm **RaGăng()**

Mặc dù giải pháp Peterson là chấp nhận được vì thỏa mãn 4 điều kiện nhưng bị hàn chế:

1. Khi một tiến trình đợi vào vùng găng tiến trình vẫn sử dụng thời gian CPU – Lãng phí CPU.

2. Khi đưa ra khái niệm độ ưu tiên cho các tiến trình giải pháp Peterson không đáp ứng được. (xét trường hợp tiến trình có độ ưu tiên thấp hơn trong vùng găng và tiến trình có độ ưu tiên cao đợi vào vùng găng.)

Giải pháp gọi

V.6.1.4 Giải pháp gọi lời gọi hệ thống SLEEP vào WAKEUP sẽ làm blocked tiến trình đợi vào vùng găng

SLEEP: Chuyển tiến trình gọi nó về trạng thái blocked cho đến khi tiến trình khác gởi đến nó tín hiệu đổi trạng thái (WAKEUP)

WAKEUP(Process) : Chuyển tiến trình Process về trạng thái Ready (tiến trình đã gọi SLEEP trước đó)

Xét bài toán : "Sản xuất – tiêu thụ ": Có hai tiến trình SảnXuất và TiêuThụ dùng chung buffer có kích thước cố định. Tiến trình SảnXuất đặt sản phẩm vào buffer nếu buffer còn trống. Tiến trình TiêuThụ lấy sản phẩm từ buffer nếu buffer khác rỗng.

```
#define  N  100
int count=0;
void SảnXuất(void)
{
        int item;
        while (TRUE)
```

```
        {
                SảnXuấtSảnPhẩm(&Item);
                if (count == N)  SLEEP();
                ĐặtSảnPhẩm(Item);
                count++;
                if (count == 1)  WAKEUP(TieuThụ);
        }
}


void  TiêuThụ(void)
        {
                int Item;
                while( TRUE)
                {
                        if (count == 0)  SLEEP();
                        LấySảnPhẩm(&Item);
                        count--;
                        if (count == N-1)  WAKEUP (SảnXuất);
                        TiêuThụsảnPhẩm(Item);
                }
        }
```

Với giải pháp này hệ thống có thể dẫn đến tình trạng Deadlock tức cả hai tiến trình đều rơi vào trạng thái Block, không có tiến trình wakeup do sử dụng biến dùng chung count không được thực hiện theo thao tác nguyên tử. Kết quả là tín hiệu WAKEUP bị mất khi tiến trình được WAKEUP chưa thật sự SLEEP.

- Cần duy trì một biến đếm cho mỗi tiến trình để đếm tín hiệu WAKEUP được gởi đến từ tiến trình khác . Mỗi khi gọi SLEEP tiến trình kiểm tra biến đếm , nếu biến đếm >0 thì giảm biến đếm xuống 1 và tiến trình vẫn không bị blocked.
- Đó chính là ý tưởng để xây dựng khái niệm Semaphore

V.6.1.5 Semaphore

Semaphore là một kiểu nguyên không âm. Một semaphore s =0 chỉ ra rằng không tín hiệu WAKEUP nào được gởi đến. Có hai thao tác nguyê tử trên semaphore được định nghĩa như sau:

DOWN(s) : if (s >0) s=s-1;

else SLEEP();

UP(s): if (s >0) s=s+1;

else nếu có một hoặc nhiều tiến trình đang bị Blocked trên semaphore s . Hệ điều hành chọn ngẫu nhiên một tiến trình cho phép thoát khỏi trạng thái Blocked. (Trong khi đó s vẫn =0.)

ngược lại: không có tiến trình nào Blocked trên s thì: s= s+1 ;

- Tất cả công đoạn kiểm tra giá trị s, thay đổi s, gọi SLEEP được tích hợp thành một thao tác duy nhất không phân chia được- gọi là thao tác nguyên tử .

- Một semaphore s được khởi tạo =1 và được sử dụng bởi nhiều tiến trình để đảm bảo chỉ một trong chúng là vào được vùng găng tại một thời điểm gọi là **semaphore nhị phân**. Vì vậy mỗi tiến trình chỉ cần gọi toán tử DOWN(s) trước khi vào vùng găng và gọi UP(s) sau khi ra khỏi vùng găng thì có thể đảm bảo được sự loại trừ lẫn nhau.
- Các loại semaphore khác gọi là **semaphore đồng bộ hóa**, nó đảm bảo một dãy các sự kiện nào đó là xuất hiện hoặc không xuất hiện.

