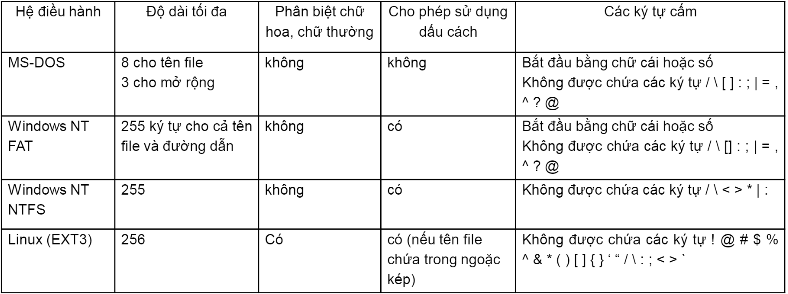
# **Chương 1: Giới thiệu về hệ điều hành**

# **Chương 2: Hệ thống File**

1. **Quy tắc đặt tên File**
2. **Hệ điều hành có cần biết và hỗ trợ các kiểu cấu trúc file không ? Tại sao ?**

Việc hỗ trợ cấu trúc file ở mức hệ điều hành có 1 số ưu và nhược điểm.

Ưu điểm:

* Các thao tác với file sẽ dễ dàng hơn đối với người lập trình ứng dụng
* HDH có thể kiểm soát được các thao tác với file

Nhược điểm:

* Tăng kích thước hệ thống
* Tính mềm dẻo của HDH bị giảm

Vì các nhược điểm nêu trên nên đa số hệ điều hành không hỗ trợ và quản lý kiểu cấu trúc file. Cấu trúc file sẽ do chương trình ứng dụng và người dùng tự quản lý. UNIX, DOS, WINDOWS là các hệ điều hành như vậy. Trong các hệ điều hành này, các file được xem như tập hợp (không có cấu trúc) các byte. Các chương trình ứng dụng khác nhau sẽ tự tạo ra và quản lý cấu trúc file riêng của mình.

1. **Các thao tác cơ bản khi làm việc với File.**

**Phân tích thao tác “Mở file” và cho biết 1 hệ thống file có nhất thiết phải có thao tác này không ?**

* Những thao tác với file thường gặp:
* Tạo file: Tạo file trống chưa có dữ liệu; File được dành 1 chỗ trong thư mục kèm theo một số thuộc tính như thời gian tạo file, tên file, người tạo file
* Xoá file: Giải phóng không gian mà dữ liệu của file chiếm trên đĩa sau đó giải phóng chỗ của file trong thư mục. Việc giải phóng không gian có thể đơn thuần là đánh dấu không gian như không gian tự do.
* Mở file: Được thực hiện trước khi đọc hoặc ghi file. Thực chất của việc mở file là đọc các thuộc tính của file và vị trí file trên đĩa vào bộ nhớ để tăng tốc độ cho các thao tác đọc ghi tiếp theo.
* Đóng file: Xóa các thông tin về file ra khỏi bảng trong bộ nhớ để nhường chỗ cho các file sắp mở. Rất nhiều hệ điều hành hạn chế số lượng file có thể mở cùng một lúc, do đó, việc đóng các file đã truy cập xong là rất quan trọng.
* Ghi vào file: Vị trí của file trên đĩa được xác định từ thông tin ghi trong thuộc tính của file. Thông thường, thông tin này đã đọc vào bộ nhớ khi tiến hành thao tác mở file. Dữ liệu được ghi vào vị trí hiện thời. Nếu vị trí hiện thời là cuối file, thông tin sẽ được thêm vào và kích thước file tăng lên. Nếu vị trí này không phải cuối file, thông tin ở vị trí đó sẽ bị ghi đè lên.
* Đọc file: Thông tin ở vị trí hiện thời sẽ được đọc.
* Định vị (seek): Đối với file truy cập trực tiếp, thao tác định vị cho phép xác định vị trí hiện thời để tiến hành đọc hoặc ghi.
* Đọc thuộc tính của file: Một số chương trình trong quá trình làm việc cần đọc các thuộc tính của file như quyền truy cập cho hệ thống bảo mật.
* Thay đổi thuộc tính của file: Một số thuộc tính của file có thể được đặt lại giá trị.
* Cắt bỏ nội dung file: Đây là thao tác xoá bỏ nội dung file và giải phóng vùng không gian do file chiếm nhưng vẫn giữ lại các thuộc tính của file trừ độ dài và vị trí.
* Khóa file (lock). Khi đồng thời nhiều tiến trình thay đổi một nội dung file, có thể dẫn đến kết quả không mong đợi nên một số HĐH cho phép khoá file. Khi một tiến trình mở file, thì tiến trình đó có thể yêu cầu khoá file và không cho phép các tiến trình khác truy cập hoặc có thể đọc file nhưng không thể ghi vào file đó.
* Một hệ thống file không nhất thiết phải có thao tác “Mở file” nhưng thường được thêm vào do nó giúp tăng hiệu quả thao tác vào ra với file.

Nếu file không được mở trước khi đọc ghi, hệ điều hành phải thực hiện nhiều công đoạn tìm kiếm file và đọc thông tin trước khi thực hiện thao tác đọc ghi, dẫn đến lặp lại công đoạn này.

Quá trình mở file bao gồm tìm kiếm file trong thư mục, đọc thông tin từ khoản mục của file vào bảng các file đang mở trong bộ nhớ. Thông tin đọc thường bao gồm tên file, quyền truy cập, chủ sở hữu file, và vị trí các khối chứa nội dung file trên đĩa.

Sau khi mở file, thao tác trả về chỉ số ứng với file trong bảng đó, được sử dụng cho các thao tác đọc ghi sau này.

Trong môi trường nhiều người dùng, quá trình mở file trở nên phức tạp hơn vì có thể xảy ra trường hợp nhiều người cùng yêu cầu mở file. Hệ điều hành sẽ xây dựng bảng lưu trữ thông tin chung và mỗi tiến trình sẽ có bảng riêng để quản lý việc sử dụng file.

Thông tin về file đang mở thường bao gồm con trỏ tới vị trí hiện thời trong file, vị trí file trên đĩa, số tiến trình mở file và quyền truy cập của người dùng đối với file.

1. **Khái niệm thư mục. Thông tin trong các khoản mục có nhất thiết phải lưu trữ gần nhau không ?**

* Không gian của đĩa được chia thành các phần (partition) gọi là đĩa lôgic. Để quản lý các file trên mỗi đĩa lôgic, thông tin về file được lưu trong hệ thống thư mục. Thư mục được tạo thành từ các khoản mục và mỗi khoản mục tương ứng với một file. Khoản mục chứa thông tin của file: tên, kích thước, vị trí, kiểu file,... hoặc con trỏ tới nơi lưu trữ thông tin này.

Thư mục như 1 bảng, mỗi dòng là khoản mục ứng với 1 file, việc tìm ra dòng cần thiết thực hiện theo tên file. Thư mục cho phép ánh xạ tên file vào file đó.

* Không cần thiết ?

1. **Cấu trúc thư mục 1 mức và 2 mức**

* Thư mục 1 mức:
* Ảnh có chứa biểu đồ, bản phác thảo, vòng tròn, hàng

  Mô tả được tạo tự độngLà kiểu tổ chức thư mục đơn giản nhất. Hệ thống chỉ có một thư mục duy nhất và tất cả các file được giữ trong thư mục này.
* Do các file được sắp xếp trong một thư mục duy nhất nên để có thể phân biệt, tên file không được phép trùng nhau => khó chọn tên file.
* Số lượng file trong thư mục rất lớn, do vậy việc tìm được file trong thư mục sẽ tốn nhiều thời gian.
* Thư mục 2 mức:
* Phân cho mỗi người dùng một thư mục riêng chỉ chứa các file do người đó tạo.
* Hệ thống vẫn duy trì một thư mục gọi là thư mục gốc. Các khoản mục ứng với thư mục người dùng được chứa trong thư mục gốc này.
* Ảnh có chứa biểu đồ, hàng, nghệ thuật gấp giấy origami, thiết kế

  Mô tả được tạo tự độngMỗi khi người dùng truy cập file, file sẽ được tìm kiếm trong thư mục ứng với tên người đó (tên đăng nhập). Do vậy, những người dùng khác nhau có thể đặt tên file trùng nhau mà không sợ lẫn lộn.
* Cách phân chia file độc lập như vậy rất thuận tiện khi người sử dụng làm việc độc lập với nhau nhưng cũng tạo tính cô lập người dùng. Hệ thống có thể tồn tại một loạt những file thường được nhiều người dùng truy cập tới như file hệ thống, những tiện ích,... Để sử dụng được các chương trình như vậy có thể chép chúng vào từng thư mục của từng người dùng. Cách này rõ ràng gây lãng phí không gian nhớ vì phải lưu nhiều bản sao của chương trình.

