BM.05-QT.DT.04
15/3/12-REV:0

TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM
**KHOA CÔNG NGHỆ THÔNG TIN**


# BÀI GIẢNG

TÊN HỌC PHẦN : **NGUYÊN LÝ HỆ ĐIỀU HÀNH**
MÃ HỌC PHẦN : **17303**
HỆ ĐÀO TẠO **ĐẠI HỌC CHÍNH QUY**

**HẢI PHÒNG, 12/2013**

# MỤC LỤC

# YÊU CẦU VÀ NỘI DUNG CHI TIẾT

Tên học phần: **Nguyên lý Hệ điều hành** Mã học phần: **17303**

*a. Số tín chỉ:* *2 TC* **BTL** ☐ **ĐAMH** ☐

*b. Đơn vị giảng dạy:* Bộ môn **Kỹ thuật Máy tính**

*c. Phân bổ thời gian:*

- Tổng số (TS): 30 tiết. - Lý thuyết (LT): 28 tiết.
- Thực hành (TH): 0 tiết. - Bài tập (BT): 0 tiết.
- Hướng dẫn BTL/ĐAMH (HD): 0 tiết. - Kiểm tra (KT): 2 tiết.

*d. Điều kiện đăng ký học phần:*

Học phần học trước: Kiến trúc máy tính và TBNV, Cấu trúc dữ liệu và giải thuật.

*e. Mục đích, yêu cầu của học phần:*

Cung cấp cho sinh viên những khái niệm tổng quan về Hệ điều hành, các phương pháp tiếp cận khi thiết kế các hệ điều hành.

*f. Mô tả nội dung học phần:*

Học phần trang bị các kiến thức về: Các khái niệm cơ bản của một hệ điều hành, các tính chất chung và nguyên tắc xây dựng, các tài nguyên của hệ thống và các phương thức quản lí chúng: Quản lí thiết bị, quản lý bộ nhớ và quản lý các tiến trình trong hệ điều hành đơn Processor, đa Processcer.

*g. Người biên soạn:* **Nguyễn Trọng Đức – Bộ môn Kỹ thuật máy tính.**

*h. Nội dung chi tiết học phần:*

| TÊN CHƯƠNG MỤC | PHÂN PHỐI SỐ TIẾT | | | | | |
|---|---|---|---|---|---|---|
| | TS | LT | BT | TH | HD | KT |
| **Chương I. Những khái niệm cơ bản** | **7** | **7** | | | | |
| *1.1. Cấu trúc phân lớp và hệ thống tính toán* | | *1* | | | | |
| *1.2. Tài nguyên hệ thống* | | *2* | | | | |
| *1.3. Định nghĩa hệ điều hành* | | *0.5* | | | | |
| *1.4. Phân loại hệ điều hành* | | *1* | | | | |
| *1.5. Tính chất chung của hệ điều hành* | | *1* | | | | |
| *1.6. Nguyên tắc xây dựng hệ điều hành* | | *1* | | | | |
| *1.7. Thành phần hệ điều hành* | | *0.5* | | | | |
| **Chương II. Quản lí thiết bị** | **7** | **6** | | | | **1** |
| *2.1. Quan hệ phân cấp trong tổ chức và quản lý thiết bị ngoại vi* | | *1* | | | | |
| *2.2. Cơ chế phòng đệm* | | *1* | | | | |
| *2.3. Cơ chế SPOOL* | | *1* | | | | |
| *2.4. Quản lý file* | | *1* | | | | |
| *2.5. Quản lý file trong hệ điều hành MSDOS* | | *2* | | | | |

| TÊN CHƯƠNG MỤC | PHÂN PHỐI SỐ TIẾT | | | | | |
|---|---|---|---|---|---|---|
| | TS | LT | BT | TH | HD | KT |
| **Chương III. Quản lí bộ nhớ** | **6** | **6** | | | | |
| *3.1. Đặt vấn đề* | | *1* | | | | |
| *3.2. Quản lý bộ nhớ logic - cấu trúc một chương trình* | | *2* | | | | |
| *3.3. Quản lý bộ nhớ vật lý* | | *2* | | | | |
| *3.4. Quản lý bộ nhớ IBM PC của MSDOS* | | *1* | | | | |
| **Chương IV. Quản lí tiến trình** | **6** | **5** | | | | **1** |
| *4.1. Quản lý tiến trình* | | *3* | | | | |
| *4.2. Quản lý Processor* | | *2* | | | | |
| **Chương V. Hệ điều hành nhiều vi xử lí** | **4** | **4** | | | | |
| *5.1. Hệ điều hành nhiều Processor* | | *1* | | | | |
| *5.2. Hệ điều hành phân tán* | | *1* | | | | |
| *5.3. Quản lý tài nguyên trong hệ điều hành phân tán* | | *2* | | | | |

*i. Mô tả cách đánh giá học phần:*

- Thời gian học tập trên lớp phải ≥75% số tiết quy định của học phần.
- Thực hiện 02 bài kiểm tra do giảng viên chấm X=(KT1+KT2)/2 với điều kiện dự thi X≥4.
- Hình thức đánh giá học phần : Thi trắc nghiệm trên máy tính.
- Điểm đánh giá học phần: Z = 0,3X + 0,7Y (với Y là điểm thi kết thúc học phần).
- Thang điểm: Thang điểm chữ A+, A, B+, B, C+, C, D+, D, F.

*k. Giáo trình:*

1. Hồ Đức Phương, *Giáo trình Nguyên lý hệ điều hành*, NXB Giáo dục, 2011.

*l. Tài liệu tham khảo:*

1. Nguyễn Thanh Tùng, *Giáo trình Hệ điều hành*, ĐH Bách Khoa HN.
2. Hà Quang Thụy, *Giáo trình nguyên lý các Hệ điều hành*, NXB KHKT, 2009.
3. Trần Hồ Thủy Tiên, *Giáo trình Nguyên lý hệ điều hành*, Đại học Đà Nẵng, 2007.
4. Milan Milenkovic, *Operating systems concept and design*, Tata Mcgraw Hill, 2001.
5. Achyut Godbole, *Operating system*, Mc Graw Hill, 2010.
6. Andrew S. Tanenbaum, *Modern Operating system*, Prentice Hall.

# Chương I: NHỮNG KHÁI NIỆM CƠ BẢN

***Quan tâm của người dùng***

- Các hệ thống chương trình có cấu trúc như thế nào?
- Các hệ thống có đặc trưng gì?
- Hệ thống cung cấp cho người dùng những tài nguyên gì?

## 1.1. Cấu trúc phân lớp và hệ thống tính toán

Khi người dùng thực hiện một chương trình, hệ thống có đáp ứng được các yêu cầu hay không

*Bao gồm:*

- Hệ thống có chương trình cần thực hiện hay không
- Có đủ bộ nhớ để làm việc hay không
- Có các thiết bị ngoại vi theo yêu cầu hay không

Tuy nhiên yêu cầu của người dùng là đa dạng, khả năng của hệ thống có hạn nên đôi khi chi phí cho hệ thống khá cao song lợi ích mà hệ thống mang lại nhỏ.

Để khắc phục đưa ra giải **pháp tăng tính vạn năng của hệ thống qua processor**:

### *1.1.1. Cơ sở hoá hệ lệnh*

Trước đây trong máy tính đã lắp ráp nhiều vi mạch thực hiện các chức năng chuyên dụng tính căn, sin, e_mũ, loga.. vì vậy khi sử dụng rất khó có thể sửa chữa, thay đổi được.

Hiện nay các chức năng này đã được thay thế bằng phần mềm do đó máy tính vạn năng hơn, tốc độ cao hơn, độ ổn định và giá thành hạ.

Các Chương trình bao quanh phần kỹ thuật tạo thành một môi trường tính toán. Mỗi Chương trình muốn được thực hiện phải gắn với môi trường và thừa hưởng ở môi trường mọi khả năng của hệ thống. Làm cho thông tin lưu chuyển dễ dàng giữa các thành phần của hệ thống. Thông tin đầu ra của một module này có thể làm đầu vào cho một module khác. Mọi biến đổi trung gian đều do hệ thống đảm nhiệm và trong suốt với người sử dụng.

### *1.1.2. Tách thiết bị ngoại vi ra khỏi processor (micro hoá procesor)*

- Chuyển giao một số công việc cho thiết bị ngoại vi đảm nhiệm
- Processor tập trung xử lý bit
- Đề suất các thuật toán giải quyết các tác vụ trên bằng các phép xử lý bit, byte, hoàn thiện phương pháp xử lý trên máy tính điện tử
- Xây dựng sẵn các Modul chương trình cung cấp cho người dùng dưới dạng các chương trình chuẩn - thư viện các chương trình

Tuy nhiên trong thực tế khi các yêu cầu gia tăng thì các chương trình dưới dạng thư viện ngày càng tăng nên số lượng, nội dung của các thư viện tăng.

Giải pháp:

- Cung cấp cho người dùng các công cụ cho phép họ mô tả các giải thuật cần thiết, đồng thời cơ sở hoá các thư viện do đó ngôn ngữ thuật toán và chương trình dịch ra đời
- Người dùng có thể tác động lên máy tính điện tử thông qua các chương trình mẫu hoặc chương trình dịch

### *1.1.3. Chuyển nguyên tắc Lệnh thành Menu*

Cơ chế ra lệnh

- Người dùng phải tự nắm bắt trước các công việc mà hệ thống có thể làm được, qua đó chỉ thị cho hệ thống làm việc.

Cơ chế Menu

- Hệ thống giới thiệu cho người dùng các khả năng phục vụ của mình dưới dạng các bảng chọn, người dùng chỉ chờ cho hệ thống trình bày danh mục các công việc và lựa chọn công việc có thể yêu cầu
- Các công việc được phân nhóm theo từng phạm trù để dễ tìm kiếm
- Hệ thống mang tính chất tự đào tạo: càng làm việc càng hiểu sâu hơn

Nguyên tắc xây dựng Menu

*Bằng lời:*

- ✓ Dùng lời chỉ chính xác công việc sẽ thực hiện, tổ chức độ phân giải tốt
- ✓ Dễ thực hiện
- ✓ Chịu hàng rào ngôn ngữ

*Bằng biểu tượng:*

- ✓ Mỗi công việc được miêu tả bằng một hình ảnh
- ✓ Hấp dẫn, dễ hiểu với mọi loại đối tượng
- ✓ Chống được hàng rào ngôn ngữ
- ✓ Khó tổ chức và độ phân giải thấp

*Khắc phục nhược điểm của hai hình thức tổ chức trên: tổ chức cả hai hình thức:*

- ✓ Khi đưa hộp sáng hay khung tích cực tới một biểu tượng thì dòng chú thích xuất hiện
- ✓ Khi đưa hộp sáng hay khung tích cực áp vào một mục nào đó bằng lời thì biểu tượng xuất hiện

Ngoài ra còn tồn tại cơ chế phím nóng, lệnh chuẩn

***Tóm lại***: Hệ thống phải có trách nhiệm đảm bảo các điều kiện vật chất về các chương trình có thể thực hiện được đồng thời phải duy trì hệ thống ở trạng thái đồng bộ (có nghĩa là hệ thống phải có chức năng quản lý tài nguyên)

## 1.2. Tài nguyên hệ thống

Bao gồm:

- Không gian: Không gian nhớ
- Thời gian: Thời gian thực hiện lệnh
- Thiết bị ngoại vi

### *1.2.1. Bộ nhớ*

- Bộ nhớ là nơi lưu trữ thông tin.
- Đặc trưng bộ nhớ
  - ✓ Thời gian truy nhập
  - ✓ Phân cấp
  - ✓ Phân loại
- Thời gian truy nhập
  - ✓ Thời gian truy nhập trực tiếp: thời gian trực tiếp để truy nhập tới địa chỉ bất kỳ trong bộ nhớ.
  - ✓ Thời gian truy nhập tuần tự: Khi tồn tại một cách tổ chức lưu trữ kế tiếp.
- Phân cấp bộ nhớ
  - ✓ Bộ nhớ thường được phân cấp theo tốc độ truy nhập trực tiếp hay kế tiếp. Bộ nhớ được gọi là thực hiện nếu processor có thể thực hiện câu lệnh bất kỳ ghi trong đó. Đặc điểm của bộ nhớ này là thời gian truy nhập thực hiện và truy nhập tuần tự là bằng nhau. Bộ nhớ trong bao giờ cũng là bộ nhớ thực hiện.
  - ✓ Không gian bộ nhớ
  - ✓ Giá thành

- Phân loại bộ nhớ

- ✓ Bộ nhớ trong: Có tốc độ truy nhập cao nhưng không gian bộ nhớ nhỏ
- ✓ Bộ nhớ ngoài: Có không gian bộ nhớ lớn nhưng tốc độ truy nhập thấp.

Thời gian truy nhập trực tiếp thường lớn hơn thời gian truy tuần tự. Loại bộ nhớ phổ biến là bộ nhớ đĩa cứng, đĩa mềm, băng từ, đĩa quang.

### *1.2.2. Thời gian thực hiện lệnh*

- Processor là một tài nguyên quan trọng của hệ thống, được truy nhập ở mức câu lệnh và chỉ có nó mới làm cho câu lệnh được thực hiện.
- Processor được dùng cho nhiều tiến trình khác nhau do đó việc phân chia thời gian sử dụng processor của mỗi tiến trình phải được tối ưu hoá, đặc biệt là khi chúng còn dùng chung tài nguyên khác: Chương trình, dữ liệu, thiết bị vào ra...
- Thời gian: thời gian thực hiện một câu lệnh
- Trong hệ thống có nhiều processor thì thời gian của mỗi processor được quản lý và phân phối riêng biệt như những tài nguyên độc lập

### *1.2.3. Thiết bị ngoại vi*

- Số lượng nhiều
- Chủng loại đa dạng
- Tốc độ xử lý << tốc độ processor
- Các thiết bị tiếp nhận, lưu trữ thông tin ở bộ nhớ ngoài trong thời gian dài được gọi là thiết bị ngoại vi (Máy in, bàn phím, màn hình, chuột, modem,… ). Chúng còn được gọi là thiết bị vào ra. Chúng thường được gắn với MTDT thông qua các thiết bị trung gian (các thiết bị quản lý, thiết bị điều khiển).
- Tài nguyên có hai loại: Phân chia được và không phân chia được.
  - ✓ Phân chia được: Cho phép nhiều người hay Chương trình sử dụng nó một cách đồng thời. Điển hình là bộ nhớ(trong và ngoài): có thể nạp nhiều Chương trình vào bộ nhớ trong, hay 1 Chương trình sử dụng nhiều tệp trên đĩa cứng.
  - ✓ Không phân chia được: phần lớn các tài nguyên còn lại. Tuy nhiên có thể phân phối việc sử dụng chúng sao cho người sử dụng cảm giác như được phục vụ đồng thời.

## 1.3. Định nghĩa hệ điều hành

Hệ điều hành là một phần quan trọng của mọi hệ thống thông tin. Một hệ thống thông tin gồm 4 thành phần: phần cứng, hệ điều hành, Chương trình ứng dụng, người sử dụng

Phần cứng: CPU, bộ nhớ, thiết bị vào ra cung cấp các tài nguyên thông tin cơ sở.

Các Chương trình ứng dụng: Chương trình dịch, hệ thống cơ sở dữ liệu, trình soạn thảo văn bản . qui định cách sử dụng các tài nguyên đó để giải quyết những vấn đề của người sử dụng.

Hệ điều hành điều khiển và đồng bộ việc sử dụng phần cứng của các Chương trình ứng dụng phục vụ các người sử dụng khác nhau với các mục đích sử dụng phong phú đa dạng.

Ta có thể hiểu Hệ điều hành là Hệ thống các Chương trình đảm bảo các chức năng **giao tiếp người máy** và **quản lý tài nguyên hệ thống tính toán**.

Tuy nhiên đứng dưới các góc độ khác nhau nên có nhiều cách tiếp cận khác nhau khi định nghĩa về hệ điều hành:

### *1.3.1. Với người dùng*

Hệ điều hành là hệ thống chương trình tạo điều kiện để khai thác tài nguyên hệ thống tính toán một cách dễ dàng, thuận tiện

Người sử dụng khi thực hiện một Chương trình nào đó trên máy tính điện tử thì chỉ quan tâm đến việc hệ thống có đáp ứng được nhu cầu của họ hay không? Có Chương trình cần thực hiện, có đủ bộ nhớ để chạy Họ không quan tâm đến việc hệ điều hành làm gì nhằm mục đích gì, có cấu trúc như thế nào?

### *1.3.2. Với người quản lý*

Hệ điều hành là tập các chương trình phục vụ quản lý chặt chẽ và sử dụng tối ưu các tài nguyên hệ thống

### *1.3.3. Với cán bộ kỹ thuật*

Hệ điều hành là hệ thống chương trình trang bị cho một máy tính cụ thể mức vật lý để tạo ra một máy logic mới với các tài nguyên và khả năng mới.

### *1.3.4. Với cán bộ lập trình hệ thống*

Hệ điều hành là một hệ thống mô hình hoá mô phỏng các hoạt động của máy, của người dùng và của thao tác viên hoạt động trong chế độ đối thoại nhằm tạo môi trường khai thác thuận tiện và quản lý tối ưu các tài nguyên của hệ thống tính toán

Đối với các cán bộ lập trình hệ thống, vị trí của họ là ở bên trong hệ điều hành. Họ quan sát các module, các thành phần của hệ thống, quan sát mối quan hệ giữa chúng. Đây là quan điểm của chúng ta trong suốt quá trình khảo sát nghiên cứu hệ điều hành.

***Tóm lại:***

Hệ điều hành là một hệ chuyên gia ra đời sớm nhất và hoàn thiện nhất vì hai yếu tố:

- ✓ Vấn đề mà hệ điều hành giải quyết nảy sinh từ những người làm tin học do đó bài toán chính xác và rõ ràng.
- ✓ Người tham gia thiết kế chương trình là các cán bộ lập trình có tay nghề cao.

## 1.4. Phân loại hệ điều hành

Bao gồm:

- ✓ Hệ điều hành đơn nhiệm và hệ điều hành đa nhiệm
- ✓ Hệ điều hành đơn Chương và hệ điều hành đa Chương (MultiUsers)
- ✓ Hệ điều hành tập trung và hệ điều hành phân tán
- ✓ Hệ điều hành phân chia thời gian và hệ điều hành thời gian thực

### *1.4.1. Hệ điều hành đơn nhiệm và hệ điều hành đa nhiệm*

Dựa vào cách thức đưa Chương trình vào bộ nhớ, chọn Chương trình có sẵn trong bộ nhớ để processor thực hiện, người ta phân thành: hệ điều hành đơn nhiệm, đa nhiệm.

