**ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

## Tài liệu hướng dẫn thực hành

# HỆ ĐIỀU HÀNH


Biên soạn: ThS Phan Đình Duy

ThS Nguyễn Thanh Thiện

KS Trần Đại Dương

KS Trần Hoàng Lộc

# MỤC LỤC

# NỘI QUY THỰC HÀNH

1. Sinh viên tham dự đầy đủ các buổi thực hành theo quy định của giảng viên hướng dẫn (GVHD) (6 buổi với lớp thực hành cách tuần hoặc 10 buổi với lớp thực hành liên tục).
2. Sinh viên phải chuẩn bị các nội dung trong phần "Sinh viên viên chuẩn bị" trước khi đến lớp. GVHD sẽ kiểm tra bài chuẩn bị của sinh viên trong 15 phút đầu của buổi học (nếu không có bài chuẩn bị thì sinh viên bị tính vắng buổi thực hành đó).
3. Sinh viên làm các bài tập ôn tập để được cộng điểm thực hành, bài tập ôn tập sẽ được GVHD kiểm tra khi sinh viên có yêu cầu trong buổi học liền sau bài thực hành đó. Điểm cộng tối đa không quá 2 điểm cho mỗi bài thực hành.

# Bài 6. QUẢN LÝ BỘ NHỚ

## 6.1 Mục tiêu

- Sinh viên nắm rõ thế nào là quản lý bộ nhớ, hiểu được các trường hợp lỗi trang và các giải thuật thay thế trang.
- Đánh giá được ưu nhược điểm của các giải thuật thay thế trang.
- Viết được chương trình mô phỏng các giải thuật thay thế trang bằng ngôn ngữ C trong môi trường Linux.

## 6.2 Nội dung thực hành

- Áp dụng các giải thuật FIFO, OPT, LRU để xử lý lỗi trang và thực hiện thay thế trang.
- Đánh giá mức độ hiệu quả của các giải thuật.

## 6.3 Sinh viên chuẩn bị

### 6.3.1 Các loại bộ nhớ

Có 03 loại bộ nhớ chính:

Cache: còn gọi là bộ nhớ đệm, nằm giữa CPU và bộ nhớ chính RAM. Bộ nhớ đệm thường được tích hợp trực tiếp lên chip, khi xử lý tốc độ truy xuất trên cache nhanh hơn rất nhiều lần so

với RAM, tuy nhiên chi phí để sản xuất cache khá đắt vậy nên bộ nhớ cache thường có dung lượng thấp (3M, 6M, 8M).

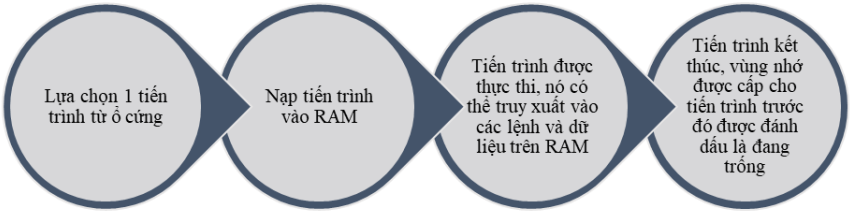
RAM: là bộ nhớ chính, khi các process chạy sẽ được nạp/cấp phát một vùng nhớ trên RAM. Khi thực thi các lệnh, CPU sẽ nạp/lưu các giá trị tính toán được từ/lên RAM. CPU truy xuất lên RAM thông qua các đường bus dữ liệu, tốc độ truy xuất này không bằng cache, tuy nhiên giá thành sản xuất lại rẻ hơn. Hiện tại, dung lượng RAM phổ biến trên các máy tính phổ thông là từ 2GB – 8GB, ngoài ra trên các máy trạm, dung lượng RAM cũng có thể lên đến 32GB hoặc hơn.

Disk: còn gọi là ổ cứng, là bộ nhớ được dùng để lưu trữ các dữ liệu và phần mềm trong hệ điều hành. Chi phí sản xuất ổ cứng tương đối rẻ do đó dung lượng của ổ cứng ngày nay khá lớn, có thể lên đến hàng TB.