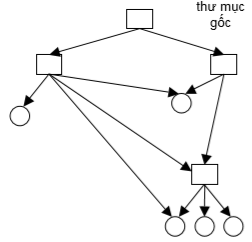
1. **Cấu trúc thư mục dạng cây và cấu trúc thư mục dạng đồ thị không có chu trình. Ưu điểm của dạng đồ thị không chu trình so với dạng cây là gì ?**

* Thư mục cấu trúc cây:
* Ảnh có chứa biểu đồ, hàng, thiết kế, nghệ thuật gấp giấy origami

  Mô tả được tạo tự độngMỗi thư mục con có thể chứa các thư mục con khác và các file. Hệ thống thư mục khi đó có thể biểu diễn phân cấp như một cây. Các cành là thư mục, còn lá là các file.
* Mỗi thư mục hoặc thư mục con khi đó sẽ chứa tập hợp các file và các thư mục con mức dưới. Để phân biệt khoản mục của file với khoản mục chỉ tới thư mục con mức dưới, người ta thường sử dụng một bit đặc biệt chứa trong khoản mục. Nếu bit này bằng 1, thì đó là khoản mục của thư mục mức dưới, nếu bằng 0, đó là khoản mục của file.
* Bản thân thư mục (hoặc thư mục con) cũng được lưu trữ trên đĩa như một file song được hệ điều hành sử dụng khác với các file thông thường. Để tạo và xoá thư mục con, hệ điều hành định nghĩa những lời gọi hệ thống riêng, khác với lời gọi hệ thống dành việc tạo và xoá file.
* Tại mỗi thời điểm, người dùng làm việc với một thư mục gọi là thư mục hiện thời hay thư mục làm việc. Trong quá trình làm việc, người dùng có thể di chuyển sang thư mục khác tức là thay đổi thư mục hiện thời.
* Tổ chức cây thư mục cho từng đĩa:

+ Trong một số hệ thống file như FAT của DOS cây thư mục được xây dựng cho từng đĩa. Hệ thống thư mục sẽ được được coi như “rừng” trong đó mỗi cây “mọc” trên một đĩa.

+ Trong các hệ điều hành khác như Linux, toàn hệ thống chỉ gồm một cây thư mục to, các đĩa sẽ là các cành mọc ra từ cây này.

* Thư mục cấu trúc đồ thị không tuần hoàn/ chu trình:
* Cho phép chia sẻ file và thư mục để có thể xuất hiện trong nhiều thư mục khác nhau. Đây là một cấu trúc đồ thị không chứa chu trình, là dạng mở rộng của cấu trúc cây, trong đó lá và cành có thể đồng thời thuộc về những cành khác nhau.
* Cách triển khai:

+ Sử dụng liên kết (link): con trỏ tới thư mục hoặc file khác.

+ Tạo ra các bản sao của các file và thư mục cần chia sẻ rồi chứa vào những thư mục khác nhau. Hệ điều hành sau đó phải theo dõi để đảm bảo tính đồng bộ và nhất quán cho các bản sao đó. Mặc dù thư mục cấu trúc đồ thị không tuần hoàn cho phép quản lý hệ thống thư mục mềm dẻo hơn cấu trúc cây song cũng phức tạp hơn cấu trúc cây rất nhiều. Do file và thư mục có mặt ở nhiều nơi khác nhau trong hệ thống thư mục, trong quá trình sao lưu và thống kê, chẳng hạn khi ta cần đếm số lượng file, phải tính tới các liên kết để tránh sao lưu hoặc thống kê cùng một file nhiều lần.

* Ưu điểm cấu trúc đồ thị không chu trình so với dạng cây:
* Cho phép chia sẻ file và thư mục để có thể xuất hiện trong nhiều thư mục khác nhau.
* Quản lý hệ thống thư mục mềm dẻo hơn cấu trúc cây song cũng phức tạp hơn cấu trúc cây rất nhiều.

1. **Thế nào là đường dẫn tuyệt đối và đường dẫn tương đối ?**

* **Đường dẫn tuyệt đối** là đường dẫn đi từ gốc của cây thư mục dẫn tới file, bao gồm tất cả các thư mục ở giữa. Các thành phần của đường dẫn, tức là tên các thư mục ở giữa, được ngăn cách với nhau và với tên file bởi các ký tự đặc biệt.

VD trong DOS: c:\bc\bin\bc.exe

VD trong Linux: /usr/ast/mailbox.

Việc sử dụng đường dẫn thuyệt đối cho phép chỉ ra vị trí của file trong cây thư mục mà không cần biết thư mục hiện thời là thư mục nào.

* **Đường dẫn tương đối** là đường dẫn tính từ thư mục hiện thời. Để có thể sử dụng đường dẫn tương đối, đa số hệ điều hành đưa thêm vào mỗi thư mục hai khoản mục đặc biệt “.” - biểu diễn thư mục hiện tại và “..” - biểu diễn thư mục mức trên. VD trong Linux: Giả sử ta đang ở trong thư mục /usr, khi đó để chỉ tới file mailbox ta chỉ cần sử dụng đường dẫn tương đối ast/mailbox là đủ. Nếu ta đang ở trong /usr/etc, đường dẫn tương đối sẽ là ../ast/mailbox.

1. **Các cấu trúc dữ liệu dùng trong tổ chức bên trong thư mục là các cấu trúc nào?**

* **Danh sách:**
* Tổ chức thư mục dưới dạng danh sách các khoản mục. Danh sách này có thể là bảng, danh sách kết nối hay một câu trúc khác.
* Việc tìm kiếm khoản mục khi đó được thực hiện bằng cách duyệt lần lượt danh sách.
* Để thêm file mới vào thư mục, trước tiên ta phải duyệt cả thư mục để kiểm tra xem khoản mụcvới tên file như vậy đã có chưa. Khoản mục mới sau đó được thêm vào cuối danh sách hoặc một ô trong bảng nếu ô này chưa được sử dụng.
* Để mở file hoặc xóa file, ta cũng tiến hành tìm tuần tự khoản mục tương ứng.
* Việc tìm kiếm trong danh sách rất chậm do phải duyệt lần lượt. Việc cache thư mục trong bộ nhớ mà nhiều hệ điều hành thực hiện chỉ cho phép tiết kiệm thời gian đọc thư mục từ đĩa chứ không giảm được thời gian tìm kiếm trong thư mục.
* **Cây nhị phân:**
* Xây dựng thư mục bằng cấu trúc dữ liệu có hỗ trợ sắp xếp như cây nhị phân (B Tree) hoặc cây nhị phân cân bằng, giúp tăng tốc độ tìm kiếm.
* Khi thêm khoản mục mới vào thư mục, hệ thống sắp xếp khoản mục vào nhánh tương ứng của cây.
* Hệ thống file NTFS của Windows NT là một ví dụ sử dụng thư mục kiểu này.
* **Bảng băm:**
* Cho phép dùng hàm băm để tính vị trí của khoản mục trong thư mục theo tên file. Thời gian tìm kiếm trong thư mục do đó giảm xuống rất nhiều so với dùng danh sách như ở phần trên.
* Việc tạo và xoá file cũng được thực hiện đơn giản bằng cách tính vị trí của khoản mục cần tạo hay xoá trong bảng băm.
* Nhược điểm lớn nhất của cấu trúc này là hàm băm phụ thuộc vào kích thước của bảng băm do đó bảng phải có kích thước cố định. Nếu số lượng khoản mục vượt quá kích thước chọn cho bảng băm thì phải chọn lại hàm băm. Ví dụ, ta có bảng băm với 128 khoản mục. Hàm băm biến đổi tên file thành các số nguyên từ 0 đến 127 bằng cách lấy phần dư sau khi chia cho 128. Nếu số lượng file trong thư mục tăng lên quá 128, chẳng hạn 130 file, kích thước bảng băm cũng phải tăng lên, chẳng hạn thành 150 khoản mục. Khi đó ta cần xây dựng hàm băm mới cho phép ánh xạ tên file sang khoảng mới từ 0 đến 149.
* **Tổ chức thư mục của DOS (FAT):**
* Mỗi đĩa lôgic trong MS-DOS có cây thư mục riêng của mình. Cây thư mục bắt đầu từ thư mục gốc ROOT.
* Ảnh có chứa văn bản, hàng, Phông chữ, biểu đồ

  Mô tả được tạo tự độngThư mục gốc được đặt ở phần đầu của đĩa, ngay sau sector khởi động BOOT và bảng FAT.
* Thư mục gốc chứa các file và thư mục con. Thư mục con có thể chứa file và thư mục mức dưới.
* Thư mục của MS-DOS được tổ chức dưới dạng bảng. Mỗi khoản mục chiếm một dòng trong bảng này và có độ dài cố định là 32 byte.
* **Tổ chức thư mục của Linux:**
* Ảnh có chứa văn bản, hàng, biểu đồ, ảnh chụp màn hình

  Mô tả được tạo tự độngThư mục hệ thống file Ext2 của Linux có cách tổ chức bên trong đơn giản.
* Mỗi khoản mục chứa tên file và địa chỉ i-node. Toàn bộ thông tin còn lại về thuộc tính và vị trí các khối dữ liệu của file được lưu trong i-node chứ không phải trong thư mục.
* Kích thước khoản mục không cố định mà phụ thuộc vào độ dài tên file. Do vậy, phần đầu mỗi khoản mục có một trường cho biết kích thước của khoản mục này.