- **Áp dụng Semaphore để giải quyết bài toán Sản suất – tiêu thụ**

```
#define  N  100

typedef  int Semaphore;

Semaphore  Mutex = 1;

Semaphore Empty =N;

Semaphore Full = 0;

void SảnXuất (void)

{

        int Item;

        while(TRUE)

        {

        SảnXuấtSảnPhẩm(&Item);

        down(&Empty);

        down(&Mutex);

        ĐặtSảnPhẩm(Item);

        up(&Mutex);
```

```
            up(&Full);
            }
        }


void TiêuThụ (void)
        {   int Item;
            while(TRUE)
            {   down(&Full);
                down(&Mutex);
                LấySảnPhẩm(&Item);
                up(&Mutex);
                up(&Empty);
                TiêuThụsảnPhẩm(Item);
            }
}
```

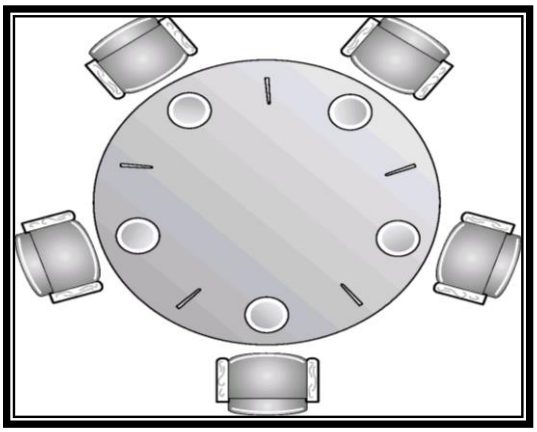
V.6.2 Áp dụng Semaphore để giải quyết bài toán cổ điển

V.6.2.1 Bài toán" Bữa ăn tối của các nhà hiền triết"

Có 5 nhà hiền triết ngồi quanh một bàn tròn trong một bữa ăn tối. Mỗi người có một dĩa mì Spaghetti. Mỗi người cần phải có 2 nĩa để có thể ăn mì. Giữa 2 dĩa có một nĩa.

Giả định rằng cuộc đời của nhà hiền triết chỉ luân phiên nhau 2 hành vi: ăn và suy nghĩ. Khi nhà hiền triết cảm thấy đói ông ta muốn lấy 2 nĩa bên trái và phải theo thứ tự nào đó. Nếu lấy được cả 2 nĩa ông ta bắt đầu ăn. Sau đó đặt nĩa xuống và tiếp tục suy nghĩ.

Yêu cầu viết chương trình cho mỗi nhà hiền triết sao cho không bị "kẹt".



```
#define  N  5

    void  HiềnTriết (int i)

    {

        while(TRUE)

        {

            SuyNghĩ();

            LấyNĩa(i);
```

```
            LấyNĩa((i+1)%N); // Nhà hiền triết I lấy nĩa bên trái, phải
            Ă n();
            ĐặtNĩa(i);
            ĐặtNĩa((i+1)%N);
        }
    }
```

Lời giải 1

```
    #define  N  5
    typedef int Semaphore;
    Semaphore Mutex=1;
    void  HiềnTriết (int i)
    {       while(TRUE)
            {       SuyNghĩ();
                    down(&Mutex);
                    LấyNĩa(i);
                    LấyNĩa((i+1)%N); // Nhà hiền triết I lấy nĩa bên trái, phải
                    Ă n();
                    ĐặtNĩa(i);
                    ĐặtNĩa((i+1)%N);
                    up(&Mutex);
            }
```

```
}
```

Lời giải trên đúng nhưng không tối ưu tài nguyên – tại một thời điểm chỉ có 1 nhà hiền triết ở trạng thái ăn trong khi đó có thể có 2

Lời giải 2

```
#define  N  5
#define LEFT (i-1)%N
#define RIGHT (i+1)%N
#define THINKING 0
#define HUNGRY   1
#define EATING     2
typedef int Semaphore;
int State [N];
Semaphore Mutex=1;
Semaphore S[N];// Khởi gán =0

void  HiềnTriết (int i)
{       while(TRUE)
        {       SuyNghĩ();
                LấyNĩa(i);// Lấy nĩa trái và phải
                Ă n();
```

```
                ĐặtNĩa(i);
        }
}
void LấyNĩa (int i)
{       down (&Mutex);
        State[i]=HUNGRY;
        Test(i);
        up(&Mutex);
        down(&S[i]);
        return ;
}

void ĐặtNĩa (int i)
{       down (&Mutex);
        State[i]=THINKING;
        Test(LEFT);
        Test(RIGHT);
        up(&Mutex);
        return ;
}

void Test (int i)
```

```
{       if( State[i]==HUNGRY  && State[LEFT]!=EATING &&
                State[RIGHT]!=EATING)
        {       State[i]=EATING;
                up(&S[i]);
        }
}
```

V.6.2.2 Bài toán" Độc giả và nhà văn"

Một cơ sở dữ liệu mà tiến trình muốn đọc(độc giả) hoặc ghi lên đó (nhà văn) . Hệ thống cho phép đồng thời có nhiều tiến trình đọc cơ sở dữ liệu nhưng chỉ duy nhất một tiến trình ghi lên CSDL tại một thời điểm. Khi có một tiến trình ghi lên CSDL thì không có tiến trình nào được phép truy cập đến CSDL kể cả tiến trình đọc .