Như đã tìm hiểu trong các phần trước, một chương trình khi chưa thực thi sẽ được lưu trữ trên ổ cứng. Khi được thực thi, chương trình đó sẽ được đưa vào 1 tiến trình và chuyển vào RAM. Trong quá trình thực thi, CPU sẽ thực thi các lệnh, truy xuất các dữ liệu thông qua các địa chỉ trên RAM, ngược lại về phía RAM, nó chỉ nhìn thấy các đường địa chỉ được truy xuất và không quan tâm đến việc địa chỉ đó từ đâu mà ra và được dùng vào mục đích gì.

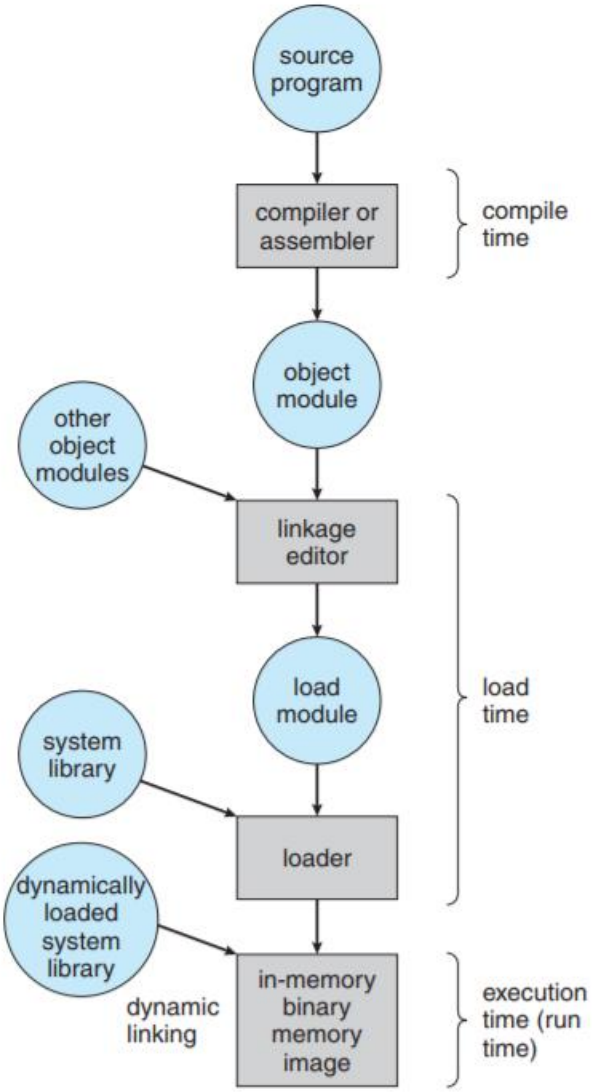
### 6.3.2 Cấp phát vùng nhớ trong hệ điều hành

Thông thường, các chương trình khi chưa được thực thi sẽ được lưu trong ổ cứng dưới dạng file thực thi bằng mã nhị phân (ví dụ như file *.exe trong hệ điều hành Windows). Để thực thi, chương trình phải được nạp lên RAM và đặt trong 1 tiến trình. Tùy thuộc vào hệ thống quản lý bộ nhớ được sử dụng, tiến trình có thể được chuyển giữa RAM và ổ cứng trong quá trình thực thi. Tiến trình trong ổ cứng chờ để được mang trở lại RAM để tiếp tục thực thi hình thành nên một hàng chờ gọi là input queue.




*Hình 6-1 Quy trình thực thi một tiến trình từ góc nhìn của RAM*

Trong hầu hết các trường hợp, một chương trình sẽ trải qua các bước sau – một vài bước có thể được lược bỏ ở vài trường hợp – trước khi được thực thi.




*Hình 6-2 Các bước để thực thi một chương trình*

Địa chỉ có thể được biểu diễn theo những dạng khác nhau qua mỗi bước kể trên. Địa chỉ trong chương trình gốc thường là địa chỉ luận lý. Trình biên dịch sẽ chuyển đổi những địa chỉ luận lý này

thành các địa chỉ tương đối. Sau quá trình nạp vào bộ nhớ (load time), các địa chỉ tương đối trên sẽ được chuyển thành các địa tuyệt đối để thực thi.