1. **Trình bày các phương pháp cấp phát không gian cho file: Ưu/ Nhược điểm và nêu VD, so sánh điểm giống và khác.**

* **Cấp phát các khối liên tiếp:**
* **Ảnh có chứa hàng, biểu đồ, văn bản, Sơ đồ

  Mô tả được tạo tự động**File được cấp phát các khối liên tiếp trên đĩa.
* HĐH chọn 1 vùng trống có đủ số lượng khối cho file. Vị trí file trên đĩa được xác định bởi vị trí khối đầu tiên và độ dài hoặc số khối mà file đó chiếm. Khoản mục của file trong thư mục sẽ chứa địa chỉ khối đầu tiên và số khối mà file chiếm.
* HĐH cấp phát trước và biết kích thước file khi tạo file.
* Ưu điểm:

+ Cho phép truy cập trực tiếp và truy cập tuần tự.

+ Truy cập file đơn giản, tốc độ cao do tiết kiệm thời gian di chuyển đầu từ khi đọc các khối.

* Nhược điểm:

+ Khó tìm ra khoảng không gian trống đủ lớn trên đĩa để cấp phát cho file. Khi có yêu cầu cấp phát, hệ điều hành phải dành thời gian kiểm tra các vùng trống để tìm ra vùng có kích thước thích hợp.

+ Gây ra lãng phí không gian đĩa do hiện tượng phân mảnh ngoài (hiện tượng các vùng trống còn lại trên đĩa có kích thước quá nhỏ và do vậy không thể cấp phát cho file có kích thước lớn hơn)

+ Phải biết kích thước file khi tạo file.

* **Sử dụng danh sách kết nối:**
* **Ảnh có chứa văn bản, biểu đồ, Phông chữ, ảnh chụp màn hình

  Mô tả được tạo tự động**File được phân phối các khối nhớ kết nối với nhau thành danh sách kết nối, đầu khối chứa con trỏ trỏ tới khối tiếp theo.
* Mỗi khi file được cấp thêm khối mới, khối vừa cấp được thêm vào cuối danh sách.
* Các khối thuộc về một file có thể nằm bất cứ chỗ nào trên đĩa chứ không nhất thiết nằm liền kề nhau.
* Để xác định vị trí file trên đĩa, khoản mục của thư mục sẽ chứa con trỏ tới khối đầu tiên của file. Để truy cập file, hệ điều hành đọc lần lượt từng khối và sử dụng con trỏ để xác định khối tiếp theo.
* Ưu điểm:

+ Không bị phân mảnh ngoài do các khối không nhất thiết nằm cạnh nhau.

+ Không yêu cầu biết trước kích thước khi tạo file.

+ Dễ tìm vị trí cho file, khoản mục đơn giản.

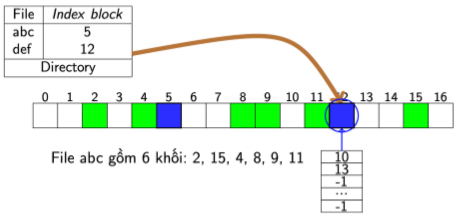
* Nhược điểm:

+ Chỉ hỗ trợ truy cập tuần tự mà không cho phép truy cập file trực tiếp. Để đọc một khối nào đó ta phải theo tất cả các con trỏ từ khối đầu tiên cho tới khối cần đọc.

+ Do các khối thuộc về một file có thể nằm rải rác ở nhiều nơi trên đĩa nên đầu từ của đĩa phải thực hiện nhiều thao tác di chuyển mới truy cập được hết dữ liệu của file. Tốc độ truy cập file không cao.

+ Việc liên kết các khối thuộc về một file bằng con trỏ cũng làm giảm độ tin cậy và tính toàn vẹn của hệ thống file. Trong trường hợp giá trị các con trỏ bị thay đổi không đúng do lỗi, việc xác định khối nào thuộc file nào sẽ không chính xác.

* **Sử dụng danh sách kết nối trên bảng chỉ số:**
* Bảng chỉ số là một bảng trong đó mỗi ô ứng với một khối của đĩa. Con trỏ tới khối tiếp theo của file không chứa ngay trong khối nữa mà được chứa trong ô tương ứng của bảng này.
* **Ảnh có chứa hàng, biểu đồ, văn bản, Phông chữ

  Mô tả được tạo tự động**Mỗi đĩa lôgic có một bảng chỉ số được lưu ở vị trí xác định, thường ở phần đầu đĩa.
* Kích thước mỗi ô trong bảng phụ thuộc vào số lượng khối trên đĩa.
* Việc sử dụng bảng chỉ số cho phép tiếp hành truy cập file trực tiếp (và truy cập tuần tự). Thay vì đọc tất cả các khối nằm trước khối cần truy cập, ta chỉ cần đi theo chuỗi con trỏ chứa trong bảng chỉ mục.
* Một ví dụ bảng chỉ số được sử dụng rất thành công là bảng FAT (file allocation table) của hệ điều hành MS-DOS, OS/2 và Windows (trừ các bản Windows sử dụng NTFS). Bảng FAT và các bản sao được lưu trữ ở đầu mỗi đĩa lôgic sau sector khởi động (BOOT). Tuỳ vào phiên bản FAT12, FAT16 hay FAT32, mỗi ô của bảng FAT có kích thước 12, 16 hay 32bit và cho phép quản lý tối đa 2^12, 2^16 hay 2^32 khối.
* **Sử dụng khối chỉ số:**
* Khối chỉ số (Index block hay Index node – I-node) là phương pháp trong đó tất cả con trỏ tới các khối thuộc về một file được tập trung một chỗ cho tiện việc truy cập trực tiếp đến từng khối của file.
* Trong phương pháp này, mỗi file có một mảng riêng của mình chứa trong một khối gọi là khối chỉ số (I-node). Mảng này chứa thuộc tính của file và vị trí các khối của file trên đĩa.
* Ô thứ i của mảng này chứa con trỏ tới khối thứ i của file.
* Vấn đề chọn kích thước cho i-node:

+ Nhỏ: tiết kiệm không gian nhưng không cho phép chứa đủ con trỏ tới các khối nếu file lớn.

+ Lớn: với file nhỏ chỉ chiếm 1 hoặc 2 ô thì các ô còn lại không được sử dụng sẽ gây lãng phí bộ nhớ.

* Giải pháp:

+ Cách 1: Cho phép thay đổi kích thước i-node bằng cách sử dụng danh sách kết nối.

+ Cách 2: Sử dụng i-node có cấu trúc nhiều mức.

* Ưu điểm:

+ Cho phép truy cập trực tiếp các khối trong file.

+ Các khối thuộc về một file không cần phải nằm gần nhau => không gây ra hiện tượng phân mảnh ngoài.

* Nhược điểm:

+ Giống dùng danh sách kết nối, đó là do các khối thuộc về 1 file không nằm gần nhau, tốc độ truy cập file bị giảm vì phải di chuyển đầu đọc nhiều lần.

1. **Các phương pháp quản lý không gian trống trên đĩa**

* **Bảng bit:**
* Bảng bit là một mảng một chiều. Mỗi ô của mảng có kích thước 1 bit và ứng với 1 khối nhớ trên đĩa. Khối đã được cấp phát có bit tương ứng là 0, khối chưa được cấp phát có bit tương ứng là 1 (hoặc ngược lại).
* Do nhiều bộ vi xử lý cung cấp các lệnh cho phép tìm ra các bit đầu tiên có giá trị 1 hoặc 0 trong một từ nhớ nên việc tìm ra vùng các khối trống có thể thực hiện rất nhanh ở mức phần cứng.
* Với những đĩa kích thước lớn, việc đọc toàn bộ vectơ bit vào bộ nhớ để tìm kiếm khối trống có thể đòi hỏi khá nhiều bộ nhớ.
* VD: Trong trường hợp sau, các khối 0, 1, 8, 9, 10 đã được cấp phát, các khối còn lại chưa bị sử dụng: 00111111000111111...
* **Danh sách kết nối:**
* Các khối trống được liên kết với nhau thành danh sách. Mỗi khối trống chứa địa chỉ của khối trống tiếp theo. Địa chỉ khối trống đầu tiên trong danh sách này được lưu trữ ở một vị trí đặc biệt trên đĩa và được hệ điều hành giữ trong bộ nhớ khi cần làm việc với các file.
* Đòi hỏi truy cập lần lượt các khối trống khi cần duyệt danh sách. Tuy nhiên, trong đa số trường hợp, hệ điều hành ít khi phải duyệt danh sách mà có thể cấp phát ngay các khối ở đầu danh sách. Việc duyệt chỉ cần thiết khi cần tìm các khối trống nằm kề nhau như trong phương pháp cấp phát khối liên tiếp.
* **Danh sách vùng trống:**
* Phương pháp này lợi dụng một đặc điểm là các khối nằm liền nhau thường được cấp phát và giải phóng đồng thời. Do đó, thay vì lưu trữ địa chỉ từng khối trống, hệ điều hành lưu vị trí khối trống đầu tiên của vùng các khối trống liên tiếp và số lượng các khối trống nằm liền ngay sau đó.
* Hai thông số này sau đó sẽ được lưu trong một danh sách riêng.