Yêu cầu: Lập trình cho hai tiến trình "Độc giả " và "nhà văn"

```
Seamphore  Mutex =1;
Semaphore  Db=1;// Truy cập vào DBF của tiến trình Writer
int  rc; // Đếm số tiến trình đọc

void Reader (void )
{       while (TRUE)
        {       down (Mutex);
```

```
            rc= rc+1;
            if ( rc==1)
                down(db);
            up(Mutex);
            ReadDBF();
            down(Mutex);
            rc=rc-1;
            if (rc==0)
                up(db);
            up(Mutex);
        }
}

void Writer (void )
{       while (TRUE)
        {       CreateData();
            down(db);
            WriteData();
            up(db);
        }
}
```

CÂU HỎI VÀ BÀI TẬP

1. Trình bày khái niệm tiến trình và giải thích quá trình chuyển đổi giữa các trạng thái tiến trình?
2. Trình bày mục tiêu của bộ lập lịch tiến trình?
3. Hãy tính thời gian chờ được xử lý trung bình của các tiến trình theo thuật toán Round Robin với Q=3

Tiến trình	Thời điểm vào	Thời gian xử lý
P1	0	10
P2	1	5
P3	4	7
P4	5	8

Hãy tính thời gian chờ được xử lý trung bình của các tiến trình theo thuật toán độ ưu tiên không độc quyền kết hợp Round Robin(Q=3) đối với các tiến trình có cùng độ ưu tiên.

Biết rằng:

- Độ ưu tiên của P1, P2, P3, P4 lần lượt là 2, 3,1, 2. (1>2>3)
- Sau mỗi lần xử lý độ ưu tiên của tiến trình giảm đi 1.

4. Hãy tính thời gian chờ được xử lý trung bình của các tiến trình theo thuật toán độ ưu tiên không độc quyền , với độ ưu tiên 1>2>3 ….

Tiến trình	Thời điểm vào	Độ ưu tiên	Thời gian xử lý
P1	0	1	7
P2	1	3	5
P3	4	2	3
P4	5	1	8

5. Trình bày nguyên tắc trao đổi thông giữa các tiến trình. Lý do cần đồng bộ hoá các tiến trình?
6. Thảo luận sự vi phạm điều kiện (4) cần đảm bảo để thiết kế hệ điều hành cho phép nhiều tiến trình sử dụng vùng nhớ dùng chung một cách đúng đắn và hiệu quả của thuật toán luân phiên ngặt?
7. Trình bày nhược điểm của giải pháp đồng bộ hoá Peterson và giải pháp gọi lời gọi hệ thống Sleep() và wakeup() ?
8. Trình khái niệm Semaphore và định nghĩa 2 toán tử Down() và Up() trên semaphore ? Cho một ví dụ tình huống sử dụng semaphore nhị phân?
9. Sử dụng semaphore thực hiện đồng bộ hóa hai tiến trình SendMessage() và ReceiveMessage(). Biết rằng hai tiến trình trên sử dụng chung một hàng đợi Queue, tiến trình ReceiceMessage() nhận message từ bàn phím và ghi vào đầu hàng đợi và tiến trình SendMessage() lấy message từ hàng đợi Queue và gởi lên mạng máy tính. Kích thước của hàng đợi Queue là N, mỗi phần tử hàng đợi chứa được một message. Khi hàng đợi đầy tiến trình ReceiceMessage() phải tạm ngưng, khi hàng đợi rỗng tiến trình SendMessage() phải tạm ngưng.
10. Söû duïng semaphore taïo ra hai tieán trình sau sao cho nb<na<nb+10:

Tiến trình 1:

```
While(TRUE)
{
        na=na+1;
}
```

Tiến trình 2:

```
        While(TRUE)
{
        nb=nb+1;
}
```

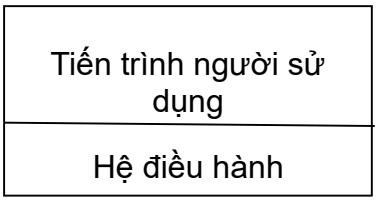
CHƯƠNG VII: HỆ THỐNG QUẢN LÝ BỘ NHỚ

VI.1 GIỚI THIỆU

- Bộ nhớ chính là thiết bị lưu trữ duy nhất thông qua đó CPU có thể trao đổi thông tin với môi trường ngoài.
- Nhu cầu tổ chức quản lý bộ nhớ là nhiệm vụ hàng đầu của hệ điều hành.
- Bộ nhớ chính được tổ chức như một mảng một chiều các ô nhớ. Việc trao đổi thông tin với môi trường ngoài được thực hiện thông qua các thao tác đọc ghidữ liệu vào địa chỉ cụ thể của bộ nhớ.
- Hầu hết các hệ điều hành hiện đại đều cho phép chế độ đa nhiệm nhằm nâng cao hiệu quả sử dụng CPU.