*Hệ điều hành đơn nhiệm*

- Tại một thời điểm xác định, khi một Chương trình được đưa vào bộ nhớ thì nó chiếm giữ mọi tài nguyên của hệ thống, và vì vậy Chương trình khác không thể được đưa vào bộ nhớ trong khi nó chưa kết thúc.
- Nhưng do các thiết bị vào ra thường làm việc với tốc độ chậm, người ta dùng kỹ thuật SPOOLING (simultanous peripheral Operation on line): cho phép tạo ra hiệu ứng song song các thiết bị chỉ cho phép vào ra tuần tự (sẽ đề cập chi tiết ở Chương sau).

*Hệ điều hành đa nhiệm*

- Hệ điều hành cho phép tại một thời điểm có nhiều Chương trình ở trong bộ nhớ trong. Chúng có nhu cầu được phân phối thời gian phục vụ CPU, bộ nhớ và thiết bị ngoại vi. Như vậy CPU, bộ nhớ, thiết bị ngoại vi v.v.. là các tài nguyên được chia sẻ cho các Chương trình đó. Vấn đề là làm sao đảm bảo tốt nhất tính bình đẳng khi giải quyết vấn đề phân phối tài nguyên.

### *1.4.2. Hệ điều hành đơn Chương và hệ điều hành đa Chương (MultiUsers)*

*Hệ điều hành đơn chương*

- Tại một thời điểm xác định hệ điều hành chỉ cho phép một người sử dụng thao tác mà thôi.

*Hệ điều hành đa chương*

- Hệ điều hành cho phép tại một thời điểm có thể phục vụ nhiều người sử dụng.

### *1.4.3. Hệ điều hành tập trung và hệ điều hành phân tán*

*Hệ điều hành tập trung*

- Trên một hệ thống máy tính chỉ có một HĐH duy nhất cài ở máy chủ. Các máy trạm được khởi động nhờ máy chủ và nó chỉ làm chức năng nhập/xuất dữ liệu. Mọi xử lý đều tập trung ở máy chủ.

*Hệ điều hành phân tán*

- Trên mỗi máy có 1 hệ điều hành khác nhau, máy chủ chịu trách nhiệm cung ứng các dịch vụ để truy nhập đến các tài nguyên chung và điều hành toàn
- hệ thống, các phép xử lý có thể tiến hành ở máy trạm.

### *1.4.4. Hệ điều hành phân chia thời gian và hệ điều hành thời gian thực*

*Hệ điều hành phân chia thời gian (Share time)*

- Một CPU luôn phiên phục vụ các tiến trình và 1 tiến trình có thể rơi vào trạng thái chờ đợi khi chưa được phân phối CPU.

*Hệ điều hành thời gian thực (Real time)*

- Một tiến trình khi đã xâm nhập vào hệ thống thì ở bất kỳ lúc nào đều được phân phối CPU.

## 1.5. Tính chất chung của hệ điều hành

### *1.5.1. Độ tin cậy cao*

Mọi hoạt động thông báo của hệ điều hành chuẩn xác tuyệt đối

Khi chắc chắn đúng thì máy mới cung cấp thông tin cho người dùng

Mọi công việc bao giờ cũng được kiểm tra, đánh giá

***Ví dụ: C:\>COPY A:\ F1.TXT B:***

Kiểm tra lệnh COPY

Kiểm tra các điều khiển

Tồn tại hay không các ổ đĩa

Động cơ có quay không

Đĩa có truy nhập được không

Tồn tại hay không tệp tin f1.txt

Chất lượng thông tin trên đĩa như thế nào?

Đọc một phần thông tin trong F1.TXT hay toàn bộ

…

### *1.5.2. Độ an toàn*

Tổ chức cho dữ liệu và chương trình không bị xoá hoặc thay đổi ngoài ý muốn.

Chức năng bảo vệ thông tin được chia thành nhiều mức:

- *Các mức do hệ thống đảm nhiệm:* Ví dụ: trong các hệ thống UNIX, khi muốn xoá hay sửa đổi nội dung một tệp, người sử dụng phải có quyền xoá sửa đối với file đó.
- *Các mức do người sử dụng đảm nhiệm:* Ví dụ: Lệnh DEL *.* của MSDOS, hệ thống hỏi lại người sử dụng một lần nữa để tránh sai sót vô ý.

### *1.5.3. Hiệu quả*

Các tài nguyên phải được khai thác triệt để ngay cả khi điều kiện tài nguyên hạn chế song vẫn có thể giải quyết các yêu cầu phức tạp.

Tính đồng bộ cao (duy trì đồng độ trong toàn bộ hệ thống)

### *1.5.4. Tổng quát*

Tính kế thừa các phiên bản trước đây

Thích nghi với những thay đổi có thể có trong tương lai

### *1.5.5. Thuận tiện*

- Dễ sử dụng
- Có nhiều mức hiệu quả khác nhau tuỳ kinh nghiệm và kiến thức người dùng:
  - ✓ Giao tiếp dạng dòng lệnh
  - ✓ Giao tiếp dạng thực đơn (Menu)
  - ✓ Giao tiếp dạng biểu tượng

## 1.6. Nguyên tắc xây dựng hệ điều hành

### *1.6.1. Modul*

Xây dựng từ các Modul độc lập quan hệ với nhau thông qua dữ liệu Vào/ra

Tồn tại cơ chế liên kết các Modul độc lập thành hệ thống có tổ chức

### *1.6.2. Nguyên tắc tương đối trong định vị*

Các Modul được viết theo địa chỉ tương đối kể từ đầu bộ nhớ, khi thực hiện chúng được định vị tại vùng nhớ cụ thể như vậy hệ thống sử dụng bộ nhớ linh hoạt hơn và hệ điều hành không phụ thuộc vào cấu hình bộ nhớ

### *1.6.3. Macroprocessor*

Khi có một công việc cụ thể, hệ thống sẽ:

- ✓ Xây dựng các phiếu yêu cầu
- ✓ Liệt kê các bước phải thực hiện
- ✓ Xây dựng chương trình tương ứng
- ✓ Thực hiện chương trình

Ví dụ: Trong MSDOS ta có các tệp config.sys và autoexec.bat

### *1.6.4. Phủ chức năng*

Một công việc của hệ điều hành có thể được thực hiện bằng nhiều phương tiện khác nhau cho phép người dùng chọn giải pháp tối ưu với bài toán của mình

Ví dụ: Khi in tệp F1.TXT có các giải pháp:

```
C:\>COPY  F1.TXT  PRN
C:\>TYPE  F1.TXT  >PRN
C:\>PRINT  F1.TXT
```

### *1.6.5. Giá trị chuẩn (ngầm định):*

Hệ thống chuẩn bị sẵn các bảng giá trị cho các tham số điều khiển

Nếu trong các câu lệnh của người dùng còn thiếu những tham số giá trị thì hệ thống sẽ tự động lấy giá trị tương ứng ở bảng giá trị chuẩn ra để thực hiện

Ví dụ: C:\BT> DIR

| | |
|---|---|
| Xem ổ đĩa nào: | C |
| Thư mục nào: | BT |
| Cái gì: | Mọi thư mục con, tệp trong thư mục này và không bị che |
| Như thế nào: | Đầy đủ thông tin, liên tục theo dữ liệu |
| Ra đâu: | Thiết bị chuẩn |
| Tham số: | Mọi tham số |

### *1.6.6. Tham số*

- Tham số vị trí: Là loại tham số mà ý nghĩa của nó xác định bởi vị trí xuất hiện trong bảng tham số. Đứng đầu dòng tham số

- Tham số khoá: Là loại tham số mà ý nghĩa xác định bằng từ khóa

Ví dụ: C:\>DIR D: /W/A/P

C:\>DIR D: /A/P/W

Trong đó:

D: là tham số vị trí

/W, /A hay /P là tham số khoá

### *1.6.7. Nguyên lý bảo vệ*

- Chương trình và dữ liệu phải được bảo vệ nhiều mức, bằng nhiều khoá.
- Ví dụ trong Linux
  + Mức 1: Người sử dụng phải có tài khoản mới được sử dụng máy tính.
  + Mức 2: Chỉ những người sử dụng thuộc nhóm A mới được truy nhập và tệp chung của nhóm A.

## 1.7. Thành phần hệ điều hành

### *1.7.1. Thành phần của hệ điều hành*

- Ngôn ngữ làm việc và giao tiếp: Hệ điều hành có quan hệ với ba đối tượng nên tồn tại ba ngôn ngữ làm việc và giao tiếp
  - ✓ *Ngôn ngữ máy (Ngôn ngữ thực hiện)*:
    Là ngôn ngữ thực hiện duy nhất của hệ thống. Mọi ngôn ngữ khác đều phải được ánh xạ sang ngôn ngữ thực hiện
  - ✓ *Ngôn ngữ vận hành (hệ điều hành):*
    Thao tác viên giao tiếp với hệ thống
  - ✓ *Ngôn ngữ thuật toán:*
    Người dùng giao tiếp với hệ thống: Pascal, C... (Cần phải có chương trình dịch).
- Các Modul chương trình của hệ thống có thể chia thành hai lớp:
  - ✓ *Chương trình điều khiển:*
    + Quản lý tài nguyên
    + Quản lý tiến trình
    + Quản lý, tổ chức dữ liệu
    + Chương trình thư ký, điều phối nhiệm vụ
  - ✓ *Chương trình phục vụ:*
    + Chương trình biên tập
    + Chương trình dịch

### *1.7.2. Thành phần của MSDOS*

Những năm 1980, khi hãng Intel cho ra đời bộ vi xử lý 16 bít 8086, Jim Paterson xây dựng hệ điều hành trang bị cho loại máy tính sử dụng bộ vi xử lý này đó là 86-DOS.

Hãng Microsoft đã mua lại hệ điều hành của Jim Paterson và phát triển thành hệ điều hành PC-DOS hay MSDOS. Phiên bản đầu tiên của MSDOS thế hệ 1.0 ra đời vào 8/1981.

- Các cải tiến cơ bản của MSDOS 1.0
  - ✓ Có thêm loại Chương trình chạy EXE bên cạnh các Chương trình COM.
  - ✓ Hệ điều hành đã tách bộ xử lý lệnh thành một phần nội trú và một phần ngoại trú.
  - ✓ Để tiện lợi cho việc quản lý đĩa người ta đưa ra bảng File Allocation Table viết tắt là FAT để quản lý đĩa. Mỗi phần tử của bảng FAT tương ứng với 521 byte trên đĩa gọi là sector, chỉ ra sector này đã có dữ liệu hay còn tự do.

- ✓MSDOS 1.0 cho phép xử lý lô (batch) một số lệnh của MSDOS bằng cách tạo một tệp batch.
- ✓ Ngày tháng tạo hay cập nhật tệp cũng được lưu trữ cùng với thông tin của tệp.

- Cùng với thời gian, hãng Microsoft đã nâng cấp hệ điều hành này lên các phiên bản mới 2.0, 3.0, 4.0…

- Các thành phần của MSDOS
  - ✓BIOS: Chứa các Chương trình của supervisor và quản lý tệp nhưng chưa kết nối thành hệ thống. Do đó cần Chương trình kích hoạt.
  - ✓Chương trình mồi Boot Strap Loader: nằm ở sector đầu tiên của đĩa từ dùng để kích hoạt toàn bộ Chương trình hệ thống.
  - ✓IO.SYS: Dưới sự hỗ trợ của BSL bao lấy BIOS, cung cấp các dịch vụ cơ bản nhất như chia sẻ tài nguyên, quản lý bộ nhớ.
  - ✓MSDOS.SYS: mở rộng IO.SYS lần nữa
  - ✓COMMAND.COM: liên lạc giữa người sử dụng và hệ thống, chứa các lệnh nội trú.
  - ✓Các lệnh ngoài: là thành phần mở rộng theo từng lĩnh vực.
  - ✓Các tiện ích khác: Chương trình nén đĩa (DBLSPACE)…

## CÂU HỎI VÀ BÀI TẬP

1.1. Hãy liệt kê sơ bộ về một số đặc trưng của các hệ điều hành đã sử dụng.

1.2. Trình bày các đặc trưng của CPU, bộ nhớ, kênh dẫn

1.3. Những đại lượng nào liên quan đến tốc độ xử lý của CPU

1.4. Anh, chị hãy lấy ví dụ minh họa về các tính chất của hệ điều hành đang sử dụng cụ thể

1.5. Anh, chị hãy trình bày về các nguyên tắc xây dựng hệ điều hành. Lấy ví dụ minh họa cụ thể.

1.6. Anh, chị hãy lấy ví dụ minh họa về các thành phần cơ bản của hệ điều hành đang sử dụng cụ thể. Nêu ý nghĩa, tác dụng của các thành phần đó.

# Chương II: QUẢN LÝ THIẾT BỊ

***Đặt vấn đề***

- Thiết bị ngoại vi đã trở thành đối tượng làm việc của hệ điều hành khi hệ thống phức tạp
- Các thiết bị ngoại vi đảm nhiệm việc truyền thông tin qua lại giữa các bộ phận của hệ thống

⇨ Vì vậy vấn đề tổ chức thông tin, phương pháp truy nhập tới chúng như thế nào

***Đề cập:***

- Tổ chức thiết bị ngoại vi
- Chiến lược điều khiển
- Phương pháp phát hiện và xử lý lỗi

## 2.1. Quan hệ phân cấp trong tổ chức và quản lý thiết bị ngoại vi

### *2.1.1. Sự đa dạng của các thiết bị ngoại vi:*

- Chuẩn: bắt buộc
- Phụ: bổ sung

### *2.1.2. Quan hệ giữa vi xử lý với thiết bị ngoại vi*

- Vi xử lý không thể làm việc trực tiếp với các thiết bị ngoại vi
- Vi xử lý cùng với thiết bị ngoại vi thực hiện các thao tác vào/ra

⇨ Tồn tại cách tổ chức sao cho vi xử lý không phụ thuộc vào các biến động của thiết bị ngoại vi

***Nguyên tắc:***

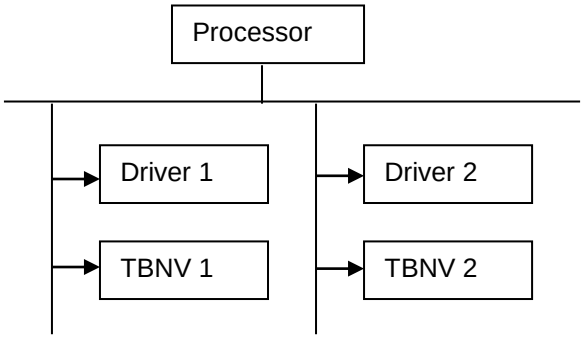
- Vi xử lý chỉ điều khiển các thao tác vào/ra chứ không trực tiếp thực hiện
- Các thiết bị ngoại vi không trực tiếp gắn vào vi xử lý mà gắn với thiết bị quản lý
- Một thiết bị điều khiển và các thiết bị ngoại vi phụ thuộc nó tạo thành một kênh (channel)

***Như vậy***: thiết bị quản lý đóng vai trò như một máy tính chuyên dụng:

- Nhiệm vụ điều khiển thiết bị ngoại vi
- Có ngôn ngữ riêng, lệnh riêng
- Thiết bị ngoại vi và thiết bị điều khiển hoạt động độc lập với nhau và độc lập với vi xử lý
- Chương trình viết trên ngôn ngữ thiết bị điều khiển và thiết bị ngoại vi gọi là chương trình kênh (channel program)

### *2.1.3. Thực hiện các phép vào/ra*

Vi xử lý tạo ra một chương trình tương ứng với công việc cần thực hiện, sau đó chuyển giao chương trình kênh và dữ liệu tương ứng cho thiết bị điều khiển và tiếp tục thực hiện chương trình của mình



Các phép vào/ra được điều khiển theo nguyên lý Macroprocessor cho phép trong lúc các phép vào/ra được thực hiện ở thiết bị ngoại vi thì vi xử lý vẫn hoạt động song song (thực hiện các tính toán và điều khiển khác khi chưa cần đến kết quả vào/ra)

Khi công việc được hoàn thành báo cho vi xử lý biết bằng tín hiệu ngắt. Tuỳ theo tín hiệu ngắt:

- ✓ Vi xử lý ngắt ngay
- ✓ Lưu trữ để chờ xử lý sau đó
- ✓ Huỷ bỏ

Để hệ thống có thể làm việc với các kênh vi xử lý phải biết ngôn ngữ kênh (ngôn ngữ được đưa vào hệ thống khi nạp hệ điều hành)

***Ví dụ: MSDOS***

Trong CONFIG.SYS

***DEVICE =...***

Đảm bảo tương tác chặt chẽ giữa thiết bị ngoại vi và vi xử lý thì kênh phát tín hiệu ngắt vào/ra, nó luôn luôn bảo vệ hệ thống một trị số qua đó có thể đánh giá chất lượng thực hiện phép vào/ra: mã trở về (return code). vi xử lý tạm dừng công việc của mình và chuyển sang phân tích mã trở về để đánh giá kết quả, chất lượng công việc

### *2.1.4. Kết thúc chương trình kênh*

Các lệnh trong chương trình kênh kết thúc khác nhau nên một phép vào/ra có thể thúc ở nhiều mức vì vậy kênh báo cho hệ thống biết kết quả phép vào/ra càng sớm càng tốt

Các chương trình ứng dụng, chương trình ngắt vào/ra, chương trình kênh tạo thành các tiến trình độc lập, hoạt động song song và chịu sự điều độ chung của hệ thống.