1. **Các yêu cầu và phương pháp để đảm bảo tính toàn vẹn của hệ thống file**

* **Yêu cầu phải đảm bảo tính toàn vẹn:**
* Hệ thống file chứa nhiều cấu trúc dữ liệu có mối liên kết. Nếu thông tin về các liên kết bị hư hại thì tính toàn vẹn của hệ thống bị phá vỡ.
* Một số trường hợp phá vỡ tính toàn vẹn:

+ Các khối không có mặt trong danh sách các khối trống, đồng thời cũng không có mặt trong một file nào.

+ Một khối có thể vừa thuộc 1 file nào đó vừa có mặt trong danh sách khối trống.

+ Một khối đồng thời thuộc về 2 file cùng 1 lúc.

+ File bị xoá trong khi khoản mục ứng với file trong thư mục vẫn còn hoặc trường hợp ngược lại.

* HDH có các chương trình kiểm tra tính toàn vẹn của hệ thống file, được chạy khi hệ thống khởi động, đặc biệt là sau sự cố (VD: chương trình SCANDISK của WIN hoặc DOS)
* Mặc dù có thể kiểm tra tính toàn vẹn của các liên kết trong hệ thống file. Tuy nhiên, kiểm tra toàn bộ hệ thống file đòi hỏi nhiều thời gian. Việc kiểm tra chỉ cho phép phát hiện lỗi sau khi đã xảy ra và không đảm bảo khôi phục dữ liệu đối với một số lỗi.
* **Phương pháp đảm bảo tính toàn vẹn:**
* Sử dụng 1 kỹ thuật dựa trên khái niệm giao tác (transaction). Giao tác là một tập hợp các thao tác cần phải được thực hiện trọn vẹn cùng với nhau.
* Với hệ thống file: mỗi giao tác sẽ bao gồm những thao tác thay đổi liên kết cần thực hiện cùng nhau (Ví dụ các thao tác cập nhật liên kết liên quan với một khối trên đĩa)
* Toàn bộ trạng thái hệ thống file, cụ thể là những thay đổi thực hiện với các liên kết và cấu trúc quản lý file, được hệ thống ghi lại trong một file log (một dạng file nhật ký ghi lại thông tin về hệ thống theo thời gian). Mỗi khi thực hiện giao tác, hệ thống kiểm tra xem giao tác có được thực hiện trọn vẹn không. Nếu giao tác không được thực hiện trọn vẹn, HDH sử dụng thông tin từ log để khôi phục hệ thống file về trạng thái không lỗi trước khi thực hiện giao tác.

1. **Cách kiểm soát truy cập file sử dụng mật khẩu và sử dụng danh sách quản lý truy cập.**

* **Sử dụng mật khẩu:**
* Mỗi file sẽ có một mật khẩu gắn với một số quyền nào đó. Khi người dùng hoặc chương trình ứng dụng truy cập file để đọc, ghi hoặc thực hiện thao tác khác, hệ thống sẽ yêu cầu cùng cấp mật khẩu tương ứng và việc truy cập chỉ được thực hiện nếu mật khẩu đúng.
* Nhược điểm chủ yếu của phương pháp này là việc nhớ mật khẩu cho từng file hoặc từng thư mục là vô cùng khó khăn nếu số lượng file lớn. Ngoài ra, do mỗi thao tác truy cập đều đòi hỏi cung cấp mật khẩu nên rất mất thời gian và không tiện lợi.
* **Sử dụng danh sách quản lý truy cập ACL (Access Control List):**
* Mỗi file sẽ được gắn một danh sách đi kèm gọi là danh sách quản lý truy cập ACL (Access Control List), chứa thông tin định danh người dùng và các quyền mà người dùng đó được thực hiện với file.
* Danh sách quản lý quyền truy cập thường được lưu trữ như một thuộc tính của file hoặc thư mục.
* Phương pháp sử dụng ACL thường được sử dụng cùng với cơ chế đăng nhập.
* Các quyền truy cập thường sử dụng với file bao gồm:

+ Quyền đọc (Read): người có quyền này được phép đọc nội dung file.

+ Quyền ghi, thay đổi (Write, Change): được phép ghi vào file, tức là thay đổi nội dung file.

+ Quyền xóa (Delete): được phép xóa file. Thực chất, quyền này tương đương với quyền thay đổi file.

+ Quyền thay đổi chủ file (Change owner).

1. **Các nội dung về hệ thống file FAT:**

* **Trình bày các bước để đọc bảng FAT từ thẻ nhớ USB (FAT16)**
* Bước 1: Xác định đường dẫn thẻ nhớ USB trên hệ điều hành. Mở ổ đĩa tương ứng vị trí thẻ nhớ USB (thường là ổ E:)
* Bước 2: Tìm và đọc bộ boot sector của thẻ nhớ USB bằng hàm đọc sector mức thấp absread.
* Bước 3: Sử dụng thông tin từ bộ boot sector (kích thước sector tính bằng byte, số lượng sector dành cho vùng đầu đĩa đến trước FAT, kích thước bảng FAT tính bằng sector), tiếp tục dùng hàm đọc sector mức thấp absread để đọc bảng FAT (File Allocation Table).
* **Trình bày các bước để đọc thư mục gốc từ thẻ nhớ USB vào bộ nhớ trong (hệ thống file là các thẻ nhớ FAT16)**
* Bước 1: Xác định đường dẫn thẻ nhớ USB trên hệ điều hành. Mở ổ đĩa tương ứng vị trí thẻ nhớ USB (thường là ổ E:)
* Bước 2: Tìm và đọc bộ boot sector của thẻ nhớ USB bằng hàm đọc sector mức thấp absread.
* Bước 3: Sử dụng thông tin từ bộ boot sector (số khoản mục tối đa trong thư mục gốc ROOT, kích thước sector tính bằng byte, số lượng sector dành cho vùng đầu đĩa đến trước FAT, kích thước bảng FAT tính bằng sector, số lượng bảng FAT), tiếp tục dùng hàm đọc sector mức thấp absread để đọc thư mục gốc ROOT.

# **Chương 3: Quản lý bộ nhớ**

1. **Địa chỉ logic và địa chỉ vật lý**

* **Địa chỉ logic**
* Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, biểu đồ

  Mô tả được tạo tự độngĐịa chỉ lôgic là địa chỉ được gán cho các lệnh và dữ liệu không phụ thuộc vào vị trí cụ thể của tiến trình trong bộ nhớ.
* Khi thực hiện chương trình, CPU “nhìn thấy” và sử dụng địa chỉ lôgic này để trỏ đến các phần khác nhau của lệnh, dữ liệu.
* Toàn bộ địa chỉ được gán trong chương trình tạo thành không gian nhớ lôgic của chương trình. Trong trường hợp sử dụng bộ nhớ ảo, địa chỉ lôgic còn được gọi là địa chỉ ảo.
* **Địa chỉ vật lý**
* Địa chỉ vật lý là địa chỉ chính xác trong bộ nhớ của máy tính và được phần cứng quản lý bộ nhớ đặt lên đường địa chỉ để truy cập ô nhớ tương ứng. Địa chỉ vật lý còn được gọi là địa chỉ tuyệt đối.
* Để truy cập bộ nhớ, địa chỉ logic cần được biến đổi thành địa chỉ vật lý. Thông thường, không gian nhớ vật lý khác với không gian nhớ lôgic của chương trình. Trong thời gian thực hiện tiến trình, địa chỉ logic được ánh xạ sang địa chỉ vật lý nhờ một cơ chế phần cứng gọi là khối quản lý bộ nhớ (MMU -Memory Management Unit).