VI.2 QUẢN LÝ BỘ NHỚ KHÔNG PHÂN TRANG, KHÔNG SWAPPING

- Bộ nhớ chỉ được chia sẻ cho hệ điều hành và một chương trình duy nhất của người sử dụng. Tức tại một thời điểm một phần bộ nhớ sẽ do HĐH chiếm giữ và phần còn lại thuộc về một tiến trình người sử dụng, ctiến trình này có toàn quyền sử dụng vùng nhớ dành cho nó.
- Mô hình

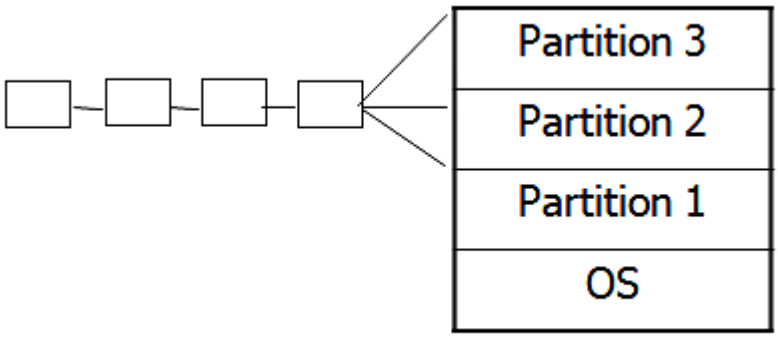


- Để bảo vệ vùng nhớ của HĐH khỏi sự xâm phạm của tiến trình người dùng cần phải tổ chức thanh ghi giới hạn.

- Địa chỉ cao nhất của vùng nhớ được cấp phát cho HĐH sẽ được nạp vào thanh ghi giới hạn.
- Tất cả các địa chỉ được tiến trình người dùng truy xuất đến sẽ được so sánh với nội dung thanh ghi giới hạn, nếu địa chỉ lớn hơn nội dung thanh ghi giới hạn thì đây là một địa chỉ hợp lệ, ngược lại một ngắt được phát sinh để thông báo cho hệ thống về một truy xuất bất hợp lệ.
- Với tổ chức như vậy thì chỉ có thể xử lý một chương trình duy nhất tại một thời điểm.
- Trong thực tế có rất nhiều tiến trình phải trải qua phần lớn thời gian chờ thao tác nhập xuất hoàn thành trong suốt thời gian này CPU nhàn rỗi ◊ để nâng cao hiệu suất sử dụng CPU cần cho phép chế độ đa chương.

VI.3 QUẢN LÝ BỘ NHỚ VỚI NHỮNG PHÂN ĐỌAN CỐ ĐỊNH

- Hệ điều hành chia bộ nhớ thành n vùng nhớ cố định(có thể không bằng nhau)
- Việc phân chia này được thực hiện vào lúc khởi động hệ thống và không thay đổi suốt quá trình chạy.
- Với tổ chức như vậy cần duy trì một hàng đợi duy nhất để lưu trữ những tiến trình chưa được cấp phát bộ nhớ.



- Tất cả tiến trình được đặt trong một hàng đợi duy nhất. Khi có một phân vùng tự do , tiến trình đầu tiên trong hàng đợi có kích thước phù hợp sẽ được đặt vào phân vùng này và cho xử lý.
- Nếu kích thước của tiến trình không vừa đúng bằng kích thước phân vùng chứa nó, phần bộ nhớ không sử dụng đến trong phân vùng sẽ bị lãng phí.
    - Xảy ra hiện tượng phân mảnh nội vi.
    - Mức độ đa chương của hệ thống bị giới hạn bởi số lượng phân vùng
- Vấn đề bảo vệ giữa các phân vùng: Để bảo vệ cần tổ chức hai thanh ghi : thanh ghi nền và thanh ghi giới hạn.
    - Khi tiến trình được tạo lập, nạp vào thanh ghi nền địa chỉ bắt đầu của phân vùng được cấp phát cho tiến trìnhvà nạp vào thanh ghi giới hạn kích thước của tiến trình
    - Sau đó mỗi đị chỉ bộ nhớ được phát sinh sẽ tự động được cộng với địa chỉ chứa trong thanh ghi nền để cho ra địa chỉ tuyệt đối trong bộ nhớ và các địa chỉ được đối chiếu với thanh ghi giới hạn để bảo đảm tiến trình không truy xuất ngoài phạm vi được cấp phát cho nó.

VI.4 QUẢN LÝ BỘ NHỚ VỚI NHỮNG PHÂN ĐỌAN ĐỘNG

- Như tổ chức quản lý bộ nhớ trên gây ra lãng phí bộ nhớ vì vậy một phương pháp mới được đề xuất là cấp pháp cho tiến trình vùng nhớ vừa đủ kích thước của tiến trình.
- Khi một tiến trình kết thúc vùng nhớ đã đã cấp cho nó sẽ được giải phóng và được cấp pháp cho tiến trình khác.
- Với giải pháp này không còn hiện tượng phân mảnh nội vi nhưng lại xuất hiện phân mảng ngoại vi. Khi các tiến trình lần lượt vào ra khỏi hệ thống, dần dần xuất hiện các khe hở giữa các tiến trình có thể dẫn đến tình huống tổng vùng nhớ còn trống đủ để thỏa mãn yêu cầu nhưng các vùng nhớ này không liên tục.

