1. Quản lý bộ nhớ là gì? Các yêu cầu đối với quản lý bộ nhớ là gì?

* Quản lý bộ nhớ là công việc của hệ điều hành với sự hỗ trợ của phần cứng nhằm phân phối, sắp xếp các process trong bộ nhớ sao cho hiệu quả.
* Các yêu cầu đối với việc quản lý bộ nhớ:
  + Cấp phát bộ nhớ cho các process
  + Tái định vị (relocation): khi swapping,…
  + Bảo vệ: phải kiểm tra truy xuất bộ nhớ có hợp lệ không
  + Chia sẻ: cho phép các process chia sẻ vùng nhớ chung
  + Kết gán địa chỉ nhớ luận lý của user vào địa

1. Trình bày đặc điểm các loại địa chỉ bộ nhớ?

* Địa chỉ vật lý (physical address) (địa chỉ thực) là một vị trí thực trong bộ nhớ chính
* Địa chỉ luận lý (logical address) là một vị trí nhớ được diễn tả trong một chương trình (còn gọi là địa chỉ ảo virtual address).
  + Các trình biên dịch (compiler) tạo ra mã lệnh chương trình mà trong đó mọi tham chiếu bộ nhớ đều là địa chỉ luận lý
  + Địa chỉ tương đối (relative address) (địa chỉ khả tái định vị, relocatable address) là một kiểu địa chỉ luận lý trong đó các địa chỉ được biểu diễn tương đối so với một vị trí xác định nào đó trong chương trình. Ví dụ: 12 byte so với vị trí bắt đầu chương trình,…
  + Địa chỉ tuyệt đối (absolute address): địa chỉ tương đương với địa chỉ thực.

1. Địa chỉ lệnh và dữ liệu được chuyển đổi thành địa chỉ thực có thể xảy ra tại những thời điểm nào?

* Địa chỉ lệnh và dữ liệu được chuyển đổi thành địa chỉ thực có thể xảy ra tại ba thời điểm khác nhau:
  + Compile time: nếu biết trước địa chỉ bộ nhớ của chương trình thì có thể kết gán địa chỉ tuyệt đối lúc biên dịch.
    - Ví dụ: chương trình .COM của MS-DOS
    - Khuyết điểm: phải biên dịch lại nếu thay đổi địa chỉ nạp chương trình
  + Load time: vào thời điểm loading, loader phải chuyển đổi địa chỉ khả tái định vị thành địa chỉ thực dựa trên một địa chỉ nền
    - Địa chỉ thực được tính toán vào thời điểm nạp chương trình => phải tiến hành reload nếu địa chỉ nền thay đổi
  + Excution time: khi trong quá trình thực thi, process có thể được di chuyển từ segment này sang segment khác trong bộ nhớ thì quá trình chuyển đổi địa chỉ được trì hoãn đến thời điểm thực thi
    - Cần sự hỗ trợ của phần cứng cho việc ánh xạ địa chỉVí dụ: Trường hợp địa chỉ luận lý là relocatable thì có thể dùng thanh ghi base và limit,..
    - Sử dụng trong đa số các OS đa dụng trong đó cócác cơ chế swapping, paging, segmentation

1. Liên kết động (dynamic linking) là gì?

* Quá trình link đến một module ngoài (external module) được thực hiện sau khi đã tạo xong load module (i.e. file có thể thực thi, executable)
  + Ví dụ trong Windows: module ngoài là các file .DLL còn trong Unix, các module ngoài là các file .so (shared library)

1. Phân mảnh ngoại và phân mảnh nội là gì?

* Phân mảnh ngoại tồn tại khi tổng lượng bộ nhớ còn trống đủ để phục vụ một yêu cầu mới từ process, tuy nhiên do các phần trống đó không liên tục nhau mà nằm ở nhiều vị trí cách xa nhau, vì vậy có bộ nhớ trống mà không thể cấp phát được.
* Phân mảnh nội là hiện tượng khi một process chiếm dung lượng bộ nhớ nhỏ hơn dung lượng mà nó được cấp phát (tức là phần bộ nhớ không được sử dụng nằm bên trong phân vùng đang được cấp phát cho process).