1. **Kĩ thuật “Tải trong quá trình chạy” và “Liên kết động và thư viện dùng chung” trong tổ chức chương trình. Phân tích ưu điểm từng kỹ thuật**

* **Tải trong quá trình chạy**
* Bình thường, toàn bộ chương trình được tải vào bộ nhớ để thực hiện. Đối với các chương trình lớn, trong một phiên làm việc, một số phần của chương trình có thể không được dùng tới, chẳng hạn các hàm xử lý lỗi. Các hàm này sẽ chiếm chỗ vô ích trong bộ nhớ, đồng thời làm tăng thời gian tải chương trình lúc đầu.
* Giải pháp:

+ Chương trình chính được tải vào bộ nhớ và chạy.

+ Hàm/ chương trình con nào chưa được gọi đến thì chưa được tải vào bộ nhớ.

+ Mỗi khi có một lời gọi hàm, chương trình gọi sẽ kiểm tra xem hàm được gọi đã nằm trong bộ nhớ chưa. Nếu chưa, chương trình tải sẽ được gọi để tải hàm vào bộ nhớ, ánh xạ địa chỉ hàm vào không gian nhớ chung của chương trình và thay đổi bảng địa chỉ ghi lại các ánh xạ đó.

+ Trong kỹ thuật này, việc kiểm tra và tải các hàm do chương trình người dùng đảm nhiệm. Hệ điều hành chỉ cung cấp các hàm phục vụ việc tải các mô đun.

* Ưu điểm:

+ Tiết kiệm và tối ưu hoá bộ nhớ.

+ Tăng tốc độ khởi động chương trình: Chỉ chương trình chính được tải vào bộ nhớ.

+ Tăng độ linh hoạt: Chỉ những hàm/ chương trình con được dùng mới phải tải vào bộ nhớ, giúp tối ưu hoá hiệu suất chương trình.

* **Liên kết động và thư viện dùng chung**
* Trong quá trình liên kết tĩnh, các hàm thư viện được liên kết luôn vào chương trình chính. Kích thước chương trình khi đó sẽ bằng kích thước chương trình vừa được dịch cộng với kích thước các hàm thư viện. Trên thực tế, có các hàm thư viện được dùng trong hầu hết các chương trình, nếu liên kết tĩnh, các hàm này sẽ có mặt lặp đi lặp lại trong các chương trình làm tăng không gian chung mà các chương trình chiếm, bao gồm cả không gian trên đĩa và không gian bộ nhớ chính khi các chương trình được tải vào để thực hiện.
* Giải pháp:

+ Sử dụng kỹ thuật liên kết động và các thư viện hàm dùng chung. Về bản chất, kỹ thuật này chỉ thực hiện liên kết thư viện vào chương trình trong thời gian thực hiện, khi chương trình đã ở trong bộ nhớ.

+ Trong giai đoạn liên kết, không kết nối các hàm thư viện vào chương trình mà chỉ chèn các thông tin về hàm thư viện đó.

+ Trong thời gian chạy, khi đoạn mã chèn vào được thực hiện, đoạn này sẽ kiểm tra xem mô đun thư viện đã có nằm trong bộ nhớ chưa. Nếu chưa, mô đun thư viện sẽ được đọc vào bộ nhớ, sau đó chương trình sẽ thay địa chỉ đoạn mã chèn thành địa chỉ mô đun thư viện. Lần tiếp theo cần sử dụng, modul thư viện sẽ được chạy trực tiếp.

+ Đối với mô đun thư viện được sử dụng bởi nhiều tiến trình, tất cả tiến trình có thể dùng chung một bản duy nhất phần mã chương trình của thư viện trong bộ nhớ. Thư viện khi đó được gọi là thư viện dùng chung.

* Ưu điểm:

+ Tiết kiệm và tối ưu hoá bộ nhớ.

+ Tăng tốc độ khởi động chương trình.

+ Tăng độ linh hoạt.

+ Sử dụng thư viện dùng chung giúp việc cập nhật và sửa lỗi thư viện dễ dàng hơn.

1. **Trình bày kĩ thuật “Phân chương cố định”**

* Bộ nhớ được phân thành những chương có kích thước cố định ở những vị trí cố định. Mỗi chương chứa được đúng một tiến trình. Khi được tải vào, tiến trình được cấp phát một chương. Sau khi tiến trình kết thúc, hệ điều hành giải phóng chương và chương có thể được cấp phát tiếp cho tiến trình mới.
* Lựa chọn kích thước các chương:

+ Bằng nhau: Đơn giản nhưng không mềm dẻo. Tiến trình có kích thước lớn hơn kích thước chương sẽ không thể tải vào chương và chạy được. Muốn cho chương chứa được các tiến trình lớn, ta phải tăng kích thước của chương bằng kích thước của tiến trình lớn nhất. Do mỗi tiến trình chiếm cả một chương, các tiến trình nhỏ cũng được cung cấp và chiếm cả chương như một tiến trình lớn. Phần bộ nhớ rất đáng kể còn lại của chương sẽ bị bỏ trống gây lãng phí bộ nhớ, gây ra phân mảnh trong.

+ Khác nhau: Có hai cách lựa chọn chương nhớ để cấp cho tiến trình đang chờ đợi.

Cách thứ nhất là lựa chọn chương nhỏ nhất có thể chứa tiến trình, tạm gọi là lựa chọn chương phù hợp nhất, để cấp. Mỗi chương khi đó có một hàng đợi riêng. Tiến trình có kích thước phù hợp với chương nào sẽ nằm trong hàng đợi của chương đó. Ưu điểm của cách cấp chương này là cho phép giảm tối thiểu phân mảnh trong và do đó tiết kiệm được bộ nhớ. Tuy nhiên nhược điểm là do mỗi chương có một hàng đợi riêng nên sẽ có thời điểm hàng đợi của chương lớn hơn thì rỗng, trong khi hàng đợi của chương nhỏ hơn thì có nhiều tiến trình. Các tiến trình nhỏ này buộc phải đợi được cấp chương nhỏ trong khi có thể tải vào chương lớn hơn và chạy.

Cách thứ hai cho phép khắc phục nhược điểm nói trên. Trong cách này, hệ điều hành sử dụng một hàng đợi duy nhất cho tất cả các chương. Mỗi khi có khi có một chương trống, tiến trình nằm gần đầu hàng đợi nhất và có kích thước phù hợp với chương sẽ được tải vào để thực hiện.

* Ưu điểm:

+ Đơn giản và ít xử lý.

+ Giảm thời gian tìm kiếm.

* Nhược điểm:

+ Số lượng tiến trình trong bộ nhớ bị hạn chế bởi số lượng chương.

+ Kích thước chương trình có thể lớn hơn kích thước chương lớn nhất.

+ Tổng bộ nhớ tự do còn lớn nhưng không dùng để nạp các chương trình khác.

1. **Trình bày kĩ thuật “Phân chương động”. Phân tích ưu/ nhược điểm so với “Phân chương cố định”. Lấy VD minh hoạ cho các chiến lược cấp phát chương động mà hệ điều hành thường sử dụng (First Fit, Best Fit và Worst Fit)**

* **Phân chương động**
* Kích thước và số lượng chương đều có thể thay đổi.
* Mỗi khi tiến trình được tải vào bộ nhớ, tiến trình được cấp một lượng bộ nhớ đúng bằng lượng bộ nhớ mà tiến trình cần, sau khi kết thúc, tiến trình giải phóng bộ nhớ. Vùng bộ nhớ do tiến trình chiếm trước đó trở thành một “lỗ” (vùng trống) trong bộ nhớ nếu các vùng nhớ trước và sau thuộc về các tiến trình khác. Như vậy, ở mỗi thời điểm, trong bộ nhớ tồn tại một tập hợp các vùng trống có kích thước khác nhau. Hệ điều hành sử dụng một bảng để biết được phần bộ nhớ nào đã được dùng, phần nào đang còn trống. Các vùng bộ nhớ cũng có thể được liên kết thành một danh sách kết nối. Tiến trình cần bộ nhớ được xếp trong hàng đợi để chờ tới lượt mình. Mỗi khi đến lượt một tiến trình, hệ điều hành sẽ cố gắng cung cấp bộ nhớ cho tiến trình đó bằng cách tìm một lỗ (vùng bộ nhớ) trống có kích thước lớn hơn hoặc bằng kích thước tiến trình.

+ Nếu không có vùng bộ nhớ trống nào thoả mãn điều kiện này, hệ điều hành có thể chờ cho tới khi một vùng bộ nhớ đủ lớn được giải phóng để cấp phát hoặc tìm kiếm trong hàng đợi một tiến trình đủ nhỏ để có thể chứa trong những vùng bộ nhớ còn trống.

+ Trong trường hợp kích thước vùng trống tìm được lớn hơn kích thước tiến trình, vùng trống được chia thành hai phần. Một phần cấp cho tiến trình, phần còn lại trở thành một vùng trống có kích thước nhỏ hơn vùng trống ban đầu và được bổ sung vào danh sách các vùng trống mà hệ điều hành quản lý.