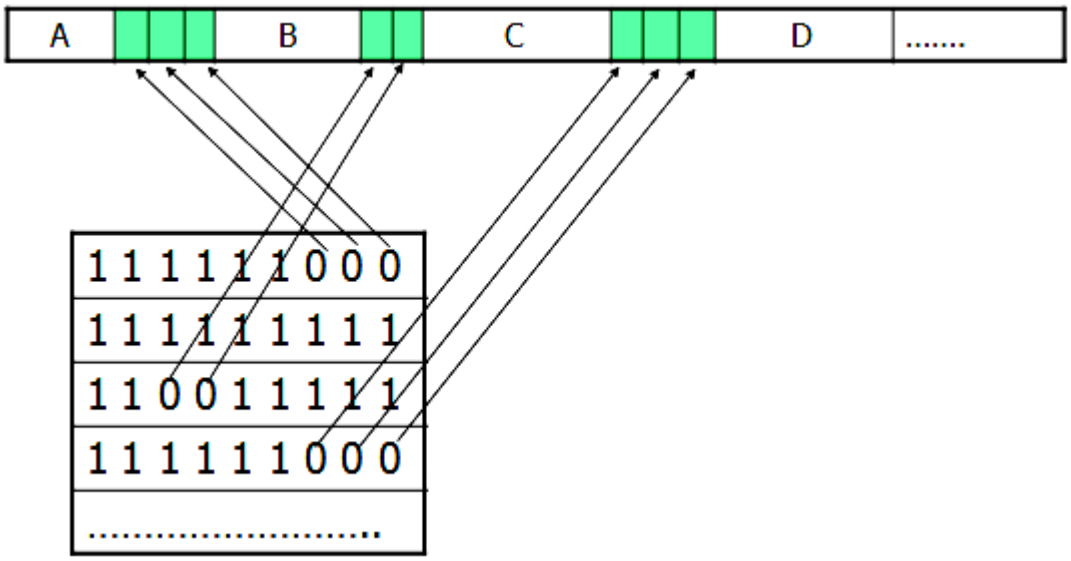
- Có thể áp dụng kỹ thuật dồn bộ nhớ để kết hợp các mảnh bộ nhớ rời rạc thành một vùng nhớ lớn liên tục.
- Một vấn đề khác nảy sinh khi kích thước của tiến trình tăng trưởng trong quá trình xử lý mà không còn vùng nhớ còn trông gần kề để mở rộng vùng nhớ cho tiến trình.

- Quản lý các ô nhớ còn trống: Cần phải lưu trữ thông tin các ô nhớ cò trống để cấp phát cho các tiến trình, Có hai phương pháp chính: Bitmap và danh sách liên kết.
- Phương pháp Bitmap:

Chia bộ nhớ thành từng đơn vị nhỏ (vài bytes) . Xây dựng bitmap, ứng với mỗi bit trong bitmap là một đơn vị bộ nhớ.

- Bit được đánh dấu là 1 khi đơn vị bộ nhớ tương ứng đã được cấp phát
- Bit được đánh dấu 0 khi đơn vị bộ nhớ tương ứng chưa được cấp phát.

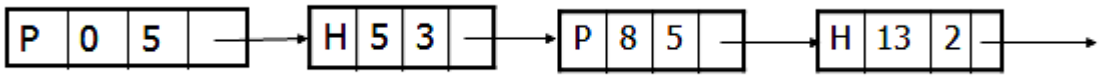
Thao tác cấp phát bộ nhớ là : Giả sử tiến trình cần k đơn vị bộ nhớ, HĐH duyệt bitmap và tìm ra k bit liên tiếp bằng 0



- Quản vùng nhớ còn trống bằng danh sách liên kết

Tổ chức một danh sách liên kết, mỗi phần tử tương ứng với một tiến trình hay lỗ hổng. Mỗi phần tử trong danh sách có 4 trường :

- Cờ biểu thị tiến trình (P) hay lỗ hổng (H)
- Địa chỉ bắt đầu của vùng nhớ tương ứng
- Kích thước của vùng nhớ
- Con trỏ Next