## 2.2. Cơ chế phòng đệm (Buffer)

Đặc điểm của thiết bị ngoại vi là tốc độ chậm (nhỏ hơn rất nhiều so với tốc độ của vi xử lý) do đó khi một thiết bị ngoại vi làm việc hệ thống cần:

- ✓ Kích hoạt thiết bị ngoại vi
- ✓ Chờ thiết bị ngoại vi đạt trạng thái thích hợp

Để đảm bảo hiệu suất sử dụng, hệ thống cần phải:

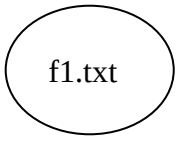
- ✓ Giảm số lượng các phép vào/ra vật lý
- ✓ Thực hiện song song các phép vào/ra và xử lý thông tin khác
- ✓ Thực hiện trước các phép nhập dữ liệu

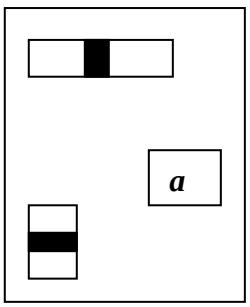
Như vậy tồn tại một số vùng nhớ trung gian làm nơi lưu trữ thông tin trong các phép vào/ra gọi là phòng đệm

- ✓ Cơ chế phòng đệm cho phép khắc phục:
- ✓ Thực hiện trước các phép nhập dữ liệu
- ✓ Tích luỹ kết quả ra
- ✓ Đảm bảo xử lý song song giữa các phép trao đổi vào/ra và xử lý
- ✓ Giảm số lần truy nhập vật lý
- ✓ Đảm bảo biến đổi topo thực hiện trước hoặc sau khi xử lý thông tin mà không làm mất tính liên tục của thông tin

Với vi xử lý thì phòng đệm chính là các thanh ghi

***Ví dụ:***



***Phân loại:***

- ✓ Phòng đệm trung gian
- ✓ Phòng đệm xử lý
- ✓ Phòng đệm vòng

### *2.2.1. Phòng đệm trung gian:*

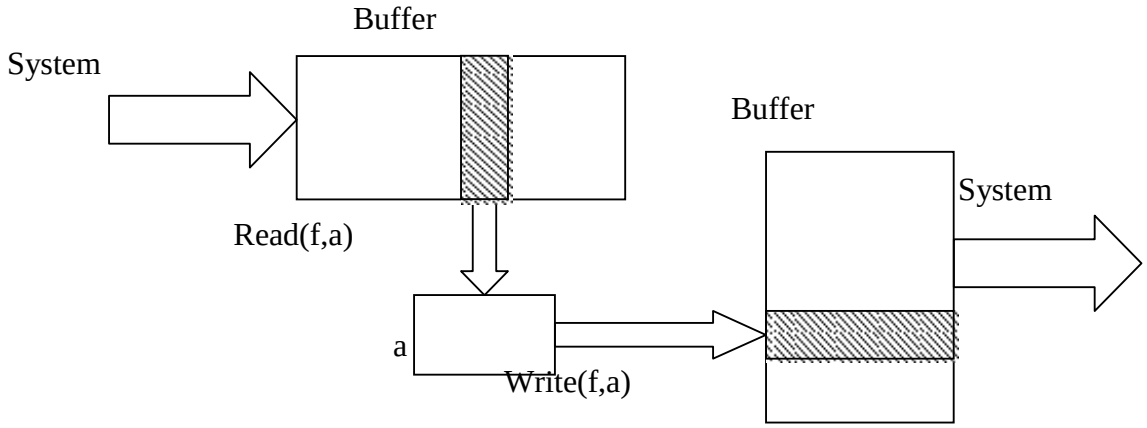
Lưu trữ tạm thời kết quả vào/ra như vậy tồn tại hai cơ chế phòng đệm:

Phòng đệm vào

- Phòng đệm chỉ dùng để nhập thông tin. Trong hệ thốngsẽ có lệnh để đưa thông tin vào phòng đệm (đọc vật lý).
- Khi gặp chỉ thị đọc (READ), thông tin sẽ được tách và chuyển từ phòng đệm vào các địa chỉ tương ứng trong Chương trình ứng dụng. Như vậy, mỗi giá trị được lưu trữ ở hai nơi trong bộ nhớ (một ở phòng đệm và một ở vùng bộ nhớ trong Chương trình ứng dụng). Khi giá trị cuối cùng của phòng đệm vào được lấy ra thì phòng đệm được giải phóng (rỗng) và hệ thống đưa thông tin mới vào phòng đệm trong thời gian ngắn nhất có thể.
- Để giảm thời gian chờ đợi, hệ thống có thể tổ chức nhiều phòng đệm vào, khi hết thông tin ở một phòng đệm, hệ thống sẽ chuyển sang phòng đệm khác.

Phòng đệm ra

- Khi có chỉ thị ghi (WRITE), thông tin được đưa vào phòng đệm. Khi phòng đệm ra đầy, hệ thống sẽ đưa thông tin ra thiết bị ngoại vi.
- Hệ thống cũng có thể tổ chức nhiều phòng đệm ra.




Ưu điểm:

- Đơn giản
- Hệ số song song cao (do tốc độ giải phóng vùng đệm lớn)
- Vạn năng, áp dụng cho mọi phép vào/ra

Nhược

- Tốn bộ nhớ
- Thời gian trao đổi
- Nhiều lỗi xử lý

### *2.2.2. Phòng đệm xử lý:*

Thông tin được xử lý ngay trong phòng đệm

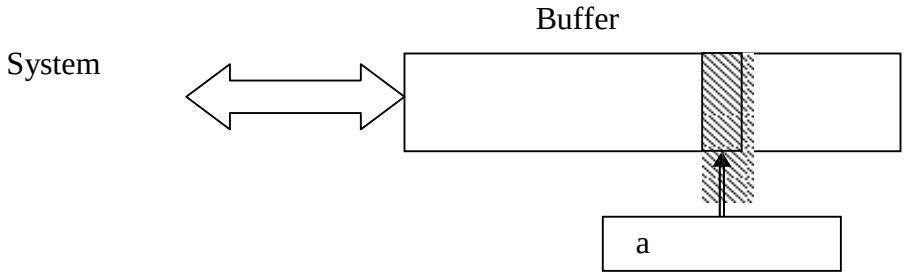
Truy nhập thông tin theo địa chỉ (tính địa chỉ của thông tin trong phòng đệm và cung cấp cho chương trình)

Ưu điểm:

- Tiết kiệm bộ nhớ
- Không mất thời gian chuyển thông tin ở bộ nhớ trong

Nhược:

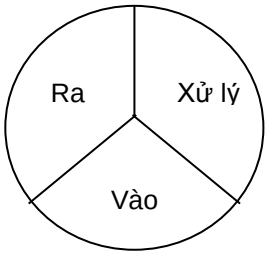
- Hệ số song song thấp
- Tốc độ giải phóng phòng đệm chậm
- Tính vạn năng không cao




### *2.2.3. Phòng đệm vòng*

Kết hợp cả 2 loại phòng đệm trên

Tổ chức 3 phòng đệm




Sau một khoảng thời gian ba phòng đệm quay vòng tròn

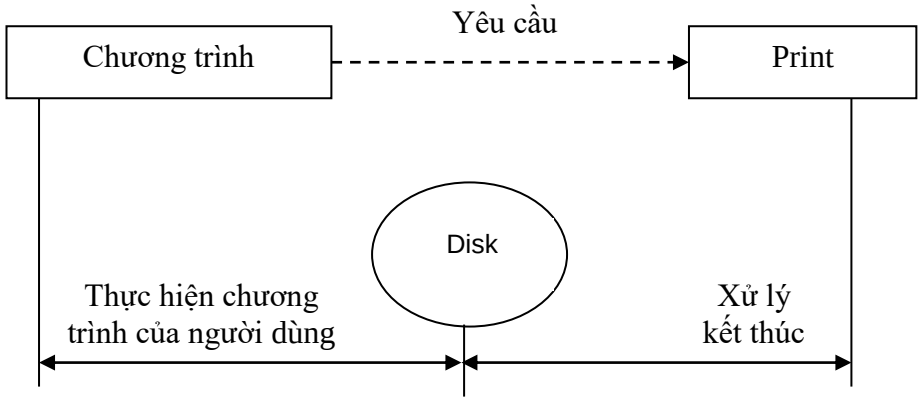
Tổ chức:

- Phòng đệm có thể gắn với từng file cụ thể: chỉ được xây dựng khi mở file hoặc đóng file
- Phòng đệm gắn với hệ thống: khi xây dựng hệ thống thì xây dựng ngay cơ chế phòng đệm và chỉ gắn vào một file cụ thể nào đó

## 2.3. Cơ chế SPOOL (Simultaneous Peripheral Operation On_Line - Hệ thống mô phỏng các phép trao đổi thiết bị ngoại vi trong chế độ trực tiếp)

Vai trò của thiết bị ngoại vi: trạm nhận chương trình kênh và dữ liệu, gửi các mã trạng thái cho hệ thống phân tích

Tuy nhiên: mọi chương trình và dữ liệu của thiết bị ngoại vi hoạt động tương tự như thiết bị ngoại vi có thực vì vậy có thể dùng phần mềm để mô phỏng hoạt động của thiết bị ngoại vi và coi nó như một thiết bị ngoại vi ảo.




Ứng dụng:

- Mô phỏng quá trình điều khiển, quản lý thiết bị ngoại vi
- Tạo ra các SPOOL, mô phỏng các phép trao đổi ngoại vi ngay trong lúc thực hiện

SPOOL: kỹ thuật xử lý mà thiết bị cuối trong chương trình của người dùng được tạm thời thay thế bởi thiết bị trung gian

- Sau khi kết thúc chương trình vào thời điểm thuận tiện thông tin sẽ được đưa ra thiết bị cuối theo yêu cầu của người dùng
- Không can thiệp vào chương trình của người dùng
- Tiến hành ngay trong lúc thực hiện phép trao đổi vào/ra

Tác dụng:

- Làm cho chương trình của người dùng thực hiện nhanh hơn
- Giảm giá thành chi phí
- Khai thác thiết bị ngoại vi tốt hơn
- Giảm yêu cầu về số lượng thiết bị
- Tạo ra kỹ thuật lập trình tương ứng

Tổ chức SPOOL: cơ chế thực hiện:

- Lưu kết quả đưa ra ở thiết bị trung gian, chuyển giao kết quả này ra phần xử lý kết thúc
- Lưu giữ chương trình kênh

## 2.4. Quản lý file

***Lý do:***

- Người dùng phải lưu trữ thông tin ở bộ nhớ ngoài vì vậy hệ điều hành phải có vai trò sao cho người dùng truy nhập thuận tiện
- Nhu cầu dùng chung các file dữ liệu

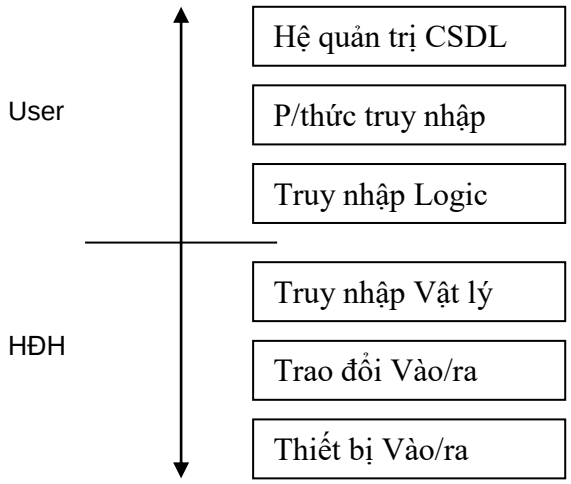
***Hệ quản lý file phải có các tính chất:***

- Tính độc lập của file với vi xử lý và với thiết bị ngoại vi vì vậy hệ thống khi làm việc phải quản lý file theo tên
- Bảo vệ dữ liệu: không để mất thông tin khi có sự cố kỹ thuật hoặc chương trình thậm chí truy nhập bất hợp lệ
- Tổ chức có hiệu quả đảm bảo tiết kiệm bộ nhớ ngoài và dễ truy nhập
  - ✓Tổ chức tuần tự theo byte: dữ liệu được tổ chức lưu trữ, đọc và ghi một cách tuần tự từng byte. Cách tổ chức này có tính vạn năng, mọi ứng dụng đều có thể sử dụng tệp.
  - ✓Tổ chức tuần tự theo bản ghi: dữ liệu được tổ chức lưu trữ, đọc và ghi một cách tuần tự từng bản ghi với kích thước cố định.
  - ✓Tổ chức cây các bản ghi: dữ liệu được tổ chức lưu trữ, đọc và ghi theo cây các bản ghi theo trường khoá.
- Mọi thao tác phức tạp phải "trong suốt " với người dùng đảm bảo công cụ truy nhập tới tay người dùng ở dạng đơn giản nhất

***Như vậy:***

- Tồn tại các câu lệnh: đọc, ghi, tạo, đổi tên, đóng, mở file...
- Tổ chức thông tin trên phương tiện mang tin và tự động ghi nhận sơ đồ
- Bố trí file để đáp ứng yêu cầu truy nhập và tìm kiếm
  - ✓ Cấu trúc lưu trữ tuần tự, tồn tại bản ghi đặc biệt lưu trữ các tham số file
  - ✓ Tồn tại cơ chế thư mục, bộ phận hoá tên file ở phạm vi nhất định, các Thông tin liên hệ với nhau bằng danh sách móc nối
- Có cơ chế bảo vệ file:
  - ✓ Tĩnh: liên quan tới toàn bộ file và cố định theo thời gian
  - ✓ Động: xác lập khi mở file đọc, ghi thông tin
- Xoá dữ liệu trong file:
  - ✓ Mức vật lý: toàn bộ nội dung file
  - ✓ Mức logic: ngắt các móc nối liên hệ với file

***Phân lớp:***




Ở mức người dùng:

- Giao diện tốt
- Mang tính đặc thù của hệ thống

Mức hệ điều hành:

- Mang tính vạn năng
- Tồn tại nhiều thành phần, phụ thuộc vào thiết bị vì nó phải liên hệ với hệ thống

## 2.5. Quản lý file trong hệ điều hành MSDOS

Bộ nhớ ngoài (đĩa từ) có hai tham số chính:

- Tham số về thiết bị đọc đĩa từ
- Tham số về bản thân đĩa

### *2.5.1. Thiết bị đọc, ghi:*

Nguyên tắc hoạt động theo nam châm điện

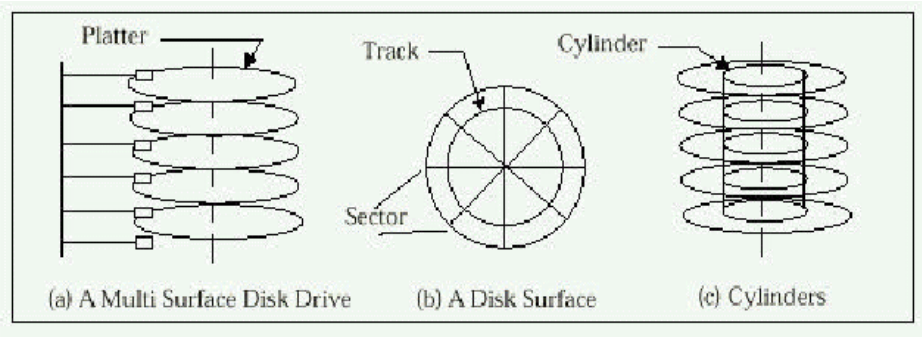
### *2.5.2. Tham số đĩa từ:*

Lưu trữ cố định trên đĩa.

Đĩa có thể có 1, 2 hay nhiều mặt (side), chúng được đánh số thứ tự từ 0. Mỗi mặt được truy nhập bằng một đầu từ (head).

Trên các mặt, thông tin được ghi theo các đường tròn đồng tâm (Track - rãnh từ) có địa điểm đầu thẳng hàng nhau. Chúng được đánh số thứ tự từ 0 và từ ngoài vào trong tâm đĩa.

Tập hợp các rãnh có bán kính bằng nhau trên các mặt tạo thành từ trụ (Cylinder)




Trên các rãnh, thông tin ghi theo từng phần một gọi là cung từ (sector) có độ dài bằng nhau, một sector có thể là 128, 256, 512, 1024… byte. Các sector được đánh số bắt đầu từ 1 (Hiện nay để nâng cao dung lượng lưu trữ trên đĩa từ, kỹ thuật LBA được sử dụng…)

Sector 1 không nằm cạnh sector 2 mà cách một khoảng nào đó gọi là hệ số đan xen (interleave). Interleave là số nguyên tố cùng nhau với số sector trên track

***Ví dụ:*** Đĩa mềm: Interleave=7

Đĩa cứng hệ số này từ 3 đến 4

Địa chỉ vật lý của 1 sector được xác định bởi:

- ✓ Số hiệu của Side/Head
- ✓ Số hiệu của Track/Cylinder
- ✓ Số hiệu của Sector

Trong thực tế còn sử dụng khái niệm liên cung (Cluster): Là số các sector liên tiếp nhau về mặt logic và là đơn vị phân phối bộ nhớ cho người dùng (1 cluster có thể là 2, 4, 8, 16, 32... sector). Địa chỉ logic

Địa chỉ logic của 1 sector còn được xác định bởi:

- ✓ Số hiệu của Cluster (tính từ Cylinder 0, Head 1)
- ✓ Số Sector/1 Cylinder
- ✓ Số hiệu của Sector (tính từ đầu Cylinder)

relSec = (CylNo * SecsPerTrack * Heads) + (HeadNo * SecsPerTrack) + (SecNo - 1)

***Đọc/ghi thông tin trên 1 sector của đĩa***

Sử dụng ngắt 13H của BIOS để đọc/ghi đĩa, với kiểu dữ liệu thanh ghi (Registers)

Giá trị các thanh ghi:

AH: 01h: Ghi Sector; 02h: Đọc Sector

AL: Số Sector cần đọc/ghi

CH: Số hiệu Track/Cylinder

CL: Số hiệu Sector

DH: Số hiệu đầu từ

DL: Số hiệu đĩa (F0h = A...; 80H = HD0; 81H = HD1)

ES:BX => địa chỉ vùng nhớ

Chú ý: Giá trị Sector gồm 6 bit và Cylinder là 10bit:

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cylinder | | | | | | | | | | Sector | | | | | |

Thủ tục mã hoá Cylinder và Sector

```
Function CylSecEncode(Cylinder, Sector : Word) : Word;
Begin
    CylSecEncode := (Lo(Cylinder) shl 8) or (Hi(Cylinder) shl 6) or Sector;
End;
```

Output:

Nếu có lỗi: Carry Flag=CY=1 và mã lỗi trong AH

Nếu không lỗi: AH = 0 và ES:BX => địa chỉ vùng nhớ

***Một đĩa cứng bao gồm:***

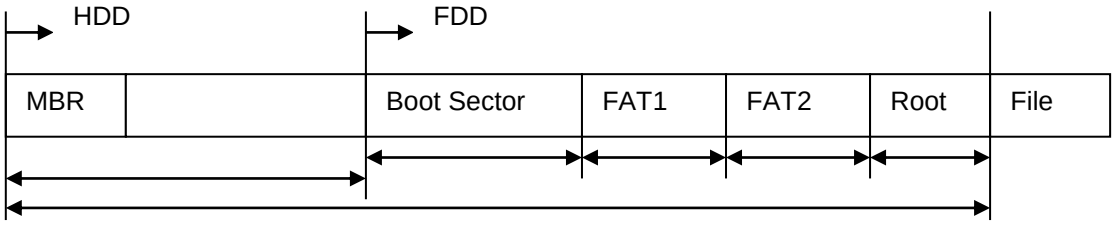
- Phần hệ thống
- Phần dữ liệu

Phần hệ thống bao gồm:

- Master boot record
- Boot sector

- FAT
- ROOT

Hình ảnh cấu trúc:




Vùng hệ thống

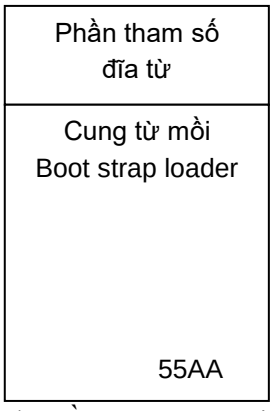
- MBR: Master Boot Boot chính của đĩa từ cứng
- MBR trỏ tới Boot Sector và những Boot Sector còn lại (nếu có)
- Boot Sector trỏ tới Root và từ Root truy nhập vào FAT1, từ FAT1 truy nhập tới File

Boot sector: luôn tồn tại ở mỗi đĩa từ

Nó bao gồm hai phần:

- Xác định tham số tổ chức của đĩa: đặc thù cho mỗi đĩa
- Chương trình mồi phục vụ cho việc nạp hệ điều hành: vùng này là bắt buộc với đĩa hệ thống, với đĩa ghi dữ liệu tì có thể bỏ trống

Nạp hệ thống: Thực chất là đọc Boot Sector và ghi vào vùng địa chỉ 7C00h




Để truy nhập thông tin trên đĩa cần quan tâm tới các thông số:

- Số byte cho một sector
- Số sector trước FAT
- Số bảng FAT
- Số mục vào (entry) cho root (32 byte cho một entry)
- Tổng số sector trên đĩa
- Số lượng sector cho một bảng FAT
- Số sector trên một track
- Số đầu đọc, ghi

Truy nhập Boot Sector

- Xác định vị trí của nó trên đĩa
- Đọc trực tiếp sector thông qua ngắt 13h hoặc 25h

Vị trí Boot Sector:

- Đĩa mềm: sec1, đầu đọc 0, cylinder 0
- Đĩa cứng: sec1, đầu đọc 1, cylinder 0

*A/ Bảng tham số:*

| Số hiệu | Địa chỉ offset | Chiều dài (byte) | Ý nghĩa |
|---|---|---|---|
| 1 | 0 | 3 | EBxx90 (số hiệu đặc biệt) |
| 2 | 3 | 8 | Tên hệ thống format đĩa |
| 3 | B | 2 | Số byte/sector (byte thấp được lưu trữ trước 1234→ 34\|12) |
| 4 | D | 1 | Sec/clus kích thước trong bảng phân phối cho người dùng |
| 5 | E | 2 | K/c từ đầu logic đĩa từ tới bảng FAT1 |
| 6 | 10 | 1 | Số bảng FAT |
| 7 | 11 | 2 | Số phần tử thư mục gốc root |
| 8 | 13 | 2 | Số sec trên đĩa nếu dung lượng đĩa nhỏ hơn 32MB |
| 9 | 15 | 1 | Loại đĩa: F8: HD; F9: FD(1.2M); F10: FD (1.44M) |
| 10 | 16 | 2 | Số sec/FAT |
| 11 | 18 | 2 | Số sec/track |
| 12 | 1A | 2 | Số đầu từ |
| 13 | 1C | 4 | Đ/c tuyệt đối boot sector |
| 14 | 20 | 4 | Số sec trên đĩa nếu dung lượng đĩa lớn hơn 32MB |
| 15 | 24 | 1 | Đ/c vật lý đĩa từ: 80: C, 81: D, 00: FD |
| 16 | 25 | 1 | Dự trữ |
| 17 | 26 | 1 | Dấu hiệu 29h |
| 18 | 27 | 4 | Serial number |
| 19 | 2B | 11 | Volume name |
| 20 | 36 | 8 | FAT |
| Còn lại 482 byte chứa Chương trình mồi | | | |

Ví dụ:

```
EB BC 90     4D 53 44 4F 53 35 2E 30    00 02    20
01 00     02     00 02     00 00     F8     80 00     3D 00
12 00     BC 19 00 00     0E 00 10 00     50     00     29
D2 15 BE 18     4E 4F 20 4E 41 4D 45 20 20 20 20
46 41 54 31 36 20 20 20 F1 33
```

Đọc MBR

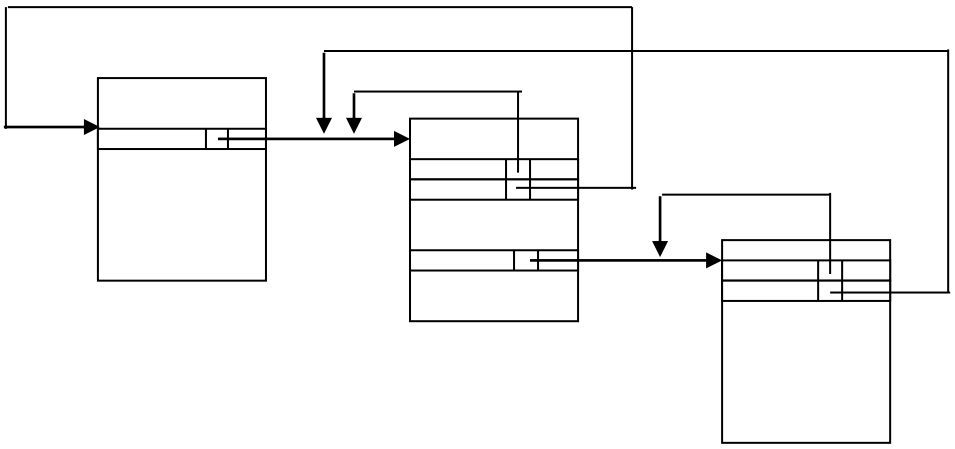
```
uses crt,MSDOS;
 const s16:string[16]='0123456789abcdef';
 var reg:registers;
 B:array[0..511]of byte;
 i:integer; j,k:byte;ch:char;
 begin clrscr;
 with reg do
 begin
 dl:=$80;dh:=0;
 cl:=1;ch:=0;
 al:=1;ah:=2;
 bx:=ofs (b);es:=seg (b);
 end;
 intr ($13,reg);
 for i:=$1be to 511 do
 begin
 j:=b[i]shr 4+1;
 k:=b[i]and$0f+1;
```

```
write (s16[j]:2,s16[k]);
if (i+1)mod 16 =0 then
begin
write (' ':5);
for j:=i-15 to i do
if (b[j]<32)or (b[j]=255)then write ('.')
else write (chr (b[j]));
if (i=255)then ch:=readkey;
writeln;
end end;
readln
end.
```

*B/ Thư mục gốc (Root Directory)*

Dãy các mục vào, mỗi mục vào 32 byte chia thành:

| Số hiệu | Địa chỉ | Độ dài | ý nghĩa |
|---|---|---|---|
| 1 | 0 | 8 | Tên t/m, tệp (nếu thiếu bổ sung dấu cách 20h) |
| 2 | 8 | 3 | Phần mở rộng |
| 3 | B | 1 | Thuộc tính |
| 4 | C | 10 | Chưa dùng tới (với MSDOS6.22) |
| 5 | 16 | 2 | Giờ tạo lập |
| 6 | 18 | 2 | Ngày tạo lập |
| 7 | 1A | 2 | Chứa liên cung khởi động |
| 8 | 1C | 4 | Kích thước tệp |



***Byte số 0 trong tên thư mục, tệp:***

- Nếu là 00h thì phần tử này chưa sử dụng bao giờ
- Nếu là E5h thì phần tử này đã được sử dụng nhưng bị xoá
- Nếu là 2E 20h (.) phần tử đầu tiên của thư mục con. Liên cung khởi động (Starting cluster) của phần tử 1 chỉ chính nó
- Nếu là 2E 2Eh (..) phần tử thứ hai của thư mục con. Liên cung khởi động (Starting cluster) của phần tử 2 chỉ thư mục mẹ
- Liên cung khởi động (Starting cluster) của thư mục gốc với số hiệu: 00h

***Byte thuộc tính:***

| | A | D | V | S | H | R |
|---|---|---|---|---|---|---|
| Trọng số: | 32 | 16 | 8 | 4 | 2 | 1 |

A: thuộc tính lưu trữ

D: thư mục

V: Nhãn đĩa

S: hệ thống: các chương trình có đặc quyền hệ thống mới có thể truy nhập

H: ẩn đánh dấu một tệp bị che

R: chỉ đọc

Ví dụ:

Tệp IO.SYS với các thuộc tính: A S H R

Trọng số : 32 4 2 1 Giá trị: $(39)_{10}$:27h

Tệp COMMAND.COM với các thuộc tính: A

Trọng số : 32 Giá trị: $(32)_{10}$:20h

Thư mục TP : D

Trọng số : 16 Giá trị: $(16)_{10}$: 10h

Kiểm tra thuộc tính một tệp:

Đọc giá trị trong byte Attribute (At)

Thực hiện phép AND tương ứng với trọng số của các thuộc tính đó

Ví dụ:

Thuộc tính H: At AND 2<>0

Thư mục con: At AND 16<>0

Gán thuộc tính một tệp:

Thực hiện phép OR tương ứng với trọng số của các thuộc tính đó

Ví dụ:

Thuộc tính H: At OR 2

Xoá thuộc tính một tệp:

Thực hiện phép AND tương ứng mã bù1 tương ứng với thuộc tính đó

Ví dụ:

Thuộc tính R (thực hiện phép AND với 1111 1110): At AND FEh

Thuộc tính H (thực hiện phép AND với 1111 1101): At AND FDh

***Ngày, giờ tạo lập hệ thống:***

Byte Time: xxxxx xxxxxx xxxxx

giờ phút giây

Byte Date: xxxxxxx xxxx xxxxx

số năm tháng ngày (Giá trị năm tính từ năm 1980)

*C/ Bảng FAT*

Chức năng:

- Tạo danh sách móc nối các Cluster của cùng một tệp (quản lý bộ nhớ đã sử dụng)
- Quản lý bộ nhớ tự do (vùng bộ nhớ chưa dành cho tệp tin hay thư mục nào)
- Đánh dấu các Bad Cluster (nâng cao độ tin cậy đĩa)

Bao gồm:

- Dãy các phần tử, mỗi phần tử có thể là: 12,16, 32bit tương ứng cho FAT12, FAT16, FAT32.
- Các phần tử được đánh số 0,1,2..
- Từ phần tử thứ 2, mỗi phần tử trong FAT tương ứng với một Cluster và ngược lại

FAT12: Dung lượng: $2^{12}$= 4096KB = 4MB

FAT16: Dung lượng: $2^{16}$= 64MB với 2Sector/1Cluster

$2^{16}$= 128MB với 4Sector/1Cluster

$2^{16}$= 1024MB với 32Sector/1Cluster

FAT32: Dung lượng: $2^{32}$= 8GB với 2Sector/1Cluster

- Phần tử thứ nhất: tất cả các bit là 1 do đó

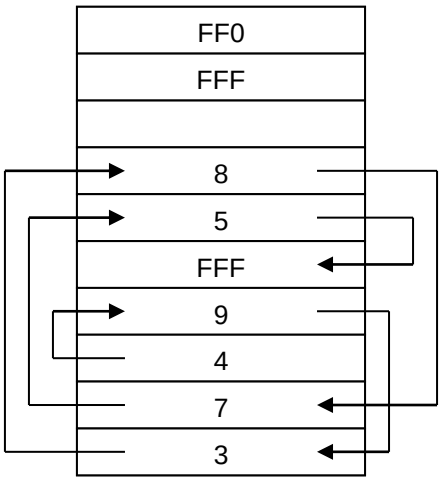
  Với đĩa cứng: HD: số hiệu FF8h

  Với đĩa mềm: FD: số hiệu FF0h

- Dấu hiệu kết thúc 1 chuỗi Cluster là FFFh hoặc FFFFh

Ví dụ: Đĩa mềm 1.44MB với FAT12

Starting Cluster: 6:




Để tăng tốc độ truy nhập, ngay lần truy nhập đầu tiên làm việc với đĩa từ, hệ thống sẽ đọc luôn FAT và ROOT vào RAM, như vậy hệ thống chỉ cần truy nhập vào bộ nhớ để lấy thông tin, không cần phải truy nhập lại đĩa từ do vậy tăng được tốc độ và giảm được di chuyển cơ khí của đầu từ.

Đọc FAT

```
uses crt,MSDOS;
 const S16:string[16]='0123456789abcdef';
 var B:array[0..511]of byte;
 i,j:integer;start:word;
 drv,cyl,head,sec,numsec,drive:byte;
 Function R_sector (drive,cyl,head,sec,numsec:byte):integer;
 var reg:registers;
 begin
 with reg do
 begin
 dl:=drive;dh:=head;ch:=cyl;
 cl:=sec;al:=numsec;ah:=2;
 es:=seg (b);bx:=ofs (b);
 end;
 intr ($13,reg);
 end;
 Function R_fat (var start:word):word;
 var k,k1,k2,k3,tg,l:integer;ch:char;
 begin
 for i:=0 to 511 do b[i]:=0;
 if (drv=0)or (drv=1)then
 begin
 drive:=0;head:=0;
 end else
 begin
 drive:=$80;head:=1;
 end;
 i:=start;
 if (drv=0)or (drv=1)then
 begin
 j:= (i*3)div 2;k:=j div 512;
```

```
tg:=R_sector (drive,0,head,2+k,1);
j:=j mod 512;
l:=memw[seg (b[j]):ofs (b[j])];
if odd (i) then l:=l shr 4
else l:=l and $0fff;
end
else
begin
j:=i*2;k:=j div 512;
tg:=R_sector (drive,0,head,2+k,1);
j:=j mod 512;
l:=memw[seg (b[j]):ofs (b[j])];
l:=l and $0fff;
end;
k1:=l shr 8+1;k2:= (l shr 4)and $0f+1;k3:=l and $0f+1;
write (s16[k1],s16[k2],s16[k3],' ');
if (s16[k1]+s16[k2]+s16[k3]<>'fff')then R_fat:=l else
begin
writeln;writeln ('End of file press any case:');
ch:=readkey;
if (ch='q')then halt (1);
end
end;
Begin clrscr;
write ('Ten odia:');readln (drv);
for start:=7 to 150 do i:=r_fat (start);
End.
```

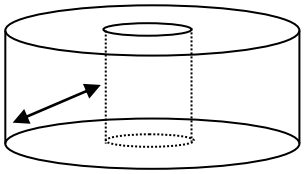
*D/ Partition*

Bao gồm: 4 phần tử, mỗi phần tử 16byte chia thành 4 trường, mỗi trường 4byte.

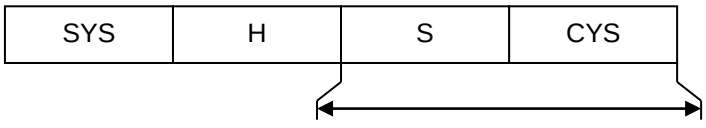
Mỗi phần tử khác không xác định 1 phần tử được sử dụng như 1 đĩa từ độc lập

Nếu biết địa chỉ vật lý đầu có thể tính được địa chỉ logic đầu, các tham số còn lại

| Đ/c V/lý đầu | Đ/c V/lý cuối | Đ/c Logic cuối | Tổng số Sector |
|---|---|---|---|



*Địa chỉ vật lý đầu:* 4byte




SYS: byte hệ thống

Bằng 00h nếu đĩa là đĩa làm việc

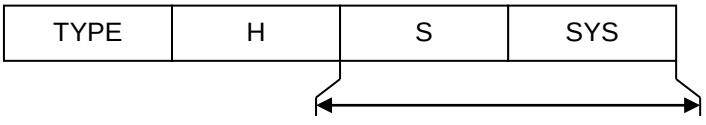
Bằng 80h nếu là đĩa hệ thống (phần chứa hệ thống được đặt tên là ổ đĩa C)

H: chứa số đầu đọc

S: chứa số sector

CYL: số cylinder

*Địa chỉ vật lý cuối:* 4byte



TYPE:

00h: cấm đọc ghi (không cho phép truy nhập)

01h: áp dụng cho đĩa có dung lượng nhỏ hơn 4MB (FAT 12)

04h: áp dụng cho đĩa có dung lượng nhỏ hơn 32MB (FAT 16)

06h: áp dụng cho đĩa có dung lượng lớn hơn 32MB (FAT 16)

0Ch: áp dụng cho đĩa có dung lượng lớn hơn FAT32

SYS:

Bằng 80h nếu là đĩa hệ thống TYPE = 51 (DM: disk manager)

Bằng 05h: loại mở rộng (extended) cấu trúc logic như một đĩa cứng vật lý vì vậy tồn tại master boot riêng, có partition riêng.

Ví dụ:

Với FAT16: địa chỉ bắt đầu của Partition là 1BEh

80 01 01 00 06 3F FF 4D 3F 00 00 00 41 00 34 00

00 00 00 00...

00 00

00 00

55 AA

## CÂU HỎI VÀ BÀI TẬP

2.1. Trình bày việc phân cấp trong tổ chức và quản lý thiết bị ngoại vi.

2.2. Trình bày vai trò của bộ đệm? Nêu các cách điều khiển bộ đệm vào ra dữ liệu.

2.3. Trình bày các phương pháp tổ chức dữ liệu trên hệ thống máy tính

2.4. Anh, chị hãy trình bày các phương pháp truy nhập dữ liệu đã được dùng phổ biến hiện nay.

2.5. Vẽ sơ đồ thuật toán việc đọc và hiển thị giá trị của 05 chỉ mục đầu tiên trong bảng FAT32

2.6. Xây dựng chương trình đọc thông tin trên 1 đĩa cứng và hiển thị ra màn hình cho biết đĩa đó có bao nhiêu mặt, trên mỗi mặt có bao nhiêu rãnh, trên mỗi rãnh có bao nhiêu sector, số sector trên một liên cung và tổng số liên cung.

2.7. Xây dựng chương trình liệt kê và lưu vào đĩa bảng phân vùng của đĩa cứng cụ thể

2.8. Xây dựng chương trình liệt kê và lưu các thông số hệ thống được lưu trữ trong Boot Record của 01 đĩa logic.

2.9. Xây dựng chương trình liệt kê các mục vào của Root trên 01 đĩa cứng cụ thể FAT16 hoặc FAT32)

2.10. Xây dựng chương trình liệt kê 01 sector bất kỳ trên đĩa.

2.11. Xây dựng chương trình liệt kê thông tin của 01 file text đã lưu trữ trên đĩa

2.12. Xây dựng chương trình liệt kê thông tin của bảng MFT trong đĩa cứng sử dụng kỹ thuật quản lý file NTFS

# Chương III: QUẢN LÝ BỘ NHỚ

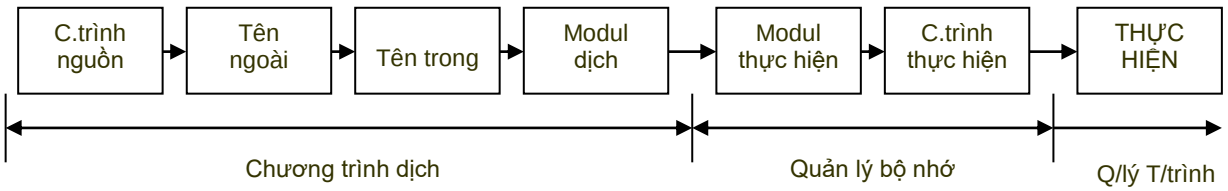
## 3.1. Đặt vấn đề

Bộ nhớ là 1 tài nguyên không thể thiếu được, đóng vai trò lưu trữ thông tin để xử lý vì vậy nó có liên quan tới tốc độ xử lý.