* Mỗi tiến trình sau khi kết thúc tạo ra một vùng trống mới. Nếu vùng trống này nằm kề cận với một vùng trống khác, chúng sẽ được nối với nhau để tạo ra vùng trống mới có kích thước lớn hơn.
* Việc phân chương động có thể sinh ra trong bộ nhớ các vùng trống kích thước quá nhỏ và do vậy không thể cấp phát tiếp cho bất kỳ tiến trình nào. Không gian mà các vùng trống này chiếm do vậy bị bỏ phí. Hiện tượng này gọi là phân mảnh ngoài.
* Để giải quyết vấn đề phân mảnh ngoài, người ta sử dụng kỹ thuật dồn bộ nhớ. Các chương thuộc về các tiến trình được dịch chuyển lại nằm kề nhau. Các vùng trống giữa các tiến trình khi đó sẽ dồn thành một vùng trống duy nhất. Kỹ thuật dồn bộ nhớ đòi hỏi một số thời gian nhất định để di chuyển các chương nhớ và làm nảy sinh vấn đề bố trí lại địa chỉ trong các tiến trình. Do các hạn chế này, việc dồn bộ nhớ không thể thực hiện quá thường xuyên.
* Các chiến lược lựa chọn vùng trống để cấp phát thường dùng:

+ Vùng thích hợp đầu tiên (first fit): chọn vùng trống đầu tiên có kích thước lớn hơn hoặc bằng kích thước cần cấp phát. Chiến lược này đơn giản và do đó thực hiện nhanh nhất.

+ Vùng thích hợp nhất (best fit): chọn vùng trống nhỏ nhất trong số các vùng trống có kích thước lớn hơn hoặc bằng kích thước cần cấp phát.

+ Vùng không thích hợp nhất (worst fit): từ nhận xét là các vùng trống sinh ra sau khi cấp phát theo chiến lược thứ hai (best fit) có kích thước bé và do đó thường không thích hợp cho việc cấp phát tiếp theo, người ta nghĩ ra chiến lược thứ ba này. Vùng trống lớn nhất sẽ được cấp phát. Không gian còn thừa từ vùng trống này sau khi cấp xong tạo ra vùng trống mới có kích thước lớn hơn so với hai chiến lược trên.

* **VD minh hoạ cho các chiến lược cấp phát động**

Trong bộ nhớ có 4 vùng trống có kích thước lần lượt là 3MB, 8MB, 7MB, và 10MB; yêu cầu cấp phát vùng nhớ kích thước 6MB. Ba chiến lược cấp phát ở trên sẽ cho kết quả như sau:

* Chiến lược first fit sẽ chọn khối 8MB để chia và cấp phát.
* Chiến lược best fit sẽ chọn vùng trống 7MB.
* Chiến lược worst fit sẽ chọn vùng trống 10MB.

Qua nhiều nghiên cứu và kết quả cho thấy: hai phương pháp đầu cho phép giảm phân mảnh ngoài tốt hơn phương pháp thứ ba. Trong hai phương pháp đầu, phương pháp thứ nhất đơn giản và có tốc độ nhanh nhất.

1. **Cơ chế ánh xạ địa chỉ khi sử dụng kĩ thuật phân chương bộ nhớ**

* Ảnh có chứa văn bản, biểu đồ, hàng, ảnh chụp màn hình

  Mô tả được tạo tự độngKhi hệ điều hành tải tiến trình vào và thực hiện, hai thanh ghi đặc biệt của CPU sẽ được sử dụng:

+ Thanh ghi cơ sở chứa địa chỉ bắt đầu của tiến trình trong bộ nhớ.

+ Thanh ghi giới hạn chứa giới hạn địa chỉ lô gic của tiến trình tức độ dài chương chứa tiến trình.

* Địa chỉ lôgic được so sánh với nội dung thanh ghi giới hạn. Chỉ những địa chỉ lôgic nhỏ hơn giá trị thanh ghi này mới được coi là hợp lệ và được ánh xạ thành địa chỉ vật lý. Địa chỉ vật lý được tạo ra bằng cách cộng nội dung thanh ghi cơ sở với địa chỉ lôgic.
* Trong trường hợp các chương bị di chuyển trong bộ nhớ, chẳng hạn như khi hệ điều hành tiến hành dồn bộ nhớ để tránh phân mảnh ngoài, nội dung thanh ghi cơ sở sẽ được thay đổi thành địa chỉ mới của chương. Các phép ánh xạ sau đó vẫn diễn ra như cũ.

1. **Kỹ thuật giúp tăng tốc độ truy cập bảng trang và bảng trang nhiều mức (đọc thêm).**

* **Một số kỹ thuật liên quan tới việc tổ chức và truy cập bảng trang**
* Lưu bảng trang trong các thanh ghi: Sử dụng một tập hợp các thanh ghi được thiết kế riêng cho việc chứa dữ liệu bảng trang và ánh xạ địa chỉ để lưu bảng trang. Tốc độ truy cập bảng phân trang khi đó rất cao nhưng kích thước và số lượng bảng phân trang sẽ bị hạn chế do số lượng thanh ghi của CPU không nhiều lắm.
* Lưu bảng trang trong bộ nhớ trong: Vị trí của mỗi bảng sẽ được trỏ tới bởi một thanh ghi gọi là thanh ghi cơ sở của bảng trang (PTBR – Page Table Base Register). Như vậy mỗi thao tác truy cập bộ nhớ trong đòi hỏi hai thao tác truy cập bộ nhớ, một để đọc ô nhớ tương ứng trong bảng trang và một để thực hiện truy cập cần thiết. Do tốc độ bộ nhớ tương đối chậm nên đòi hỏi nhiều thời gian cho việc truy cập bảng => Sử dụng bộ nhớ cache tốc độ cao.
* **Bảng trang nhiều mức**
* Các hệ thống tính toán hiện đại cho phép sử dụng không gian địa chỉ lôgic rất lớn (2^32 đến 2^64). Kích thước bảng phân trang do đó cũng tăng theo.
* Do bảng trang có kích thước lớn, cần chia bảng trang thành các phần nhỏ hơn sao cho các phần có thể được lưu trong bộ nhớ độc lập với nhau.
* Việc chia nhỏ được thực hiện bằng cách tổ chức bảng phân trang nhiều mức. Mỗi khoản mục của bảng mức trên không chỉ tới một trang mà chỉ tới một bảng phân trang khác.

1. **Khái niệm phân đoạn bộ nhớ và ưu/ nhược điểm của phương pháp phân đoạn bộ nhớ (so sánh với phân chương (động) bộ nhớ ?).**

* **Khái niệm:**
* Là cách tổ chức cho phép chương trình được chia thành những phần kích thước khác nhau gọi là đoạn (segment) tuỳ theo ý nghĩa của chúng. Chẳng hạn, ta có thể phân biệt:

+ Đoạn chương trình: chứa mã toàn bộ chương trình, hay một số hàm hoặc thủ tục của chương trình.

+ Đoạn dữ liệu: chứa các biến toàn cục, các mảng.

+ Đoạn ngăn xếp: chứa ngăn xếp của tiến trình trong quá trình thực hiện.

* Mỗi đoạn chiếm một vùng liên tục có vị trí bắt đầu và kích thước và có thể nằm tại bất cứ đâu trong bộ nhớ. Đối tượng, phần tử trong từng đoạn được xác định bởi vị trí tương đối so với đầu đoạn
* Không gian địa chỉ lôgic của tiến trình khi đó sẽ gồm tập hợp các đoạn. Mỗi đoạn tương ứng với không gian địa chỉ riêng, được phân biệt bởi tên và độ dài của mình. Ngoài cách dùng tên, đoạn cũng có thể được đánh số để phân biệt. Mỗi địa chỉ lôgic do CPU sinh ra khi đó sẽ gồm hai phần: số thứ tự của đoạn và độ dịch trong đoạn.
* **Ưu điểm:**
* Tránh hiện tượng phân mảnh trong, dễ sắp xếp bộ nhớ.
* Dễ chia sẻ các đoạn giữa các tiến trình khác nhau.
* Kích thước mỗi đoạn có thể thay đổi không ảnh hưởng đoạn khác
* **Nhược điểm:**
* Gây phân mảnh ngoài.
* **So sánh với phân chương động:**
* Giống:

+ Bộ nhớ được cấp phát theo từng vùng kích thước thay đổi.

+ Đều gây phân mảnh ngoài.

* Khác:

+ Mỗi chương trình có thể chiếm nhiều hơn 1 đoạn bộ nhớ và không cần nằm liền kề nhau.