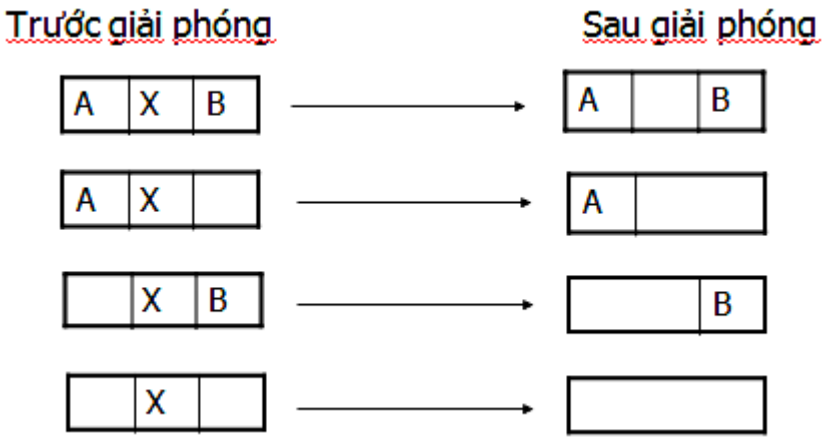


- **Thao tác cấp phát bộ nhớ**
  - First Fit : Xác định lỗ hổng đầu tiên trong danh sách có kích thước đủ lớn để cấp phát cho tiến trình và lỗ hổng này được chia làm 2 phần : 1 phần cho tiến trình và phần kia là lỗ hổng mới.
  - Best Fit : xác định lỗ hổng bé nhất có kích thước đủ lớn để cấp phát cho tiến trình.
  - Worst Fit : Cấp phát phân đoạn tự do lớn nhất đủ lớn để cấp phát cho riến trình.

Có thể làm tăng tốc độ bằng cách cho cả 3 thuật toán trên bằng cách tổ chức 2 danh sách: 1 cho tiến trình và một cho lỗ hổng. Tuy nhiên lại làm chậm thao tác giải phóng bộ nhớ.

- **Thao tác giải phóng bộ nhớ**

- Khi thực hiện thao tác giải phóng chú ý cần xem xét các trường hợp khác nhau của 2 phần tử kề nhau. Nhờ đó thực hiện thao tác kết hợp hai lỗ hổng kề nhau một cách phù hợp:



- Quá trình Swapping

Trong hệ thống đa nhiệm nhiều người dùng thường là bộ nhớ không đủ chỗ để lưu trữ tất cả các tiến trình vì thế hệ thống dùng DSLK để lưu trữ tạm thời một số tiến trình nào đó chưa thật sự chạy, khi được cấp phát CPU thì hệ thống phải mang nó vào bộ nhớ chính . Việc di chuyển tiến trình từ bộ nhớ vào đĩa và ngược lại gọi là quá trình swapping.

- Bộ nhớ ảo

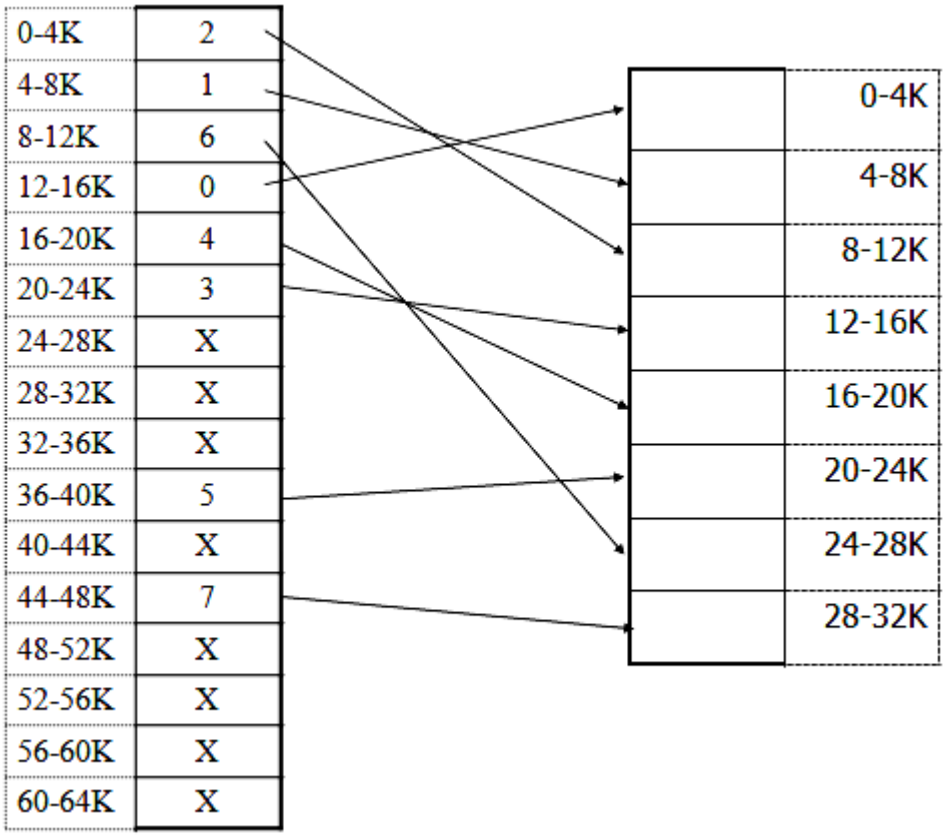
Bộ nhớ ảo được đưa ra nhằm khắc phục tình trạng kích thước chương trình vượt quá kích thước bộ nhớ vật lý. Hệ điều hành chia chương trình thành nhiều phần và giữa lại những phần chương trình đang chạy hiện thời vừa đủ chứa trong bộ nhớ, phần còn lại lưu trữ trên đĩa. HĐH theo dõi và quản lý tất cả công việc swapping giữa đĩa và bộ nhớ. Dĩ nhiên bộ nhớ ảo vẫn có thể thiết kế cho hệ thống đa nhiệm.