Một phần bộ nhớ trong dùng lưu trữ nhân (kernel) của hệ thống- tập các chương trình điều khiển thường xuyên có mặt ở bộ nhớ trong để thực hiện khi cần.

Chức năng khác của hệ điều hành là bảo vệ chương trình và dữ liệu khỏi bị hư hỏng, truy nhập một cách không hợp thức khi các chương trình khác hoạt động.

Các bước xử lý chương trình:




Chương trình nguồn: Các chương trình được viết dưới dạng ngôn ngữ thuật toán qua chương trình dịch dịch sang ngôn ngữ máy.

Các phép ánh xạ: hệ thống phải chuyển đổi các tên ngoài thành tên trong

Tên ngoài: do người dùng đặt

Tên trong: tên do hệ thống đặt trong quá trình dịch, dùng phân phối bộ nhớ và xác lập mối quan hệ đơn trị tên địa chỉ (do hàm địa chỉ thực hiện)

Hàm địa chỉ xác lập quan hệ giữa không gian tên và không gian bộ nhớ (bộ nhớ logic)

Modul dịch của chương trình là chương trình viết trên ngôn ngữ máy, nhưng nó mới được xét độc lập không nhúng vào quan hệ chung của toàn hệ thống vì vậy cần phải chuyển thành chương trình thực hiện.

Việc tạp hợp các chương trình modul dịch thành các chương trình dạng thực hiện do chương trình biên tập (LINK) đảm nhiệm.

Định vị chương trình: nạp chương trình vào bộ nhớ trong cụ thể, đặt vào vị trí xác định và sửa địa chỉ cho thích hợp với môi trường khai thác cụ thể.

Với chương trình .COM: dung lượng nhỏ hơn 64KB nên có thể đặt ở một nơi nào đó và thực hiện ngay không cần sửa đổi

Với chương trình .EXE: chương trình được chuẩn bị gần dạng thực hiện nhưng chưa lắp ráp vì vậy khi đưa vào bộ nhớ phải láp ráp theo chương trình điều khiển (biên tập lại)

Chế độ lập trình:

- ✓ $V_{LG}$: dung lượng bộ nhớ logic
- ✓ $V_{PH}$: dung lượng bộ nhớ vật lý

Chế độ bộ nhớ thực:

Yêu cầu $V_{LG} < V_{PH}$ như vậy bộ nhớ sử dụng nhỏ hơn bộ nhớ ta có

Chế độ bộ nhớ ảo:

Không có ràng buộc giữa $V_{LG}$ và $V_{PH}$ như vậy khi quản lý không gian bộ nhớ:

- ✓ Logic: quan tâm tới chương trình được bố trí như thế nào\
- ✓ Vật lý: quan tâm tới chương trình và dữ liệu

Tóm lại: quan tâm tới việc xác lập quan hệ giữa hai bộ nhớ

### 3.2. Quản lý bộ nhớ logic - cấu trúc một chương trình

Một chương trình có thể bao gồm nhiều modul, các modul có thể có cùng một dạng cấu trúc hoặc có những cấu trúc khác nhau

Chương trình có thể có các dạng cấu trúc:

- Tuyến tính
- Động
- Overlay
- Phân đoạn
- Phân trang

#### *3.2.1. Cấu trúc tuyến tính*

Sau khi biên tập các modul được tập hợp thành 1 chương trình hoàn thiện chứa đầy đủ thông tin để có thể thực hiện

Thực hiện: định vị 1 lần vào bộ nhớ

Ưu điểm:

- Đơn giản, chỉ việc tìm các móc nối
- Không có sự gò bó về thời gian
- Tính lưu động cao: có thể chuyển từ nơi này tới nơi khác

Nhược:

- Lãng phí bộ nhớ vì phải sử dụng vùng bộ nhớ lớn hơn mức cần thiết

#### *3.2.2. Cấu trúc động*

Từng modul được biên tập riêng biệt

Khi thực hiện chỉ việc nạp modul đầu tiên vào bộ nhớ

Khi cần modul khác người sử dụng phải sử dụng lệnh macro hệ thống để nạp định vị modul hoặc xoá modul ra khỏi bộ nhớ

Ví dụ: Lệnh Macro

Attach: nạp, gắn vào

Load: nạp modul vào nhưng chưa thực hiện

Delete: xoá modul khỏi bộ nhớ

Người dùng có thể tham gia trực tiếp voà quá trình định vị

Ưu điểm:

- Tiết kiệm bộ nhớ

Nhược:

- Yêu cầu người dùng phải biết kích thước hệ thống
- Thời gian thực hiện lớn, vừa thực hiện vừa định vị
- Kém linh động

#### *3.2.3. Cấu trúc Overlay*

Các modul chương trình được chia thành từng lớp

- Lớp 0: modul gốc- modul đầu tiên được gọi
- Lớp 1: modul được modul lớp 0 gọi (không cần được gọi đồng thời)
- Lớp 2: modul được modul lớp 1 gọi
- …

Bộ nhớ dành cho chương trình được chia thành các phần, mức bộ nhớ và mức chương trình

Để biết modul nào thuộc mức nào người dùng phải cung cấp thông tin cho biết:

- Số mức, modul tương ứng với mức (gọi là sơ đồ overlay hay file Overlay - OVL)
- Modul mức 0 được để ở 1 file chương trình riêng, khi cần nạp modul nào thì hệ thống tìm kiếm trong overlay và nạp vào bộ nhớ ở mức overlay tương ứng
- Duy trì hoạt động chương trình theo sơ đồ overlay gọi là supervisor overlay

Khi nạp vào mức đã dùng rồi thì modul cũ bị xoá

Ưu điểm:

- Tiết kiệm bộ nhớ
- người dùng không phải can thiệp vào chương trình nguồn
- Các modul không phải lưu trữ nhiều lần

Nhược:

- Người dùng phải cung cấp sơ đồ overlay
- Hiệu quả sử dụng bộ nhớ tăng dần tới 1 mức nào đó thì dừng lại
- Hạn chế 1 số cách gọi chương trình con

### *3.2.4. Cấu trúc phân đoạn*

Khi chương trình của người dùng được biên tập tạo thành các modul riêng biệt, tập hợp các chương trình là 1 bảng điều khiển cho biết chương trình có thể sử dụng những modul nào thông qua SCB (segment control block)

SCB chứa 1 số thông tin trợ giúp định vị chương trình, dựa vào SCB nạp modul vào trong bộ nhớ.

Khi thực hiện chương trình dựa vào SCB kiểm tra xem modul có trong bộ nhớ hay không, nếu chưa có trong bộ nhớ thì chương trình được nạp vào bất kỳ vùng nhớ nào .

Ưu điểm:

- Các modul không cần phải naptucj và không cần có vị trí cố định
- Người dùng không cần phải khai báo bất kỳ thông tin phụ nào
- Thực hiện nhanh hơn so với sơ đồ overlay
- Hiệu quả tăng dần theo kích thước bộ nhớ

Nhược:

- Phụ thuộc cấu trúc ban đầu của chương trình nguồn

### *3.2.5. Cấu trúc phân trang*

Chương trình của người dùng được chia thành từng trang có kích thước giống nhau được quản lý bởi bảng quản lý trang

Khi thực hiện sẽ nạp dần từng trang theo nhu cầu vì vậy hạn chế lãng phí bộ nhớ

Có sự hỗ trợ của phần cứng

Đẩy hệ số tích trữ bộ nhớ lên cao

Phân cấp bộ nhớ:

***Phân Trang:***

ROM (384B):

- Ghi đọc 1 lần
- Không cần nguồn để giữ
- Tốc độ truy nhập cao

Sơ cấp: 640KB đầu tiên

Expended:

- Phục vụ cho Vào/ra
- Processor cần làm việc trực tiếp với bộ nhớ này
- Extended: đòi hỏi chế độ mở rộng

***Ngoài:***

Disk: khối lượng lớn, thời gian lưu trữ lâu dài

***Chuyên dụng:***

- ✓ CMOS (64KB): lưu trữ thông tin cấu hình
- ✓ R: truy nhập nhanh, phải phối hợp với CPU
- ✓ Buffer: hoạt động như 1 máy tính chuyên dụng
- ✓ Cache: phục vụ Vào/ra

### 3.3. Quản lý bộ nhớ vật lý

Bộ nhớ có lích thước cụ thể vì vậy nhạy cảm với các kiểu sử dụng cụ thể

#### *3.3.1. Phân chương cố định*

Bộ nhớ được chia thành n phần không nhất thiết phải bằng nhau, mỗi phần sử dụng như 1 bộ nhớ độc lập gọi là ***Chương***

Bao nhiêu Chương thì có bấy nhiêu chương trình

Mỗi chương trình có 1 danh sách quản lý bộ nhớ tự do chưa sử dụng riêng

Chương trình được nạp vào chương nào sẽ tồn tại ở đó cho tới khi kết thúc

Ưu điểm:

- Đơn giản
- Dễ bảo vệ
- Tồn tại công cụ bên trong bộ nhớ có thể phân chia lại hệ thống
- Có thể phân loại các chương trình trước khi thực hiện vì vậy có thể tổ chức phục vụ gần tối ưu

Nhược:

- Bộ nhớ bị phân đoạn nên khi phân chia lại sẽ thay đổi đường biên vì vậy thông tin bị xoá

#### *3.3.2. Chế độ phân chương động*

Chỉ tồn tại 1 danh sách quản lý bộ nhớ tự do cho toàn bộ hệ thống nhớ.

Mỗi chương trình khi xuất hiện được phân phối 1 vùng nhớ riêng liên tục được sử dụng như 1 bộ nhớ độc lập

Ưu điểm:

- Hệ số song song cao, không cố định
- Số chương trình thực hiện có thể thay đổi
- Không bị phân đoạn nên có thể thực hiện 1 chương trình bất kỳ miễn là có đủ bộ nhớ
- Cơ chế đợi chờ (đủ bộ nhớ thì làm việc)
- Hệ thống điều khiển không bị sao chép đi nơi khác

Nhược:

- Hiệu quả sử dụng bộ nhớ không cao
- Nếu có sự cố kỹ thuật thì chương trình sẽ bị phá huỷ
- Sơ đồ phức tạp
- Xuất hiện hiện tượng phân đoạn ngoài

Khắc phục:

- Bố trí lại bộ nhớ tìm thời điểm thích hợp làn lượt dừng các chương trình đang được thực hiện
- Đưa 1 số chương trình từ vùng nhớ trang sang nhớ ngoài

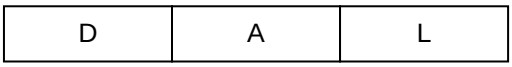
#### *3.3.3. Chế độ phân đoạn*

Chương trình có cấu trúc phân đoạn. (Được biên tập thành các modul riêng biệt → Tồn tạo bảng SCB)

Người dùng hoàn toàn không quan tâm tới SCB và chương trình của họ được bố trí như thế nào trong bộ nhớ

SCB bao gồm các phần tử, mỗi phần tử tương ứng với 1 modul độc lập.

Mỗi phần tử bao gồm 3 trường:




Trường D:

0: chưa nạp vào bộ nhớ

1: đã nạp

Trường A: địa chỉ nơi nạp modul vào bộ nhớ

Trường L: độ dài modul

Ban đầu D và L có giá trị, L chỉ kích thước modul và D=0 (chưa nạp), SCB được xây dựng ngay khi biên tập

Khi thực hiện SCB được nạp vào trong bộ nhớ, địa chỉ của nó được đưa vào thanh ghi quản lý đoạn $R_S$ (register segment)

Địa chỉ truy nhập dữ liệu được biểu diễn dưới dạng cặp (s,d)

s: số hiệu segment (modul) cần truy nhập

d: địa chỉ tương đối tính từ segment

Truy nhập: 2 lần hướng tới bộ nhớ

- Lần 1: Lấy nội dung của thanh ghi ($R_S$) ghép với s để truy nhập tới phần tử thứ s trong bảng SCB
- Lần 2: dựa vào đó (khi D=1) lấy a+d để truy nhập tới dữ liệu

Ưu điểm:

- Áp dụng trên máy bất kỳ
- Cho phép sử dụng chung các modul trong bộ nhớ

Nhược:

- Hiệu quả phụ thuộc cấu trúc ban đầu của chương trình nguồn
- Phân đoạn ngoài: bố trí lại bộ nhớ

Nếu xuất hiện nhu cầu bố trí lại:

- Đưa ra modul tồn tại duy nhất trong bộ nhớ
- Đưa ra modul có lần sử dụng cách đay lâu nhất
- Đưa ra modul có tần suất sử dụng thấp nhất

### *3.3.4. Chế độ phân trang (ánh xạ bộ nhớ logic thành vật lý)*

Bộ nhớ vật lý được chia thành từng phần bằng nhau gọi là ***Trang***, các trang được đánh số thứ tự 0,1,2..

Chương trình phải có cấu trúc trang

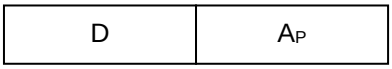
Trang trong chương trình phải có cùng kích thước trang vật lý

1 trang vật lý: 256byte-4KB

Khi làm việc chương trình được biên tập theo từng trang tạo ra các PCB

PCB: là tập hợp các phần tử mỗi phần tử ứng với 1 trang của chương trình

Bao gồm 2 trường:




Trường D: dấu hiệu cho biết trang được nạp vào bộ nhớ hay chưa

0: chưa

1: đã nạp

Trường $A_P$: địa chỉ trang

Khi thực hiện :

PCB được nạp vào bộ nhớ

Địa chỉ đầu được đưa vào thanh ghi $R_P$

Địa chỉ dữ liệu được biểu diễn dưới dạng: (p,d)

p: số hiệu trang

d: offset tính từ đầu trang

Truy nhập dữ liệu: 2 hướng tới bộ nhớ

- Lần 1: lấy $R_P$+p để truy nhập tới trang p trong PCB
- Lần 2: đợi d=1 lấy Ap ghép với d truy nhập dữ liệu

Ưu:

- Không có hiện tượng phân đoạn ngoài
- Hạn chế việc thiếu bộ nhớ
- Khi thiếu bộ nhớ có thể giải phóng bằng cách đưa 1 trang ra ngoài
    - Trang tồn tại lâu nhất trong bộ nhớ
    - Trang có số lần sử dụng cách đay lâu nhất
    - Trang có tần suất sử dụng thấp nhất

Nhược:

- Bảng PCB có thể có kích thước lớn

### *3.3.5. Chế độ kết hợp phân trang và phân đoạn*

Bộ nhớ được tổ chức theo kiểu phân trang

Chương trình được tổ chức theo kiểu phân đoạn (Tồn tại SCB)

Mỗi modul được biên tập theo chế độ phân trang vì vậy mỗi modul có 1 PSB riêng

Mỗi phần tử của SCB sẽ quản lý các PCB tương ứng của modul

D: xác định PCB vào bộ nhớ hay chưa

A: địa chỉ đầu PCB

L: độ dài Modul

Khi thực hiện một chương trình SCB được nạp vào trong bộ nhớ. Địa chỉ đầu của nó được đưa vào thanh ghi Rs

Bộ nhớ được chia thành 3 phần:

Phần1: chứa SCB

Phần2: chứa các PCB

Phần3: chứa các trang chương trình và dữ liệu

Truy nhập:

Địa chỉ biểu diễn: (s,p,d)

Trong đó:

s: modul cần truy nhập

p: trang cần truy nhập

d: địa chỉ offset tính từ đầu trang

Truy xuất ô nhớ: Mỗi lần truy nhập cần 3 lần hướng tới bộ nhớ

- Lần1: lấy nội dung $R_S$+s: truy nhập phần tử s của SCB

- Lần2: d=1: lấy A+p truy nhập phần tử thứ p của PCB thứ s
- Lần3: Dp=1: lấy $A_P$ ghép với d truy nhập dữ liệu

Ưu:

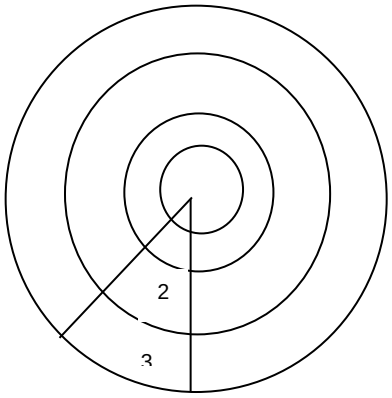
- Kết hợp ưu của phân đoạn và phân trang
- Chống hiện tượng phân đoạn ngoài
- Đảm bảo sử dụng hiệu quả bộ nhớ

### 3.4. Quản lý bộ nhớ IBM PC của MSDOS

Hệ thống MSDOS được chia thành 4 mức 0,1,2,3

- Mức 0: Nhân hệ điều hành (Kennel)
- Mức 1: Quản lý thiết bị, File
- Mức 2: Chương trình phục vụ hệ thống
- Mức 3: Chương trình ứng dụng

Mức ưu tiên 0-3




Một chương trình chỉ được quyền truy nhập tới chương trình và dữ liệu cùng mức ưu tiên hoặc kém mức ưu tiên hơn

Bộ nhớ phân phối cho 1 chương trình chia làm 2 loại:

Bộ nhớ chung:

- Vùng nhớ mà mọi chương trình đều được biết và được quyền truy nhập
- Có bảng tham số điều khiển GDT (Global Descreption Table)

Bộ nhớ riêng:

- Phân phối cho chương trình nào thì chỉ có chương trình đó được biết và được quyền truy nhập
- Có bảng tham số LDT (Local Descreption Table)

Với máy PC có 2 chế độ làm việc là:

- Chế độ thực (Real mode)
- Chế độ bảo vệ (protect mode)

Nguyên tắc:

Bộ nhớ được chia thành từng khối

Real mode:

- Dung lượng khối <64KB
- Các khối được đánh số 0,1,2... gọi là Index
- Khối 0-9 dành cho người dùng sấp xỉ 640KB: bộ nhớ cơ sở
- Khối A,B cho các phương pháp tổ chức truy nhập

- Khối C-F: ROM
- Khối F,E: ROMBASIC

Protect mode:

- 0-3FFF: 16K khối
- Mỗi khối tương ứng với 1 vùng bộ nhớ thực RAM
- Từ bộ nhớ khối sang bộ nhớ logic
- Khi 1 khối được nạp trong bộ nhớ thì phần tử tương ứng sẽ thuộc 1 vùng nhớ vật lý

**CÂU HỎI VÀ BÀI TẬP**

3.1. Anh chị hãy trình bày tổng quan về kỹ thuật Swapping bộ nhớ

3.2. Anh chị hãy trình bày tổng quan về các chiến lược điều khiển trang bộ nhớ.

3.2 Anh chị hãy trình bày tổng quan về kỹ thuật quản lý bộ nhớ ảo trong Windows NT

# Chương IV: QUẢN LÝ TIẾN TRÌNH

## 4.1. Quản lý tiến trình

### *4.1.1. Khái niệm*

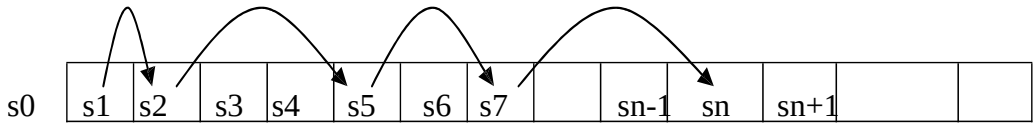
Phương pháp tiếp cận:

Coi tiến trình là nhóm các byte có nội dung thay đổi theo 1 luật nào đó, luật hướng dẫn Processor thực hiện.