1. **Cơ chế ánh xạ địa chỉ khi sử dụng kĩ thuật phân đoạn bộ nhớ**

* Địa chỉ: Khi sử dụng phân đoạn, không gian nhớ lô gic sẽ bao gồm không gian nhớ của các đoạn. Do vậy, địa chỉ lô gic bao gồm hai thành phần, chỉ rõ hai thông tin: địa chỉ thuộc đoạn nào, và độ dịch từ đầu đoạn là bao nhiêu, có cấu trúc như sau: < số thứ tự đoạn (s), độ dịch trong đoạn (o) >
* Ánh xạ địa chỉ: Sử dụng bảng đoạn (segment table) cho mỗi tiến trình. Mỗi phần tử của bảng tương ứng 1 đoạn, chứa:

+ Dấu hiệu (0/1): Đoạn đã tồn tại trong bộ nhớ.

+ Địa chỉ cơ sở: vị trí bắt đầu của đoạn trong bộ nhớ máy tính.

+ Giới hạn đoạn: là độ dài đoạn và sẽ được sử dụng để chống truy cập trái phép ra ngoài đoạn.

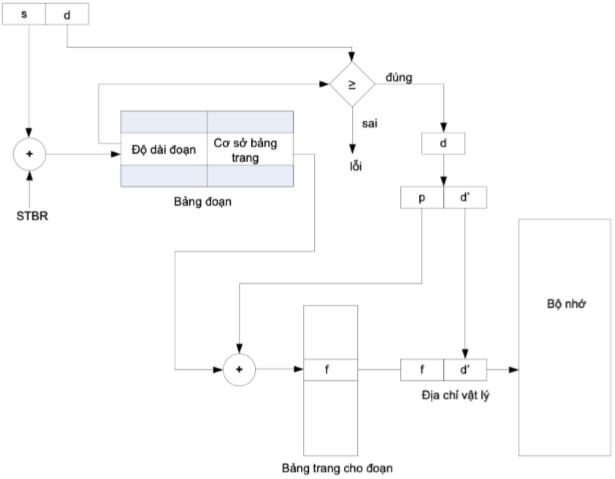
* Bảng đoạn được sử dụng kết hợp với một cơ chế phần cứng cho phép biến đổi từ địa chỉ lôgic sang địa chỉ tuyệt đối:

+ Bước 1: Phần s trong địa chỉ được sử dụng để tìm tới ô thứ s trong bảng đoạn và truy cập hai giá trị giới hạn và cơ sở chứa trong ô này.

+ Bước 2: Phần độ dịch o của địa chỉ được so sánh với phần giới hạn chứa trong ô. Nếu o giới hạn thì địa chỉ này vượt ra ngoài phạm vi của đoạn => lỗi truy cập.

+ Bước 3: Trong trường hợp truy cập hợp lệ, phần độ dịch o được cộng với địa chỉ cơ sở để tạo ra địa chỉ vật lý.

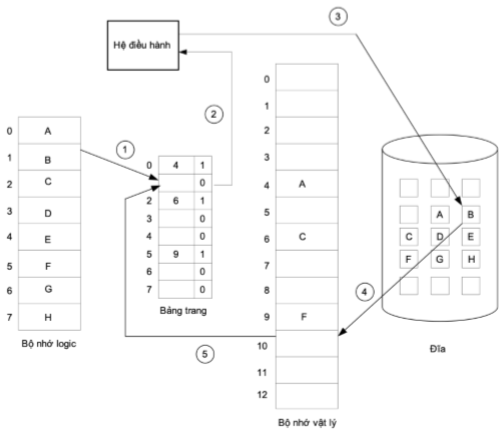
1. **Trình bày phương pháp kết hợp phân trang và phân đoạn. Vẽ sơ đồ và giải thích cơ chế ánh xạ địa chỉ.**

* Tiến trình bao gồm nhiều đoạn. Mỗi đoạn lại được phân trang. Như vậy mỗi tiến trình có một bảng phân đoạn riêng và mỗi đoạn lại có bảng phân trang riêng của mình.
* Để định vị ô nhớ trong phương pháp này, địa chỉ phải gồm ba phần (s,p,o):

+ Phần thứ nhất s là số thứ tự đoạn. s cho phép xác định khoản mục ứng với đoạn trong bảng chia đoạn. Khoản mục chứa con trỏ tới bảng chia trang cho đoạn đó.

+ Số thứ tự trang p cho phép xác định khung trang tương ứng.

+ Độ dịch o sẽ được cộng vào địa chỉ khung để tính ra địa chỉ vật lý.

1. **Trình bày kỹ thuật “Nạp trang theo nhu cầu” cho bộ nhớ ảo. Cho VD minh hoạ quá trình thực hiện thiếu trang?**

* Nạp trang theo nhu cầu dựa trên phân trang kết hợp trao đổi bộ nhớ đĩa. Tiến trình được phân trang và các trang chứa trên đĩa.
* Khi cần thực hiện tiến trình, ta nạp tiến trình vào bộ nhớ. Tuy nhiên, không phải toàn bộ các trang của tiến trình được nạp cùng một lúc. Chỉ những trang đang cần đến mới được nạp vào.
* Tại một thời điểm, mỗi tiến trình sẽ gồm có tập hợp các trang đang ở trong bộ nhớ và những trang còn trên đĩa. Để nhận biết trang đã ở trong bộ nhớ hay ở ngoài, người ta thêm một bit (tạm gọi là bit P) vào khoản mục trong bảng phân trang. Giá trị của bit bằng 1(0) cho thấy trang tương ứng đã ở trong (ngoài) bộ nhớ hoặc ngược lại. Khi tiến trình truy cập tới một trang, bit P của trang sẽ được kiểm tra. Nếu trang đã ở trong bộ nhớ, việc truy cập diễn ra bình thường. Ngược lại, nếu trang chưa được nạp vào, một sự kiện gọi là thiếu trang hay lỗi trang (page-fault) sẽ xảy ra. Phần cứng làm nhiệm vụ ánh xạ địa chỉ trang sinh ra một ngắt và được chuyển cho hệ điều hành để xử lý.
* Thủ tục xử lý ngắt sinh ra do thiếu trang gồm các bước sau:

+ Hệ điều hành tìm một khung trống trong danh sách các khung trống.

+ Trang bị thiếu tương ứng sẽ được đọc từ đĩa vào khung trang vừa tìm được.

+ Sau khi trang được đọc vào, khoản mục tương ứng trong bảng phân trang sẽ được sửa đổi tương ứng với vị trí của trang trong bộ nhớ, trạng thái bit P thể hiện việc trang đã ở trong bộ nhớ.

+ Khôi phục lại trạng thái của tiến trình và thực hiện tiếp lệnh bị ngắt.

* Chiến lược nạp trang hoàn toàn theo nhu cầu:

+ Bắt đầu một tiến trình mà không nạp bất kỳ trang nào của tiến trình vào bộ nhớ.

+ Khi con trỏ lệnh được hệ điều hành chuyển tới lệnh đầu tiên (chưa được tải vào bộ nhớ) của tiến trình để thực hiện, sự kiện thiếu trang sẽ sinh ra và trang tương ứng được nạp vào.

+ Tiến trình sau đó thực hiện bình thường cho tới lần thiếu trang tiếp theo.

* Chiến lược nạp trang trước: các trang chưa cần đến cũng được nạp vào bộ nhớ. Các nghiên cứu cho thấy, chiến lược nạp trang trước là không hiệu quả về nhiều mặt.

1. **Phân tích rõ cùng 1 lệnh có thể xảy ra nhiều sự kiện lỗi trang không ? (đọc thêm)**

Có. Nếu một lệnh yêu cầu truy cập đến nhiều trang, và một trong các trang đó không có trong bộ nhớ, thì sẽ xảy ra một sự kiện lỗi trang. Lệnh sẽ bị ngắt và hệ điều hành sẽ tiến hành nạp trang đó vào bộ nhớ. Sau khi trang được nạp vào, lệnh sẽ tiếp tục thực hiện và có thể tiếp tục yêu cầu truy cập đến các trang khác. Nếu một trong các trang đó vẫn không có trong bộ nhớ, thì sẽ xảy ra một sự kiện lỗi trang khác.