- Sự phân trang (paging)

Hầu hết những hệ thống có sử dụng bộ nhớ ảo đều sử dụng kỹ thuật phâh trang. Địa chỉ ảo là địa chỉ tham khảo từ chương trình. Tập tất cả địa chỉ ảo gọi là vùng địa chỉ ảo, đối với những hệ thống không dùng bộ nhớ ảo, địa chỉ ảo cũng là địa chỉ vật lý.

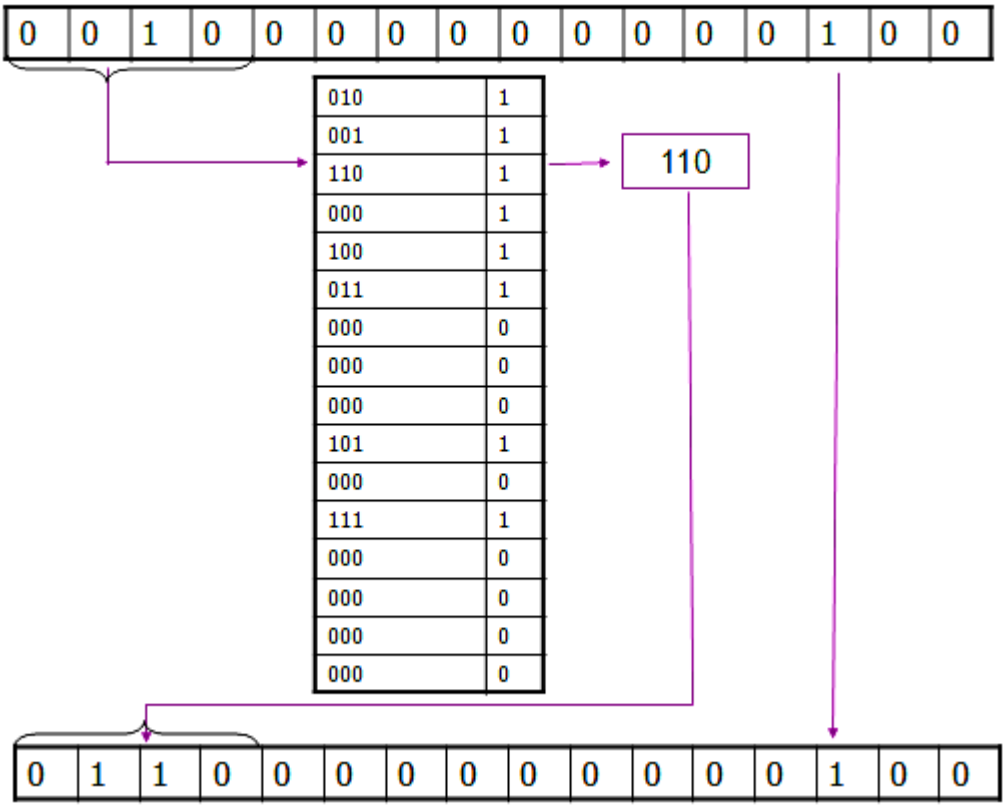
Vùng địa chỉ ảo được chi thành nhiều trang (page) có kích thước bằng nhau tướng ứng với những khung trang (Page frame) trong bộ nhớ vật lý. Trang và khung trang luôn có kích thước bằng nhau.

Ví dụ: có 32K bộ nhớ vật lý chia thành 8 khung trang có kích thước 4K và 64K bộ nhớ ảo chia thành 16 trang kích thước 4K. Trong bất kỳ thời điểm nào chỉ có 8 trang từ bộ nhớ ảo ánh xạ vào bộ nhớ vật lý như sau:



- Nếu lệnh nào đó trong chương trình tham khảo đến địa chỉ ảo không thuộc về bất kỳ trang nào đang được ánh xạ hiện thời gọi là lỗi trang. Trong trường hợp này HĐH tìm ra một khung trang nào ít sử dụng nhất trục xuất khỏi bộ nhớ và tìm kiếm trên đĩa trang nào chứa địa chỉ mà chương trình muốn tham khảo đến để chuyển vào bộ nhớ vật lý. Tiếp đến hệ thống cập nhật ánh xạ bộ nhớ cho trang mới.
- Bảng trang

Hệ thống phải duy trì một bảng trang thể hiện ánh xạ từ bộ nhớ ảo vào bộ nhớ vật lý và chứa thông tin trang nào hiện thời đang dược ánh xạ



VI.5 CÁC THUẬT TOÁN THAY THẾ TRANG

Mỗi khi gặp lỗi trang HĐH phải chọn 1 trong các trang náo đó trục xuất ra khỏi bộ nhớ. Nếu nội dung trang đó bị thay đổi yhì phải ghi nội dung mới của trang lên đĩa. Vấn đề là chọn trang nào để trục xuất để giảm tổn phí.

Xét ví dụ một tiến trình truy xuất đến một dãy các trang sau:

7, 0,1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

giả sử có 3 khung trang và ban đầu cả 3 khung trang đều rỗng.