- Saltzer: Tiến trình là chương trình do 1 processor logic thực hiện
- Dijkstra: Tiến trình là những gì liên quan đến hệ thống tính toán xuất hiện khi thực hiện 1 chương trình
- Định nghĩa của Horning & Randell: Tiến trình như 1 quá trình chuyển từ trạng thái này sang trạng thái khác dưới tác động của hàm hành động và xuất phát từ trạng thái ban đầu nào đó

  Hàm hành động : ánh xạ trạng thái sang hành động, hành động dựa vào trạng thái ban đầu

  Từ chuỗi các trạng thái đến công việc




- Quan điểm của người dùng: Tiến trình là một quá trình thực hiện chương trình

### *4.1.2. Tổ chức tiến trình*

Tổ chức

Tiến trình tương đương cấu trúc thông tin cho phép xác định đơn trị tiến trình (cấu trúc thông tin này gọi là khối mô tả thông tin bao gồm):

- Biến trạng thái thông tin : Trạng thái hiện tại của tiến trình
- Vùng bộ nhớ lưu trữ giá trị của các thanh ghi tiến trình sử dụng
- Thông tin về tài nguyên tiến trình đang sử dụng hoặc có quyền sử dụng.

Hình thành tiến trình

- Khung chương trình gán cho các giá trị và tài nguyên cụ thể
- Thông tin được xây dựng khi có yêu cầu và huỷ bỏ khi công việc đã hoàn thành

Phân loại tiến trình

- TT tuần tự : một tiến trình chỉ bắt đầu sau khi tiến trình kia kết thúc
- TT song song: Thời điểm bắt đầu của tiến trình này nằm giữa thời điểm bắt đầu và kết thúc của một tiến trình khác.

Quan hệ:

*Độc lập*: 2 tiến trình không có quan hệ trực tiếp gì với nhau

Yêu cầu : bảo vệ thông tin sao cho một tiến trình không làm hỏng dữ liệu và chương trình của tiến trình khác, như vậy phải phân phối tài nguyên hợp lý

*Tiến trình trao đổi thông tin với nhau*: một tiến trình có thể gửi thông báo cho tiến trình khác, tổ chức các vùng nhớ làm hòm thư.

*Phân lớp*: Trong quá trình hoạt động của một tiến trình có thể khởi tạo một tiến trình khác hoạt động song song: (chương trình chính, chương trình con)

Cơ chế cấp phát tài nguyên:

- Phân tán: Phân phối tài nguyên cho cả chương trình chính và chương trình con
- Tập chung: Tài nguyên chỉ được phân phối cho tiến trình chính

*Tiến trình đồng mức*: Những tiến trình có một số tài nguyên sử dụng chung theo nguyên tắc lần lượt.

### *4.3.3. Điều độ tiến trình - Tài nguyên Găng*

Tài nguyên Găng: Tài nguyên phân phối cho một người phục vụ, như vậy tại một thời điểm nếu đồng thời có nhiều tiến trình muốn sử dụng tài nguyên Găng: điều độ tiến trình để không có khi nào có một tiến trình chiếm dụng tài nguyên

Đoạn chương trình có sử dụng tài nguyên Găng gọi là đoạn Găng

Ví dụ:

*TTA ghi nội dung biến Dem vào TgA (biến cục bộ)*

*TTB ghi nội dung biến Dem vào TgB*

*TTA tăng TgA*

*TTB tăng TgB*

Nếu không để ý kỹ, có thể hiểu lầm là biến Dem tăng 2 đơn vị. Song thực chất cả 2 tiến trình A và B đều tăng nội dung Dem, song nội dung này chỉ tăng 1 đơn vị. Cần phải có cách giải quyết cụ thể.

- Dem : Tài nguyên Găng
- Đoạn chương trình xử lý biến Dem : Chương trình găng : Đoạn găng.

Khắc phục đụng độ :

- Tại một thời điểm có không quá một tiến trình nằm trong đoạn Găng
- Không một tiến trình nào được phép ở lâu vô hạn trong đoạn Găng
- Không một tiến trình nào phải chờ vô hạn ngoài đoạn Găng

Công cụ điều độ tiến trình qua đoạn găng :

- Cấp thấp: nằm ngoài tiến trình được điều độ
- Cấp cao: nằm trong tiến trình

Công cụ điều độ cấp thấp *:*

- Phương pháp khoá trong
- Phương pháp kiểm tra và xác lập
- Kĩ thuật đèn báo

*a, Phương pháp khoá trong (Kiểm tra luân phiên)*

Nguyên tắc: hai hay nhiều tiến trình cùng định ghi vào một địa chỉ nào đó của bộ nhớ trong thì sơ đồ kĩ thuật chỉ cho phép một tiến trình làm việc còn tiến trình khác phải chờ

Mỗi tiến trình: sử dụng một byte trong vùng bộ nhớ chung làm khoá, khi vào được đoạn Găng, gán giá trị là 1, thông báo cho các tiến trình khác biết đã có tiến trình sử dụng tài nguyên găng

Giải thuật Delker

```
Begin
     k1 := 0; k2:= 0; tg:=1;
     kt1:=1; kt2:=1;
          begin
               repeat
                    k1:=1;
                    While k2=1 do Ct2
                    if Tg=2 then  begin
                         k1:=0;
                         While tg=2 do Ct2
                         k1:=1;
                         end;
```

```
            k1:=0; tg:=2;
        until kt1=0;
        repeat
            k2:=1;
            While k1=1 do Ct2
            if Tg=2 then  begin
                k2:=0;
                While tg=1 do Ct2
                k2:=1;
                end;
            k2:=0; tg:=1;
        until kt2=0;
```

Ưu điểm

- Dễ tổ chức thực hiện
- Có tính chất vạn năng áp dụng cho mọi công cụ và mọi hệ thống.

Nhược:

- Độ phức tạp tỷ lệ với số lượng tiến trình và số tài nguyên găng
- Một tiến trình có thể bị ngăn chặn bởi tiến trình thứ 3
- Khi tốc độ hai tiến trình khá chênh lệch, một trong hai tiến trình phải chờ

*b. Phương pháp kiểm tra và xác lập (Phương pháp Perterson)*

Tương đương với phương pháp khoá trong sử dụng các giá trị kiểm tra là các biến trạng thái: tham số (cục bộ, toàn cục).

Giải thuật

PAR là một lệnh gồm hai tham số:

- ✓ L: cục bộ (Local)
- ✓ G: toàn cục (Global)

Chức năng PAR

Gán L = G và gán G = 1;

- ✓ Hai lệnh trên phải được thực hiện liên tục không bị chia rẽ.
- ✓ Mỗi tiến trình sẽ sử dụng hai biến là biến local của mình và biến global của toàn Chương trình.

Giải thuật

```
Var L1, L2, G: byte;
Begin
G:=0;
begin
TT:=1;
repeat
   L1:=1;
   while L1=1 do PAR(L1);
   {đoạn giữa tiến trình 1}
   G:=0;
   {phần còn lại của tiến trình 1}
until false
TT:=2;
repeat
   L2:=1;
   while L2=1 do PAR(L2);
   {đoạn giữa tiến trình 2}
   G:=0;
   {phần còn lại của tiến trình 2}
until false
end;
End;
```

Ưu điểm:

- Khắc phục được độ phức tạp của thuật toán, độ phức tạp thuật toán không phụ thuộc vào số lượng tiến trình.

Nhược điểm:

- Vẫn còn hiện tượng chờ đợi tích cực.

*c. KT đèn báo (Semaphore - Dijkstra)*

Hệ thống sử dụng biến đèn báo nguyên đặc biệt (Semaphore) s. Ban đầu s nhận một giá trị bằng khả năng phục vụ của tài nguyên găng. Hệ thống có hai phép để thao tác trên s là P(s) và V(s).

P (s): Proberen (tiếng Hà Lan) có nghĩa là giảm

Giảm S đi 1 đơn vị

Nếu s ≥ 0 tiếp tục thực hiện tiến trình

Ngược lại đưa tiến trình vào dòng xếp hàng

V (s): Verhogen có nghĩa là kiểm tra

Tăng S lên 1

Nếu s ≤ 0 kích hoạt một tiến trình ra hoạt động

Giải thuật:

```
Var s: byte;
Begin
s:=1;
begin
tt:=1;
repeat
P(s)
{đoạn giữa tiến trình 1}
V(s);
{phần còn lại của tiến trình 1}
until false
tt:=2;
repeat
    P(s)
{đoạn giữa tiến trình 2}
    V(s);
    {phần còn lại của tiến trình 2}
until false
end;
End;
```

- Đặc điểm quan trọng là 2 phép P và V là liên tục, trong quá trình thực hiện P hoặc V thì processor không bị ngắt để chuyển sang công việc khác.

- Tuy nhiên các phép xử lý này có thể không tồn tại trên các máy vì P và V phải làm việc với dòng xếp hàng và thông tin lưu trữ khá lớn. Để khắc phục điều này người ta xây dựng các thủ tục procedure để thực hiện các phép xử lý này.

+ Đầu của thân thủ tục bao giờ cũng ra lệnh cấm ngắt tức là chặn mọi tín hiệu vào processor CLI, trừ những tín hiệu bắt buộc (ngắt không che được).

+ Cuối thân thủ tục có lệnh giải phóng ngắt (STI).

*d. Công cụ điều độ cấp cao – chương trình thư kí (Monitor)*

Đặc điểm:

- Nằm ngoài tiến trình của người sử dụng
- Người sử dụng không biết tài nguyên gì và khi nào thuộc đoạn găng

Chương trình thư ký (Monitor): cấu trúc đặc biệt bao gồm các thủ tục, các biến và cấu trúc dữ liệu hoạt động trong chế độ phân chia thơi gian , hỗ trợ việc thực hiện tiến trình, với các thuộc tính:

- Các biến và cấu trúc dữ liệu trong Monitor chỉ có thể được thao tác bởi các thủ tục định nghĩa bên trong Monitor
- Tại một thời điểm, một tiến trình duy nhất được làm việc với chương trình thư ký
- Mỗi lần sử dụng tài nguyên mới, hệ thống gắn chương trình thư ký với tiến trình

Trong một Monitor có thể định nghĩa các biến điều kiện C và hai thao tác là Wait () và Signal ():

- Wait (C): chuyển trạng thái tiến trình sang trạng thái khoá và đặt tiến trình vào hàng đợi trên biến điều kiện C
- Signal (C): nếu có một tiến trình đang bị khoá trong hàng đợi của C thì tái kích hoạt tiến trình đó và tiến trình sẽ ời khỏi Monitor

Thuật toán

```
Wait (C)
begin
    status (p)=khoá
    enter (p, f (C)) { đưa p vào hàng đợi}
end;
Signal (C)
begin
    if f (C)<>nil then
    exit (q, f (C)) { đưa q ra khỏi hàng đợi}
end;
```

### *4.1.4. Tình trạng tắc nghẽn*

Tắc nghẽn: Khi có nhiều tài nguyên găng trong một tiến trình, các tiến trình sẽ rơi vào tình trạng chờ đợi lẫn nhau

Tình trạng tắc nghẽn: hai hay nhiều tiến trình cùng chờ đợi một sự kiện và nếu không có tác động đặc biệt từ ngoài thì sự chờ đợi ấy là vô hạn

- Phòng chống:
- Phòng ngừa : tránh không để tiến trình rơi vào tình trạng tắc nghẽn
- Dự báo và tránh : Kiểm tra xem tiến trình có rơi vào tình trạng tắc nghẽn hay không, thông báo kịp thời trước khi tắc nghẽn sảy ra
- Nhận biết và khắc phục : Phát hiện các tiến trình bị tắc nghẽn và giải quyết

*a. Phòng ngừa*

Xem xét các điều kiện tắc nghẽn:

- Thiếu tài nguyên Găng
- Chờ vô hạn khi chưa được vào đoạn Găng
- Không có hệ thống phân phối lại tài nguyên
- Tồn tại chờ đợi vòng

Điều kiện 1: Dùng kĩ thuật SPOOL: Khi kết thúc tiến trình thì kết quả được chuyển ngược lại tài nguyên vật lý mà sever yêu cầu, việc chuyển ngược này theo nguyên tắc lần lượt và do chương trình hệ thống đảm nhận như vậy không xảy ra xung đột

Điều kiện 2: Phân phối trước tài nguyên, tiến trình chỉ được bắt đầu khi nhận đủ tài nguyên trong một số lần phân phối

Điều kiện 3: Tạo các điểm gác: Hệ thống sẽ lưu lại toàn bộ thông tin trạng thái tiến trình, nếu cần thiết có thể huỷ tiến trình, giải phóng tài nguyên, sau đó nếu cho phép sẽ tiếp tục công việc bằng cách khôi phục trạng thái cuối.

Điều kiện 4: Chờ đợi vòng: Phân lớp tài nguyên, tiến trình chỉ nhận được tài nguyên mức cao hơn sau khi đã trả lại tài nguyên mức thấp.

*b. Dự báo và phòng tránh*

Không phòng ngừa nhưng mỗi lần phân phối tài nguyên thì kiểm tra xem việc phân phối đó có khả năng đẩy hệ thống vào tình trạng tắc nghẽn không? Nếu xuất hiện nguy cơ trên thì tìm cách giải quyết cụ thể trước khi tắc nghẽn có thể xảy ra

Thuật toán:

- Có n tiến trình
- Hệ thống có k thiết bị
- Tiến trình i yêu cầu tối đa một lúc max (i) đơn vị thiết bị để có thể thực hiện, nhưng hiện chỉ nhận được f (i) đơn vị thiết bị
- Tiến trình i kết thúc kt (i)=true

Thuật toán :

```
t:=k;
for i:=1 to n do
begin
        t:=t-f (i);
        cl[i]:=max[i];
        kt[i]:=false;
    end;
Flag:=True;
While Flag do
    begin
        flag:=false
        For i:=1 to n do
        if not kt[i] and (cl[i] <=t) then
            begin
            kt[i]:=true;
            t:=t+cl[i]
            Flag:=true;
            end;
end;
if t=k then "An toàn"
else "không an toàn"
```

*c. Nhận biết và khắc phục*

Quan sát trạng thái các tiến trình đang chờ, xem những tiến trình bị rơi vào tắc nghẽn, tuỳ tình hình cụ thể áp dụng các biện pháp cần thiết

Khi phát hiện tắc nghẽn:

- Đình chỉ hoạt động của tiến trình liên quan đưa tiến trình về trạng thái ngắt
- Thu hồi tài nguyên

Đưa tiến trình về trạng thái ngắt:

- Đưa tất cả các tiến trình trong tình trạng tắc nghẽn về ngắt.
- Đưa từng tiến trình khi không còn chu trình gây tắc nghẽn theo các tiêu chí:
    - Độ ưu tiên
    - Thời gian xử lý
    - Số lượng tài nguyên tiến trình đang chiếm dụng
    - Số lượng tài nguyên tiến trình yêu cầu

Thu hồi tài nguyên: thu hồi tài nguyên của một số tiến trình và cấp phát các tài nguyên này cho tới khi loại bỏ được chu trình tắc nghẽn

- Lựa chọn tiến trình thu hồi, những tài nguyên nào bị thu hồi
- Phục hồi trạng thái tiến trình ở trạng thái gần nhất trước đó mà không xảy ra tắc nghẽn
- Tránh cho một tiến trình nào đó luôn bị thu hồi tài nguyên

Ví dụ:

Các tiến trình: P1, P2, P3, P4

Các tài nguyên: R1, R2, R3

Tổng các tài nguyên của hệ thống k = 9R1 + 3R2 + 6R3

Trạng thái hiện thời các tiến trình:

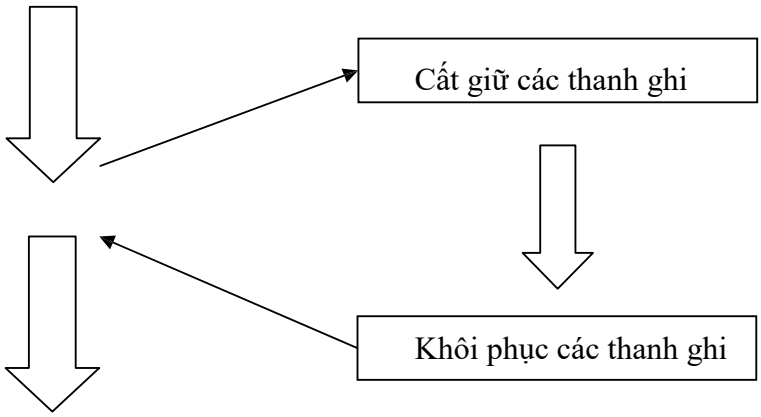
| Tiến trình | Max (i) | | | f (i) | | | t | | |
|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 3 | 2 | 2 | 1 | 0 | 0 | 4 | 1 | 2 |
| P2 | 6 | 1 | 3 | 2 | 1 | 1 | | | |
| P3 | 3 | 1 | 4 | 2 | 1 | 1 | | | |
| P4 | 4 | 2 | 2 | 0 | 0 | 2 | | | |

Giả sử P2 có yêu cầu 4R1 và 1R3, khi đó việc thoả mãn P2 có đẩy hệ thống tới tình trạng tắc nghẽn hay không?