1. **Lý do phải “Đổi trang” và các bước tiến hành quá trình “Đổi trang”.**

* **Lý do phải đổi trang**
* Khi xảy ra thiếu trang, hệ điều hành tìm một khung trống trong bộ nhớ, đọc trang thiếu vào khung và tiến trình sau đó hoạt động bình thường.
* Tuy nhiên, do kích thước của các tiến trình có thể lớn hơn kích thước bộ nhớ thực rất nhiều nên tới một lúc nào đó sẽ xảy ra tình trạng toàn bộ bộ nhớ đã được cấp phát, HĐH không thể tìm được khung trống để tải trang mới vào.
* Giải pháp đưa ra:

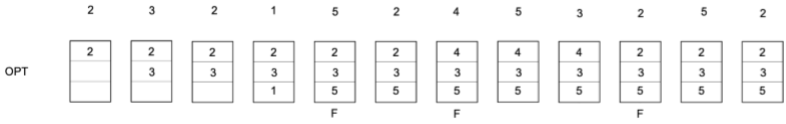
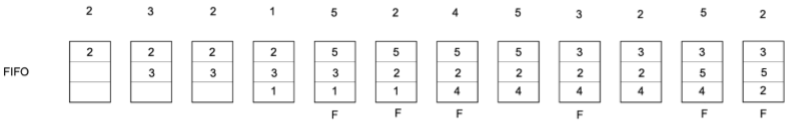
+ Cách 1: HĐH kết thúc tiến trình do không thoả mãn được nhu cầu bộ nhớ. Nhưng bộ nhớ ảo cho phép các tiến trình sử dụng được không gian nhớ lớn hơn không gian nhớ thực và tăng tính đa chương trình của hệ thống. Tiến trình và người dùng cần được đáp ứng nhu cầu về bộ nhớ.

+ Cách 2: Tạm trao đổi tiến trình ra đĩa, giải phóng toàn bộ không gian mà tiến trình chiếm trong bộ nhớ và chờ tới khi thuận lợi (nhiều bộ nhớ trống hơn) mới nạp lại tiến trình vào bộ nhớ để thực hiện tiếp.

+ Cách 3: Sử dụng kỹ thuật đổi trang, sử dụng trong đa số trường hợp.

* **Quá trình đổi trang**
* Bước 1: Xác định trang trên đĩa cần nạp vào bộ nhớ
* Bước 2: Nếu có khung trống thì chuyển sang bước 4
* Bước 3: Lựa chọn 1 khung trên bộ nhớ theo 1 thuật toán nào đó để giải phóng, ghi nội dung khung bị đổi ra đĩa (nếu cần), cập nhật bảng trang và bảng khung
* Bước 4: Đọc trang cần nạp vào khung vừa giải phóng; cập nhật bảng trang và bảng khung
* Bước 5: Thực hiện tiếp tiến trình từ điểm bị dừng trước khi đổi trang

1. **Trình bày các chiến lược đổi trang “Đổi trang tối ưu” (OPT/MIN) và “Vào trước ra trước” (FIFO).**

* **Đổi trang tối ưu**
* HĐH chọn trang nhớ sẽ không được dùng tới trong khoảng thời gian lâu nhất trong tương lai để đổi hay trang có lần sử dụng tiếp theo xa nhất.
* Cho phép giảm tối thiểu sự kiện thiếu trang và tối ưu theo tiêu chuẩn này.
* Tuy nhiên, để sử dụng chiến lược này, hệ điều hành cần đoán trước nhu cầu sử dụng các trang trong tương lai. Điều này rất khó thực hiện trên thực tế do thứ tự truy cập trang không cố định và không biết trước. Chiến lược đổi trang tối ưu, do đó, không thể áp dụng trong thực tế mà chỉ được dùng để so sánh với các chiến lược đổi trang khác.
* **Vào trước ra trước**
* Trang được đọc vào bộ nhớ trước sẽ bị đổi ra trước khi có yêu cầu đổi trang.
* Chiến lược đơn giản nhất, có thể triển khai một cách đơn giản bằng cách sử dụng hàng đợi FIFO.
* Trang bị trao đổi là trang nằm trong bộ nhớ lâu nhất.

1. **Kỹ thuật “Đệm trang” và ưu điểm ?**

* Hệ điều hành dành ra một số khung trống được kết nối thành danh sách liên kết gọi là các trang đệm.
* Khi có yêu cầu đổi trang, một trang bị đổi như bình thường nhưng nội dung trang này không bị xóa ngay khỏi bộ nhớ để nhường cho trang mới. Thay vào đó, khung chứa trang được đánh dấu là khung trống và thêm vào cuối danh sách trang đệm.
* Thay vì nạp vào khung chứa trang vừa bị đổi, trang mới sẽ được nạp vào khung đứng đầu trong danh sách trang đệm.
* Tới thời điểm thích hợp, hệ thống sẽ ghi nội dung các trang trong danh sách đệm ra đĩa.
* Kỹ thuật đệm trang cho phép cải tiến tốc độ do một số lý do sau:

+ Nếu trang bị đổi có nội dung cần ghi ra đĩa, hệ điều hành vẫn có thể nạp trang mới vào ngay, việc ghi ra đĩa sẽ được lùi lại tới một thời điểm muộn hơn. Thao tác ghi ra đĩa có thể thực hiện đồng thời với nhiều trang nằm trong danh sách được đánh dấu trống. Việc ghi nhiều trang đồng thời như vậy thường tiết kiệm thời gian hơn do thao tác ghi đĩa được tiến hành theo khối lớn.

+ Trang bị đổi vẫn được giữ trong bộ nhớ một thời gian. Trong thời gian này, nếu có yêu cầu truy cập, trang sẽ được lấy ra từ danh sách đệm và sử dụng ngay mà không cần nạp lại từ đĩa. Vùng đệm khi đó đóng vai trò giống như bộ nhớ cache.

1. **Các phương pháp xác định số lượng khung trang tối đa cấp cho mỗi chương trình và phạm vi cấp phát.**

* **Phương pháp cấp phát số lượng khung cố định**
* HĐH cấp cho tiến trình một số lượng cố định khung để chứa trang nhớ của mình. Số lượng này được xác định vào thời điểm tạo mới tiến trình và không thay đổi trong quá trình tiến trình tồn tại.
* Đến đây lại có hai cách tính số lượng khung tối đa:

+ Cấp phát bằng nhau: Mỗi tiến trình được cấp số khung tối đa giống nhau, không phụ thuộc vào đặc điểm riêng của tiến trình. Số lượng khung tối đa khi đó được xác định dựa trên kích thước bộ nhớ và mức độ đa chương trình mong muốn.

+ Cấp phát không bằng nhau: Số lượng khung tối đa cấp cho tiến trình có thể khác nhau và được tính toán dựa trên đặc điểm tiến trình. Cách đơn giản nhất là cấp cho mỗi tiến trình số lượng khung tỷ lệ thuận với kích thước tiến trình. Trong những hệ thống có quy định mức ưu tiên cho tiến trình, tiến trình với mức ưu tiên cao hơn có thể được cấp nhiều khung hơn tiến trình với mức ưu tiên thấp.

* **Phương pháp cấp phát số lượng khung thay đổi**
* Số lượng khung tối đa cấp cho mỗi tiến trình có thể thay đổi trong quá trình thực hiện.
* Việc thay đổi số khung tối đa phụ thuộc vào tình hình thực hiện của tiến trình. Tiến trình có tần suất đổi trang cao sẽ được cấp thêm khung nhớ, trong khi đó tiến trình có tần suất đổi trang thấp có thể bị thu hồi bớt khung.
* Phương pháp này cho phép sử dụng bộ nhớ hiệu quả hơn phương pháp cố định. Tuy nhiên, để thay đổi số lượng khung tối đa một cách hợp lý, HĐH cần theo dõi và xử lý thông tin về tình hình sử dụng bộ nhớ của tiến trình.
* **Phạm vi cấp phát khung**
* Phạm vi cấp phát được phân thành:

+ Cấp phát toàn cục (global): Cho phép tiến trình đổi trang mới vào bất cứ khung nào (không bị khóa), kể cả khung đã được cấp phát cho tiến trình khác.

+ Cấp phát cục bộ (local): Trang chỉ được đổi vào khung đang được cấp cho chính tiến trình đó.

* Phạm vi cấp phát có quan hệ mật thiết với số lượng khung tối đa:

+ Số lượng khung cố định tương ứng với phạm vi cấp phát cục bộ.

+ Số lượng khung thay đổi tương ứng với phạm vi cấp phát toàn thể.

1. **Tình trạng trì trệ (thrashing) trong quản lý bộ nhớ là gì ?**

* Khi số khung cấp cho tiến trình giảm xuống một mức nào đó, tiến trình sẽ rơi vào tình trạng thiếu bộ nhớ và phải đổi trang liên tục, tình trạng này được gọi là trì trệ (thrashing). Một tiến trình rơi vào tình trạng trì trệ khi thời gian đổi trang của tiến trình lớn hơn thời gian thực hiện.
* Xảy ra khi:

+ Bộ nhớ máy tính có kích thước hạn chế.

+ Tiến trình đòi hỏi truy cập đồng thời nhiều trang nhớ.

+ Hệ thống có mức độ đa chương trình cao.

* Tình trạng trì trệ bộ nhớ ảnh hưởng nghiêm trọng tới tốc độ máy tính.

# **Chương 4: Quản lý tiến trình**