VI.5.1 Thuật toán FIFO

- Ghi nhận thời điểm 1 trang được mang vào bộ nhớ chính khi cần thay thế trang , thì trang lâu nhất sẽ được chọn để trục xuất.

7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
	0	0	0	0	3	3	3	2	2	2	2	1	1	1	1	1	1	0	0
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
*	*	*	*		*	*	*	*	*	*			*	*			*	*	*

- Với thuật toán này có thể có trang được chọn để thay thế có thể chứa nhiều dữ liệu cần thiết thường xuyên được nạp vào sớm , do vậy khi bị chuyển ra bộ nhớ phụ sẽ nhanh chóng gây ra lỗi trang.

VI.5.2 Thuật toán tối ưu

- Thay thế trang sẽ lâu nhất được sử dụng trong tương lai

7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
*	*	*	*		*		*			*			*				*		

- Thuật toán này bảo đảm lỗi trang là ít nhất tuy nhiên thuật toán này không khả thi vì không thể biết trước dãy các trang được truy xuất theo thứ tự.

VI.5.3 Thuật toán lâu nhất chưa sử dụng (LRU)

- Với mỗi trang ghi nhận thời điểm cuối cùng trang được truy cập, trang được thay thế là trang lâu nhất chưa sử dụng.

7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
*	*	*			*		*	*	*	*			*		*		*		

- Thuật toán đài hỏi một cơ chế xác định thứ tự các trang theo thời điểm truy xuất cuối cùng. Dùng stack
    - Cần tổ chức một stack lưu trữ số hiệu các trang
    - Mỗi khi thực hiện một truy xuất đến 1 trang, số hiệu của trang sẽ được xóa khỏi vị trí hiện hành trong stack và đưa lên đỉnh stack.
    - Trang ở đỉnh stack là trang được chọn để thay thế.

VI.5.4 Thuật toán Not Recently Used (NRU)

- Thuật toán dựa trên sự tồn tại 2 bit gắn liền với mỗi trang để chỉ ra trang đó có vừa mới được tham khảo hay là được ghi lên hay không. Những bit này chứa trong bảng trang
    - ✓ Bit R =1: nếu trang đó truy cập ngược lại chưa được truy cập
    - ✓ Bit M =1 : Trang đó đã bị thay đổi ngược lại chưa bị thay đổi
    - Cứ mỗi lần truy cập bộ nhớ cần phải cập nhật lại các bit này
    - Khi một tiến trình bắt đầu cả 2 bit trong tất cả các trang được đặt bằng 0. theo một chu kỳ thời gian bit R được đặt lại bằng 0 để phân biệt những trang mới được truy cập với trang đã truy cập trước đó.
    - Khi xuất hiện một lỗi trang HĐH duyệt toàn bộ bảng trang và chia chúng thành 4 lớp dựa vào các bit R, M như sau:

Class 0: R=0, M=0

Class 1: R=0, M=1

Class 2: R=1, M=0

Class 3: R=1, M=1

Sau đó thuật toán chọn ra một trang để thay thế thuộc thuộc lớp thấp nhất khác rỗng.

CÂU HỎI VÀ BÀI TẬP

1. Trình bày hiện tượng phân mảnh nội vi và ngoại vi trong quá trình tổ chức quản lý bộ nhớ phân đoạn cố định và động?
2. Trình bày thuật toán cấp phát bộ nhớ (thuật toán first Fit, Best Fit, Worst Fit) trong kỹ thuật quản lý bộ nhớ động bằng danh sách liên kết và kỹ thuật bitmap?
3. Trình bày thuật toán giải phóng bộ nhớ trong kỹ thuật quản lý bộ nhớ động bằng danh sách liên kết?
4. Trình bày phương pháp chuyển đổi địa chỉ ảo sang địa chỉ vật lý trong phương pháp quản lý bộ nhớ ảo? Kích thước trang và khung trang, nội dung bảng trang như mục VI.4?
5. Trong một thống cài đặt bộ nhớ ảo theo kỹ thuật phân trang, giả sử một chương trình truy cập đến các trang sau: 1, 0,4, 2, 1, 3, 1, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 3, 0, 1. Giả sử hệ thống có 3 khung trang lúc đầu còn trống. Có bao nhiêu lỗi trang xảy ra khi hệ thống sử dụng thuật toán thay thế trang LRU, FIFO ?

---HẾT---

TÀI LIỆU THAM KHẢO

[1] **Giáo trình hệ điều hành 1, Lê Khắc Nhiên Ân**, Hoàng Kiếm

[2] **Giáo trình hệ điều hành 2, Lê Khắc Nhiên Ân**, Hoàng Kiếm

[3] **Hỗ trợ kỹ thuật lập trình hệ thống**, Nguyễn Tín

[4] **Virus tin học huyền thoại và thực tế**, Ngô Anh Vũ

[5] **Operating Systems**, Andrew S.Tanenbaum

[6] **Operating System Concepts**, Abraham Siberschatz, Peter B. Galvin