### *4.1.5. Ngắt (Interupt)*

Phương tiện để các thiết bị trong hệ thống báo cho Processor biết việc thay đổi trạng thái của mình - công cụ chuyển điều khiển tới một tiến trình khác

- Ngắt là hiện tượng tạm ngừng thực hiện một tiến trình để chuyển sang thực hiện một tiến trình khác khi có một sự kiện xảy ra trong hệ thống tính toán.




- Có thể hiểu tạm nghĩa "thực hiện một tiến trình" là thực hiện một Chương trình, tiến trình bị ngắt có thể coi là Chương trình chính, còn tiến trình xử lý ngắt có thể coi là Chương trình con.

- Chương trình con xử lý ngắt là một Chương trình ngôn ngữ máy hoàn toàn bình thường. Chương trình này địa chỉ kết thúc bằng lệnh IRET (Interupt RETurn), nó ra lệnh cho bộ xử lý quay về thực hiện tiếp Chương trình chính đúng từ chỗ mà nó bị ngắt.

- Đối với các hệ thống tính toán việc gọi ngắt dùng cho việc các bộ phận khác nhau của hệ thống tính toán báo cho processor biết về kết quả thực hiện công việc của mình.

***Phân loại ngắt:***

- Ngắt trong: ngắt do các tín hiệu của procesor báo cho processor
- Ngắt ngoài: ngắt do các tính hiệu bên ngoài báo cho processor
- Ngắt cứng: ngắt được gọi bởi các Chương trình được cứng hoá trong các mạch điện tử.
    - Ngắt che được: (Maskable Interupt):
      Là ngắt có thể dùng mặt nạ để ngăn cho không ngắt hoạt động. Ta có thể đặt các bít trong mặt lạ bằng lệnh CLI (CLear Interupt flag).
      Ví dụ: Ngắt chuột là ngắt cứng có thể bị che
    - Ngắt không che được (Non Maskable Interupt):
      Là ngắt không thể dùng mặt nạ che được (có độ ưu tiên cao nhất)
      Ví dụ: Ngắt 2 báo hiệu có lỗi trong bộ nhớ.

- Ngắt mềm: ngắt được gọi bằng một lệnh ở trong Chương trình. Lệnh gọi ngắt từ Chương trình ngôn ngữ máy là lệnh INT (INTerupt), các lệnh gọi ngắt từ Chương trình ngôn ngữ bậc cao sẽ được dịch thành lệnh INT.
- Các ngắt khác

***Xử lý ngắt***

- ✓ Lưu đặc trưng sự kiện gây ngắt vào nơi quy định
- ✓ Lưu trạng thái của tiến trình bị ngắt vào nơi quy định
- ✓ Chuyển điều khiển tới Chương trình xử lý ngắt
- ✓ Thực hiện Chương trình xử lý ngắt, tức là xử lý sự kiện
- ✓ Khôi phục tiến trình bị ngắt

***Véc tơ ngắt:***

- Khi ngắt được tạo ra, nơi phát sinh nó không cần biết địa chỉ của Chương trình xử lý ngắt tương ứng mà chỉ cần biết số hiệu ngắt. Số hiệu này chỉ đến một phần tử trong một bảng gọi là bảng các vector ngắt nằm ở vùng có địa chỉ thấp nhất trong bộ nhớ và chứa địa chỉ của Chương trình con xử lý ngắt. Địa chỉ bắt đầu của mỗi Chương trình con được xác định bởi địa chỉ đoạn và địa chỉ offset được đặt trước đoạn.
- Hai địa chỉ này đều là 16 bit (2 byte), như vậy mỗi địa chỉ ngắt chiếm 4 byte trong bộ nhớ. Máy tính PC có 256 ngắt khác nhau được đánh số từ 0 đến 255 do vậy độ dài của cả bảng do vậy sẽ là 256*4 = 1024. Bảng vector ngắt chiếm các ô nhớ từ địa chỉ 0 đến 3FFh. Số thứ tự của ngắt bằng số thứ tự của vector ngắt. Địa chỉ của Chương trình xử lý số i được chứa trong bảng véc tơ ngắt từ địa chỉ offset 4*(i-1) đến 4*(i-1) + 3.

Một số ngắt thường dùng

| STT | Số hiệu ngắt | Chức năng | STT | Số hiệu ngắt | Chức năng |
|---|---|---|---|---|---|
| 1 | 00 | Ngắt chia cho 0 | 7 | 20H | Kết thúc Chương trình |
| 2 | 04 | Ngắt tràn số | 8 | 21H | Gọi các hàm của DOS |
| 3 | 08 | Ngắt thời gian | 9 | 25H/26H | Đọc/ghi đĩa |
| 4 | 09 | Ngắt bàn phím | 10 | 27H | Kết thúc nhưng thường trú |
| 5 | 10H | Ngắt phục vụ màn hình | 11 | 33H | Ngắt phục vụ chuột |
| 6 | 19H | Ngắt khởi động hệ thống | 12 | 67H | Quản lý bộ nhớ mở rộng |

(Tham khảo thêm Vi xử lý)

## 4.2. Quản lý Processor

***Đặt vấn đề***

Chương trình không thể thực hiện được nếu nó không được nạp vào bộ nhớ, song ngay cả khi đã được nạp vào bộ nhớ nếu nó không có quyền sử dụng Processor thì vẫn không thể thực hiện được.

- Processor: Tài nguyên phục vụ cho việc thực hiện chương trình. Đơn vị công việc giao cho processor phục vụ là tiến trình, nhiều tiến trình có thể sản sinh từ chương trình.
- Tiến trình: đối tượng mà ta có thể phân phối Processor cho nó.

### *4.2.1. Processor vật lý và Processor logic*

*Processor vật lý*: tất cả các hệ điều hành thực hiện song song đều do một Processor của hệ thống – Processor vật lý điều khiển.

*Processor logic:* người sử dụng đánh giá hoạt động của Processor trên cơ sở quan sát và đánh giá chương trình của mình được thực hiện như thế nào. Processor mà người sử dụng quan sát và đánh giá được gọi là Processor logic - liên quan tới việc thực hiện tiến trình.

Với chế độ xử lý kế tiếp đơn chương trình (Tiến trình tuần tự): $P_{VL} \approx P_{LG}$

Với các tiến trình hoạt động song song quan tâm các chiến lược điều độ Processor ( điều độ tiến trình mức Processor).

*Vấn đề cần quan tâm:*

Nên tạo ra bao nhiêu Processor logic là thích hợp

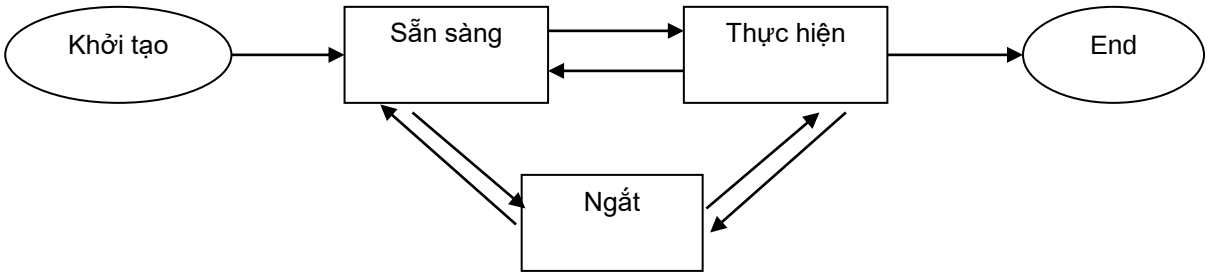
Độ dài khoảng thời gian gắn liên tục Processor vật lý cho Processor logic là bao nhiêu thì hợp lý

Sau khi một Processor logic hết quyền sử dụng Processor vật lý thì cần chọn tiến trình nào để phân phối Processor vật lý.

### *4.2.2. Phân phối Processor*

Trong chế độ đa nhiệm, mỗi tiến trình có thể thuộc một trong ba trạng thái:

- ✓Sẵn sàng
- ✓Thực hiện
- ✓Ngắt




*Trạng thái Thực hiện:* Nếu hệ thống chỉ có một Processor thì mỗi thời điểm chỉ có một tiến trình dành được Processor để thực hiện lệnh của mình. Tiến trình này nằm trong trạng thái thực hiện.

*Trạng thái Ngắt:* Nếu tiến trình không thể thực hiện tiếp được vì bị thiếu một vài điều kiện nào đó tiến trình sẽ nằm trong trạng thái ngắt. Tiến trình gọi tới một môđun nhưng môđun chưa được nạp và định vị trong bộ nhớ. Khi đó tiến trình có thể được lưu trữ tại bộ nhớ ngoài.

*Trạng thái Sẵn sàng:* Tiến trình được phân phối đầy đủ tài nguyên (trừ Processor): tiến trình nằm trong trạng thái sẵn sàng, khi processor rỗi tiến trình sẽ được thực hiện.

Tiến trình có thể rời bỏ trạng thái Thực hiện bởi một trong ba lý do:

- ✓Tiến trình đã hoàn thành mọi việc cần thiết, khi đó nó trả lại processor và chuyển sang chờ xử lý kết quả.
- ✓Tự ngắt: Tiến trình chuyển sang trạng thái ngắt khi nó chờ mội sự kiện nào đó.
- ✓Tiến trình đã sử dụng hết thời gian processor vật lý dành cho nó và được chương trình điều độ chuyển nó từ trạng thái thực hiện sang trạng thái sẵn sàng (phân phối lại tài nguyên hệ thống).

### *4.3.3. Điều độ tiến trình*

Một trong những chức năng của chương trình điều độ là chọn tiến trình để thực hiện (chọn tiến trình đã sẵn sàng và phân phối processor vật lý cho nó).

Mỗi tiến trình sẵn sàng được gắn một thứ tự ưu tiên, thứ tự này được xác định dựa vào các yếu tố:

- ✓Thời điểm hình thành
- ✓Tổng thời gian tiến trình được thực hiện
- ✓Thời gian người sử dụng dự báo kết thúc tiến trình.
- ✓Tiêu chuẩn đánh giá chất lượng điều độ: Thời gian chờ đợi xử lý – thời gian một tiến trình ở trạng thái sẵn sàng chờ được phân phối Processor vật lý.
- ✓Các chiến lược thường gặp và cơ chế tổ chức của các chiến lược đó

***A. Chế độ một dòng xếp hàng***

Nguyên tắc: đảm bảo cho mọi tiến trình được phục vụ như nhau, không có một tiến trình nào phải chờ đợi lâu hơn tiến trình khác.

- ✓Để đánh giá chất lượng điều độ ta có thể dựa vào thời gian chờ đợi trung bình của các tiến trình.
- ✓Thời gian chờ đợi của các tiến trình được tính từ khi tiến trình ở trạng thái sẵn sàng tới khi tiến trình chuyển sang trạng thái thực hiện.
- ✓Với mỗi tiến trình ta đo khoảng thời gian này nhiều lần, khi đó có thể tính được thời gian trung bình.
- ✓Kết hợp việc đo thực nghiệm và phân tích giải thuật điều độ để đánh giá chất lượng điều độ để có được thời gian chờ đợi trung bình chính xác cho các tiến trình.
- ✓Quan sát và thống kê thời gian của từng tiến trình rút ra thời gian chờ đợi trung bình của hệ thống.

*a. Chiến lược phục vụ bình đẳng FCFS (First Come First Served)*

Đảm bảo mọi tiến trình đều có một thời gian chờ đợi trung bình như nhau, các tiến trình được phục vụ đến khi nó kết thúc hoặc khi phải chuyển sang trạng thái ngắt.

ưu điểm:

- ✓Processor không bị phân phối lại
- ✓Chi phí thấp: không phải thay đổi thứ tự ưu tiên điều độ

Nhược điểm:

- ✓Tiến trình ngắn cũng phải chờ như tiến trình dài
- ✓Thời gian chờ đợi trung bình tăng vô hạn khi hệ thống tiệm cận tới khả năng phục vụ của mình
- ✓Khi gặp tiến trình bị ngắt, các tiến trình khác sẽ bị xếp hàng lâu.

*b.Chiến lược ưu tiên những tiến trình có thời gian thực hiện ngắn nhất SJN (Shortest Job Next)*

Xác định thứ tự ưu tiên điều độ trong quá trình thực hiện tiến trình chứ không phải ở lúc khởi tạo.

Đặc điểm:

- ✓Không phân phối lại Processor
- ✓Thời gian chờ đợi của các tiến trình ngắn nhỏ hơn so với phương pháp FCFS
- ✓Thời gian chờ đợi của các tiến trình dài lớn hơn so với phương pháp FCFS
- ✓Không dự đoán được khi nào tiến trình dài được thực hiện.

*c. Chiến lược ưu tiên các tiến trình có thời gian còn lại ít nhất SRT (Shortest Remaining Time)*

Nhược điểm của FCFS là các tiến trình ngắn phải chờ đợi như tiến trình dài, với SJN thì không dự đoán được khi nào tiến trình dài được thực hiện. Khắc phục các nhược điểm này: so sánh thời gian thực hiện của tiến trình dài đang được thực hiện với thời gian thực hiện tiến trình ngắn được dự báo trước để xem xét độ ưu tiên

Nếu thời gian thực hiện của tiến trình dài đang thực hiện còn lại là nhỏ hơn thì tiếp tục thực hiện tiến trình dài, ngược lại đưa tiến trình về trạng thái ngắt và thực hiện tiến trình ngắn.

*d. Chiến lược xếp hàng lần lượt RR (Round Robin) – phân phối lại Processor*

Nguyên tắc: mỗi một tiến trình trong dòng xếp hàng lần lượt được phân phối một lượng tử thời gian để thực hiện. Sau khoảng thời gian đó, nếu tiến trình chưa kết thúc hoặc không rơi vào trạng thái ngắt thì nó được chuyển về cuối dòng xếp hàng: tiến trình xếp hàng vòng tròn.

Khi có một tiến trình mới, nó sẽ được đưa vào dòng xếp hàng vòng tròn và được đặt ở vị trí được phục vụ ngay lập tức.

Với các tiến trình dài: phân thành m lớp, lớp thứ i tiến trình được phục vụ với khoảng thời gian $T_i$, sau khi đã được thực hiện, tiến trình chưa kết thúc hoặc không bị ngắt nó được chuyển sang lớp thứ i+1 với thời gian phục vụ $T_{i+1} > T_i$.

***B. Chiến lược nhiều dòng xếp hàng***

Dựa vào thông tin do người sử dụng cung cấp và kết quả phân tích của hệ thống, phân lớp các tiến trình và đưa ra chiến lược phục vụ tương ứng.

Các tiến trình có thể được phân thành các lớp:

- ✓Tiến trình thời gian thực
- ✓Tiến trình của chế độ sử dụng tập thể phân chia thời gian
- ✓Tiến trình xử lý lô

## CÂU HỎI VÀ BÀI TẬP

4.1. Anh chị hãy cho biết trên hệ điều hành đang dùng hiện đã sử dụng chiến lược điều khiển tiến trình nào? Cho ví dụ minh họa

4.2. So sánh nguyên tắc, ưu nhược điểm của các chiến lược điều độ tiến trình trong chế độ một dòng xếp hàng.

4.3. Xây dựng chương trình nhận 1 ký tự chữ thường từ bàn phím và chuyển thành ký tự chữ hoa.

4.4. Xây dựng chương trình thường trú để giám sát các ứng dụng thực hiện trên hệ điều hành Windows

# Chương V: HỆ ĐIỀU HÀNH NHIỀU PROCESSOR

## 5.1. Hệ điều hành nhiều Processor

Sự kết hợp của các Processor trong một hệ thống tính toán, sự kết hợp của các hệ thống tính toán đơn Processor.

Mục đích:

Sự chuyên môn hoá các Processor làm giảm gánh nặng xử lý

- ✓Hoạt động ổn định và năng suất cao
- ✓Độ tin cậy cao
- ✓Làm cho các tài nguyên có giá trị cao, mang tính khả dụng đối với bất kỳ người dùng người dùng nào trên mạng.
- ✓Tăng độ tin cậy của hệ thống nhờ khả năng thay thế khi xảy ra sự cố đối với một máy tính nào đó

### *5.1.1. Cấu hình nhiều Processor*

Tồn tại nhiều phương thực kết nối hai hay nhiều Processor.

Sự kết hợp của máy tính với các hệ thống truyền thông, đặc biệt là mạng viễn thông đã tạo lên mô hình tập trung các máy tính đơn lẻ được kết nối với nhau để cùng thực hiện công việc. Môi trường làm việc nhiều người dùng, cho phép nâng cao hiệu quả khai thác tài nguyên chung từ những vị trí địa lý khác nhau (bộ nhớ, chương trình, nhiệm vụ...)

***Cấu hình phân cấp: Client/ Server:*** một Processor ngoại vi và có thể hoạt động độc lập trong khi giải quyết nhiệm vụ của mình.

Đặc điểm:

- Chương trình dễ tổ chức
- Chương trình điều khiển không phải sao chép nhiều lần.
- Không phải tổ chức kiểu module vào/ra nhiều lần
- Thực hiện ngắt tăng.

***Sơ đồ liên kết mềm linh hoạt:*** Các processor có quan hệ bán phụ thuộc

- Mỗi processor xử lý tiến trình của mình từ khi hình thành tới khi kết thúc.
- Các processor có thể liên hệ, trao đổi thông tin và chuyển giao tiến trình trước khi nó được bắt đầu thực hiện.

Đặc điểm:

- Giảm gánh nặng xử lý tại một processor
- Các processor có thể trao đổi tiến trình, cơ chế điều độ đơn giản

***Sơ đồ liên kết bình quyền:*** Các processor được coi như tập các tài nguyên cùng loại

Thay cho việc thực hiện từng chương trình trên từng processor, phân chia công việc điều khiển cho tất cả các processor. Như vậy một tiến trình có thể bắt đầu ở processor này nhưng có thể kết thúc ở processor khác.

Đặc điểm:

- Giảm gánh nặng xử lý tại một processor
- Các processor có thể trao đổi tiến trình, cơ chế điều độ đơn giản
- Khó đánh giá kết quả thực hiện tiến trình

### *5.1.2. Hệ điều hành nhiều processor:*

Tồn tại một hệ điều hành có chức năng quản ly dữ liệu, tính toán và xử lý một cách thống nhất: Hệ thống như vậy gọi là hệ điều hành nhiều processor.