Việc chuyển đổi các địa chỉ được mô tả bên trên có thể được thực hiện thông qua bất kỳ bước nào sau đây:

Complie time: nếu như trong quá trình biên dịch đã biết được tiến trình sẽ được đặt ở đâu trong bộ nhớ chính, địa chỉ tuyệt đối có thể được sinh ra ngay lúc này. Ví dụ, nếu như đã biết tiến trình sẽ được đặt vị trí bắt đầu từ R, như vậy các đoạn mã sau khi được biên dịch sẽ được bắt đầu từ R và mở rộng ra tùy thuộc vào độ lớn của nó. Nếu sau đó, vị trí bắt đầu cấp phát bị thay đổi, quá trình biên dịch bắt buộc phải được thực hiện lại một lần nữa. Các chương trình dạng .COM của hệ điều hành MS-DOS thực hiện theo cách này.

Load time: nếu như trong quá trình biên dịch không biết tiến trình sẽ được chính xác ở đâu trong bộ nhớ chính, trình biên dịch lúc này sẽ chuyển đổi các địa chỉ luận lý của chương trình thành các địa chỉ tương đối. Trong trường hợp này, việc chuyển thành địa chỉ tuyệt đối sẽ phụ thuộc vào thời gian nạp (load time). Nếu như địa chỉ bắt đầu cấp phát bị thay đổi, chúng ta chỉ cần reload lại để thống nhất lại sự thay đổi này.

Execution time: nếu như tiến trình có thể bị di chuyển trong quá trình thực thi – từ segment này sang segment khác, như vậy

việc chuyển đổi thành địa chỉ luận lý phải chờ cho đến bước thực thi tiến trình. Các phần cứng đặc biệt phải được đáp ứng cho cơ chế này. Hầu hết các hệ điều hành ngày nay sử dụng phương thức này.

### 6.3.3 Câu hỏi chuẩn bị

Sinh viên chuẩn bị câu trả lời cho những câu hỏi sau trước khi bắt đầu phần thực hành:

1. Lỗi trang là gì? Khi nào xảy ra trường hợp lỗi trang? Vẽ lưu đồ mô tả cách hệ điều hành xử lý lỗi trang? Trình bày cách cài đặt Demand paging?
2. Tại sao phải thực hiện chiến lược thay thế trang?
3. Vẽ lưu đồ thuật toán mô tả cách thức xử lý của 3 giải thuật thay thế trang: FIFO, OPT, LRU? Vẽ sơ đồ trình bày thay thế trang bằng 3 giải thuật trên với chuỗi tham chiếu:

   *0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1*

   Giải sử có 3 khung trang và các khung trang ban đầu là rỗng. Xác định số Page Fault từ đó đưa ra đánh giá các giải thuật.

## 6.4 Hướng dẫn thực hành

Sử dụng ngôn ngữ lập trình C viết chương trình mô phỏng các giải thuật thay thế trang đã nêu trong câu số 3 mục **6.3.3** với các yêu cầu sau:

- Giao diện ban đầu của chương trình được thể hiện như bên dưới:

--- **Page Replacement algorithm** ---

1. Default referenced sequence
2. Manual input sequence

Trong đó **"Default referenced sequence"** được sinh ra từ mã số sinh viên kèm với mã "007", ví dụ sinh viên có mã số là 13520462 thì "Default referenced sequence" sẽ là: 1, 3, 5, 2, 0, 4, 6, 2, 0, 0, 7.

"Manual input sequence" là chuỗi được người dùng nhập vào.

- Sau khi lựa chọn chuỗi tham chiếu, chương trình chuyển sang giao diện cho phép người dùng nhập số khung trang:

--- **Page Replacement algorithm** ---

Input page frames:

- Sau khi lựa chọn khung trang, chương trình chuyển sang giao diện cho phép người dùng lựa chọn giải thuật thay thế trang:

--- **Page Replacement algorithm** ---

1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm

- Sau khi lựa chọn giải thuật sử dụng, chương trình hiện kết quả như sau, ví dụ:

Có chuỗi tham chiếu: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5. Có 3 khung trang và chọn giải thuật FIFO chương trình hiển thị output như sau:

```
--- Page Replacement algorithm ---
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 4 4 4 5 5 5 5 5 5
  2 2 2 1 1 1 1 1 3 3 3
    3 3 3 2 2 2 2 2 4 4
* * * * * * *       * *
Number of Page Fault: 9
```

## 6.5 Bài tập ôn tập

1. Nghịch lý Belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lý này.
2. Nhận xét về mức độ hiệu quả và tính khả thi của các giải thuật FIFO, OPT, LRU.
    - ❖ Giải thuật nào là bất khả thi nhất? Vì sao?
    - ❖ Giải thuật nào là phức tạp nhất? Vì sao?