Với các tiếp cận:

- Tập trung: Tôn trọng hệ điều hành cục bộ đã có trên các hệ thống tính toán, hệ điều hành nhiều processor được cài đặt như một tập các chương trình tiện ích chạy trên hệ thống.
- Phân tán: Bỏ qua hệ điều hành cục bộ đã có trên các hệ thống, cài đặt một hệ điều hành thuần nhất trên toàn mạng

Với mô hình tập trung:

- Cung cấp cho mỗi người dùng một tiến trình đồng nhất làm nhiệm vụ cung cấp một giao diện đồng nhất với tất cả các hệ thống cục bộ đã có
- Tiến trình này quản lý cơ sở dữ liệu chứa thông tin về hệ thống cục bộ và về các chương trình và dữ liệu của người dùng thuần tuý:
  - ✓ Bộ xử lệnh
  - ✓ Dựng các lệnh của người dùng $\rightarrow$ ngôn ngữ lệnh của hệ thống $\rightarrow$ gửi tới P để thực hiện
- Đặc điểm:
  - ✓ Đơn giản, không làm ảnh hưởng tới các hệ thông cục bộ đã có
  - ✓ Khó thực hiện I/O

Với mô hình Phân tán:

- Mô hình tiến trình: Mỗi tài nguyên được quản lý bởi một tiến trình nào đó và hệ điều hành điều khiển sự tương tác giữa các tiến trình đó
- Mô hình đối tượng: Coi các tiến trình và các đối tượng, mỗi đối tượng có một kiểu, một biểu diễn và một tập các thao tác có thể thực hiện trên nó
  - o Như vậy:
  - o Tiến trình của uses phải được phép thao tác trên đối tượng
  - o Hệ điều hành quản lý việc thao tác của tiến trình trên đối tượng.

## 5.2. Hệ điều hành phân tán (Distribute Operating System)

### *5.2.1. Khái niệm:*

Tập các chương trình phục vụ tập trung như một giao diện quá trình ứng dụng và hệ thống tính toán nhằm đạt được tính hiệu quả an toàn, dễ sử dụng hệ thống tính toán.

Chức năng của hệ điều hành :

- Điều độ Processor
- Đồng bộ giữa các quá trình tương tác
- Quản lý tài nguyên hệ thống
- Đảm bảo điều khiển truy nhập và bảo vệ tính toàn vẹn hệ thống, phục hồi và cung cấp giao diện người dùng

Quan niệm về hệ điều hành:

- Máy ảo: Trừu tượng hoá hệ thống máy tính (mục tiêu thiết kế cơ bản)
- Quản trị tài nguyên : Phương tiện để đạt được mục đích

Như vậy:

- Hệ điều hành tập trung: Quan tâm tới việc quản trị tài nguyên hệ thống
- Hệ điều hành phân tán: Trừu tượng hoá máy tính

Vào thời điểm mới ra đời: Các hệ điều hành được thiết kế tập trung để chạy trên các hệ thống có một hay nhiều bộ xử lý (Processor)

Với tiếp cận mạng máy tính ngày nay: Các hệ thống làm trên phạm vi rộng, phân tán ở nhiều địa điểm khác nhau đòi hỏi cơ chế quản lý phân tán.

### *5.2.2. Đặc trưng của hệ điều hành phân tán*

#### *a. So với PC*

Khả năng dùng chung dữ liệu:

- Nhiều PC dùng trên nhiều bản sao của dữ liệu tại nhiều nơi, vì vậy chi phí cho việc đồng bộ quản lý truy nhập và bảo mật tốn kém
- MSDOS: Dùng trên một số ít các bản dữ liệu, chi phí giảm

Khả năng dùng chung thiết bị:

- Mỗi PC phải trang bị đầy đủ các thiết bị ngoại vi song nếu được kết nối trong môi trường MSDOS các thiết bị ngoại vi có thể được sử dụng chung bởi nhiều người dùng trong hệ thống, nhờ vậy tiết kiệm và hiệu quả

Khả năng truyền thông:

- Kết nối của PC nhờ dịch vụ mạng viễn thông, thời gian chờ đợi để được phục vụ là không an toàn.
- Với MSDOS: môi trường phân tán

Tính linh hoạt:

- Việc phân chia lại tài nguyên gây ra chi phí tốn kém: lưu chuyển tài liệu, thiết bị, dữ liệu...
- MSDOS: Sử dụng các chức năng chuyên biệt của hệ thống

#### *b. So với hệ điều hành tập trung*

Tốc độ: Năng lực kế toán cao khi tập trung một số bộ vi xử lý trên một máy tính

Tính kinh tế: Tỷ suất giá cả hiệu nâng cao

Tính phân bố: Liên kết các ứng dụng trên các máy riêng biệt

Tính ổn định và tin cậy: Hệ thống vẫn làm việc khi một máy gặp sự cố

Tính mở: Có thể từng bước mở rộng quy mô hệ thống

#### *c. Hạn chế của hệ điều hành phân tán*

- Phần mềm: đòi hỏi hệ điều hành, các ngôn ngữ hình thức, các chương trình ứng dụng phù hợp: thiết kế, cài đặt khó, phức tạp
- Vấn đề mạng: Thay thế toàn bộ hệ thống cũ
- Vấn đề truyền thông: an toàn dữ liệu, lưu lượng đường truyền, quá trình thay thế khi có sự cố.
- Vấn đề bảo mật: Giá thành cao, khó sử dụng chung dữ liệu, chương trình

#### *d. Yêu cầu thiết kế hệ điều hành phân tán*

Tính trong suốt

- Tính trong suốt với người dùng: người dùng nghĩ rằng hệ thống phân tán chỉ là một tập máy tính hoạt động ở chế độ phân chia thời gian
- Tính trong suốt hệ thống: hệ thống trong suốt đối với chương trình, lời gọi hệ thống phải được thiết kế sao cho sự có mặt của nhiều processor là không thể thấy được từ chương trình

Thể hiện:

- *Trong suốt về định vị:* người dùng không thể nói chính xác các tài nguyên nằm ở đâu (tài nguyên được mã hoá vị trí)

- *Trong suốt về ánh xạ:* tên tài nguyên không thay đổi khi di chuyển từ máy này sang máy khác
    - *Trong suốt về lặp lại:* hệ thống có thể (và cần thiết) lưu một số bản sao của cùng một tài nguyên mà người dùng không biết
    - *Trong suốt đồng thời:* nhiều tiến trình có thể cùng truy nhập một tài nguyên, các tiến trình có thể không cần biết tới sự có mặt của tiến trình khác
    - *Trong suốt song song:* nhiều hoạt động song song được che đối với người dùng
    - *Trong suốt lỗi:* cơ chế phục hồi lỗi trong hệ thống được che đối với người dùng
    - *Trong suốt kích thước:* cho phép hệ hống mở rộng qui mô dần dần ma không tác động tới người dùng
    - *Trong suốt về quan sát*: điểm nhìn phần mềm không thể thấy được đối với người dùng.

Tính modul hoá: Hệ thống được phân chia làm nhiều modul như cho phép bổ sung, thay đổi dễ dàng

Tính khả mở: qui mô hệ thống thường xuyên thay đổi do các yêu cầu nâng cấp

Tính độc lập, quy mô: Mở rộng quy mô mà năng lực hệ thống không thay đổi

Tính chịu lỗi: Thường xuyên sao lưu để phục hồi lỗi

## 5.3. Quản lý tài nguyên trong hệ điều hành phân tán

### *5.3.1. Quản lý thiết bị, quản lý File*

Khái niệm File: đơn vị thông tin nhỏ nhất của người dùng, được quản lý thông qua tên file.

- Người dùng phải lưu trữ thông tin ở bộ nhớ ngoài vì vậy hệ điều hành phải có vai trò sao cho người dùng truy nhập thuận tiện
- Nhu cầu dùng chung (chia sẻ) các file dữ liệu.
- Vấn đề đặt ra đối với hệ thống quản lý file: ngoài các tính chất và yêu như đối với hệ quản lý file trong hệ điều hành tập trung, hệ quản lý file trong hệ điều hành phân tán phải đảm bảo:
- Tính trong suốt của hệ thống
- Dịch vụ thư mục
- Hiệu năng hệ thống, độ tin cậy
- Độ an toàn.

***Tính trong suốt***

- Tính trong suốt đăng nhập: người dùng có thể đăng nhập vào các trạm trong hệ thống với cùng một thủ tục đăng nhập
- Trong suốt truy cập: Các tiến trình chạy trên hệ thống có cùng cơ chế truy nhập vào các tệp tin mà không cần để ý xem tệp đó là cục bộ hay từ xa
- Sự độc lập về định vị tệp tin: Các tệp tin có thể được chuyển từ vị trí này tới vị trí khác mà không làm thay đổi tên: trong suốt đối với người dùng
- Tính trong suốt tương tranh: các file được chia sẻ bởi nhiều người dùng, việc truy cập tới một tệp từ một tiến trình không ảnh hưởng tới sự thành lập của tiến trình khác
- Trong suốt lặp: Các tệp được sao lưu để dự phòng cho phép truy nhập đồng bộ (người dùng không biết các bản sao).

***Thiết kế và thực hiện hệ thống tệp tin phân tán***

Đối với người dùng, một tệp tin bao gồm ba thành phần logic:

- Tên tệp và hệ thống tệp
- Các thuộc tính
- Các đơn vị dữ liệu

Các tệp và hệ thống tệp

Các tệp được tạo ra bởi người dùng đi kèm với tên, khi truy nhập tệp, tên tệp sẽ xác định giá trị ID của tệp và giá trị này cũng là giá trị duy nhất xác định vị trí vật lý của tệp

Các thuộc tính: Các thông tin về quyền sở hữu, quyền truy nhập, dạng tệp, kích thước, dấu hiệu thời gian

Các đơn vị dữ liệu

Đơn vị dữ liệu : Byte, khối

Cơ chế truy nhập:

- Tuần tự: Con trỏ định vị tệp được duy trì bởi hệ thống cho phép xác định vị trí đơn vị dữ liệu kế tiếp được truy nhập giữa các tiến trình
- Trực tiếp (truyền thông không liên kết): Vị trí đơn vị dữ liệu cho việc đọc, ghi là rõ ràng. Cơ chế này liên quan tới kích thước của đơn vị dữ liệu, các thao tác đọc ghi phải bao hàm các thông tin điều khiển.
- Chỉ số: Đơn vị dữ liệu được địa chỉ hoá bởi chỉ số hay khoá đi kèm mỗi khối dữ liệu.

***Vấn đề bảo mật***

Bảo vệ dữ liệu: không để mất thông tin khi có sự cố kỹ thuật hoặc chương trình thậm chí truy nhập bất hợp lệ

Kĩ thuật bảo vệ dữ liệu cho hệ phân tán:

Phương pháp mã hoá dữ liệu với thuật toán DBS

- Khoá bí mật: Thuật toán giải mã
- Khoá công khai: sinh mã:

Kerberos: Sự xác nhận là đúng của các thành phần dựa trên cơ sở tin tưởng vào thành phần thứ 3 (mật khẩu)

Chữ ký điện tử: Xác nhận tính nguyên bản mà các văn bản (Digital Signature)

### *5.2.2. Quản lý bộ nhớ*

Ngoài các phương pháp quản lý bộ nhớ như trong hệ điều hành tập trung, vấn đề quan tâm trong việc quản lý bộ nhớ ở hệ điều hành phân tán đó là việc đảm bảo tính chia sẻ bộ nhớ.

Chia sẻ bộ nhớ: Truy nhập bộ nhớ từ xa:

Việc truy nhập được thực hiện tại một nút xa

- Khối dữ liệu xa được di chuyển tới nút cục bộ: truy nhập cục bộ
- Khối dữ liệu xa được sao lưu lại tại nút cục bộ: truy nhập đồng bộ

***Các phương thức***

- Đọc từ xa (Read remote): Khối dữ liệu dùng chung không được di chuyển hay sao lưu, máy trạm gửi yêu cầu tới máy chủ, máy chủ gửi trả lời về dữ liệu cho việc đọc, và báo nhận cho việc ghi.
- Đọc/ghi ánh xạ (Read/write migrate): Nhờ việc truy cập tới một khối dữ liệu từ xa mà khối dữ liệu được di chuyển tới tiến trình yêu cầu. Tiến trình sẽ cập nhập tới bảng ánh xạ khối vật lý - trang ảo của dữ liệu

### *5.2.3. Quản lý tiến trình*

Khái niệm về tiến trình: Đơn vị thực hiện được nhỏ nhất thấy bởi người dùng

Luồng (Thread): đơn vị thực hiện được nhỏ nhất thấy bởi hệ điều hành, được hệ điều hành cấp phát thời gian Processor.

***Quan hệ giữa tiến trình và luồng:***

Tiến trình là không gian địa chỉ trong đó luồng được thực hiện. Hai tiến trình cùng không gian địa chỉ - hai luồng thuộc một tiến trình.

***Quản lý tiến trình:*** việc quản lý các tiến trình thông qua các khối điều khiển tiến trình.

Khối điều khiển: Bản ghi chứa các khối điều khiển các luồng, các cổng thông tin , các tài nguyên hệ thống mà tiến trình đang sử dụng, các thông tin trạng thái tiến trình: Sẵn sàng, thực hiện và ngắt

***Quản lý luồng:*** Khối điều khiển luồng:

Bộ đếm lệnh: PC

- Con trỏ ngăn xếp: SP
- Tập các thanh ghi: Rs
- Trạng thái: Flag

Các chức năng quản lý tiến trình và luồng thông tin chia làm ba loại

- Truyền thông: Đảm bảo sự liên kết giữa các tiến trình
- Đồng bộ: Đảm bảo thực hiện các tiến trình tối ưu
- Điều độ : Đảm bảo các tiến trình sử dụng tài nguyên chia sẻ đúng đắn

***Cài đặt luồng:***

Trong không gian người dùng: Khi luồng gọi một thủ tục hệ thống nó thực hiện liên kết vào thư viện động. Thủ tục thư viện động kiểm tra xem có cần treo luồng đó không, nếu cần nó treo luồng này và chuyển điều khiển cho luồng khác.

Trong nhân hệ thống: Khi luồng gọi một thủ tục hệ thống, nó sẽ được gắn vào nhân hệ thống.

***Truyền thông giữa các tiến trình***

Mô hình truyền thống OSI: mỗi tầng có một chức năng riêng, thông điệp truyền giữa hai ứng dụng dựa trên giao thức, khi qua mỗi tầng nó được gắn thêm vùng header.

Mô hình Cilent/ Server

- Client truyền thông điệp cho server yêu cầu dịch vụ
- Server thực hiện dịch vụ tương ứng và gửi thông điệp trả lời

Các vấn đề:

- Định vị yêu cầu từ Clinet nào:
- Gắn cho mỗi Client một địa chỉ ID
- Client chọn địa chỉ ngẫu nhiên, thông báo được gửi cho taat cả các Server

Đưa tên Server vào Client khi chạy chương trình.

- Chế độ chuyển thông điệp: khoá, không khoá:
  - Khoá: khi có một thông điệp được chuyển, tiến trình của Client bị treo và chờ cho tới khi có trả lời hoặc báo lỗi
  - Không khoá: tiến trình vẫn tiếp tục thực hiện các công việc khác
- Chế độ có bảo đảm và không bảo đảm:
  - Có bảo đảm: Server nhận được thông điệp từ Client nó sẽ phúc đáp lại để Client biết.
  - Không bảo đảm: khi tiến trình gửi thông điệp, nó không được bảo đảm là thông điệp đã đến đích

Mô hình truyền thông nhóm:

Nhóm: tập các tiến trình, vì vậy khi một thành viên nhận được thông điệp tất cả các tiến trình trong nhóm đều có thể được chia sẻ.

***Đồng bộ các tiến trình***

Đồng bộ: Đảm bảo thứ tự thực hiện đúng đắn của các luồng, các tiến trình

Đồng bộ đồng hồ thời gian thực:

Giả sử có tệp .OBJ trên một máy được biên dịch từ tệp A.ASM trên một máy khác. Từ một máy thứ ba, người dùng gọi trình LINK để tạo A.EXE từ A.OBJ. Trình liên kết so sánh thời gian

cập nhập cuối cùng của A.OBJ và A.ASM để quyết định có biên dịch lại A.ASM hay không. Nếu đồng hồ của máy chứa tệp .OBJ nhanh hơn đồng hồ của máy chứa .ASM thì có thể .ASM đã cập nhật mà .OBJ vẫn mới hơn, kết quả là LINK không liên kết lại .ASM và dùng .OBJ cũ dẫn đến sai mà không biết vì sao.

Khắc phục: đồng bộ thời gian thực:

- Mô hình chuẩn tập trung: các máy trạm đều đặn gửi thông điệp hỏi giờ tới máy chủ chuẩn để thường xuyên hiệu chỉnh gìơ của mình.
- Mô hình chuẩn trung bình: máy chủ đều đặn hỏi các máy trạm giwof của chúng, tính trung bình rồi gửi phản hồi lại thời gian chung.
- Mô hình trung bình phân tán: chia thời gian thành các khoảng đồng bộ lại $t_i = T_0 + i*R$, cứ mỗi thời điểm $t_i$ mỗi máy gửi thời gian trỏ bởi đồng hồ của mình cho mọi máy khác và cũng nhận thời gian từ mọi máy khác gửi tới, tính trung bình và hiệu chỉnh đồng hồ của mình.

Đồng bộ thời gian logic

Giả sử tiến trình A gửi thông điệp cho B, thời điểm thông điệp xuất phát là $t_1$, thời điểm nhận thông điệp là $t_2$. Vì xung nhịp của hai máy khác nhau nên có thể $t_2 < t_1$ khi đó B sẽ huỷ thông điệp

Khắc phục: Trong thông điệp bao hàm cả thời gian xuất phát.

Đồng bộ thứ tự sử dụng đoạn Găng

Thuật toán tập trung: Tiến trình định sử dụng tài nguyên Găng nó sẽ gửi thông điệp tới server xem có quyến sử dụng không?

Thuật toán phân tán: Tiến trình sử dụng tài nguyên Găng nó sẽ gửi thông điệp tới các tiến trình khác. Các tiến trình khi nhận được thông điệp xin phép:

- Nếu nó không ở trong đoạn găng, không có nhu cầu sử dụng tài nguyên găng, nó sẽ cho phép.
- Nếu nó đang trong đoạn găng, nó không trả lời và xếp hàng thông điệp mới đến để trả lời sau.

Nếu nó đang định sử dụng đoạn găng, so sánh thời gian gửi thông điệp của nó trước đây với thời gian gửi của thông điệp mới đến, nếu thấy thông điệp này xuất phát trước nó sẽ tự động đi vào trạng thái chờ và trả lời cho phép.

## CÂU HỎI VÀ BÀI TẬP

5.1. Xây dựng chương trình truyền 1 ký tự giữa 2 máy tính

5.2. Xây dựng chương trình khởi động 1 chương trình từ xa trên hệ thống mạng máy tính