1. Như thế nào là phân vùng động, phân vùng cố định? Các chiến lược placement dùng để làm gì? Có những chiến lược nào?

* Phân vùng cố định:
  + Khi khởi động hệ thống, bộ nhớ chính được chia thành nhiều phần rời nhau gọi là các partition có kích thước bằng nhau hoặc khác nhau. Từng phần sẽ được cung cấp cho một và chỉ một process. Đó là Fixed-Partition.
  + Process nào vừa với kích thước partition thì có thể được nạp vào partition đó (vừa tức là có kích thước nhỏ hơn hoặc bằng kích thước partition).
  + Nếu chương trình có kích thước lớn hơn partition thì phải dùng cơ chế overlay.
  + Fixed-Partitioning không hiệu quả do bị phân mảnh nội : Một chương trình dù lớn hay nhỏ đều được cấp phát trọn một partition.
  + Các chiến lược Placement (đưa process vào bộ nhớ) :
    - Partition có kích thước bằng nhau.
      * Nếu còn partition trống => Process mới sẽ được nạp vào partition đó.
      * Nếu không còn partition trống, nhưng trong đó có process đang bị blocked => Swap process đó ra bộ nhớ phụ, nhường chỗ cho process mới.
    - Nếu các partition không bằng nhau :
      * Giải pháp 1 :
        + Gán mỗi process vào partition nhỏ nhất phù hợp với nó.
        + Có hàng đợi cho mỗi partition.
        + Giảm thiểu phân mảnh nội.
        + Vấn đề ở đây là : Sẽ có một số hàng đợi trống không (vì không có process với kích thước tương ứng), trong khi đó lại có partition với hàng đợi dày đặc.
      * Giải pháp 2 :
        + Dùng chung 1 hàng đợi cho tất cả các partition.
        + Khi cần nạp một process vào bộ nhớ chính => Chọn partition nhỏ nhất còn trống.
* Phân vùng động:
  + Dynamic Partitioning sẽ chia bộ nhớ thành nhiều phần như Fixed-Partition, tuy nhiên số lượng partition không cố định và partition có thể có kích thước khác nhau.
  + Mỗi process được cấp phát chính xác dung lượng bộ nhớ cần thiết.
  + Dynamic Partitioning không gây ra hiện tượng phân mảnh nội, nhưng lại gây ra hiện tượng phân mảnh ngoại.
  + Chiến lược placement :
    - Best-Fit : Chọn khối nhớ trống nhỏ nhất đủ cho process.
    - First-Fit : Chọn khối nhớ trống phù hợp đầu tiên kể từ đầu bộ nhớ.
    - Next-Fit : Chọn khối nhớ trống phù hợp đầu tiên kể từ vị trí cấp phát cuối cùng (nếu chưa có khối nhớ nào được cấp phát thì khối nhớ đầu tiên của bộ nhớ sẽ được chọn).
    - Worst-Fit : Chọn khối nhớ trống lớn nhất.

1. Cơ chế phân trang là gì? Cơ chế chuyển đổi địa chỉ trong phân trang được thực hiện như thế nào?

* Cơ chế phân trang (paging) là một phương pháp quản lý bộ nhớ trong hệ thống máy tính, nơi bộ nhớ vật lý được chia thành các phần nhỏ gọi là trang (page) có kích thước cố định và các quá trình hoặc chương trình cũng được chia thành các khối tương ứng gọi là trang với kích thước tương đương. Mỗi trang trong bộ nhớ vật lý được ánh xạ tới một trang tương ứng trong bộ nhớ ảo hoặc địa chỉ luận lý.
* Cơ chế chuyển đổi địa chỉ trong phân trang thường được thực hiện thông qua bảng trang (page table). Bảng trang lưu trữ ánh xạ giữa các trang vật lý và trang ảo tương ứng của chúng. Khi một quá trình cố gắng truy cập dữ liệu, hệ điều hành sẽ sử dụng cơ chế này để dịch địa chỉ luận lý thành địa chỉ vật lý.
* Quá trình chuyển đổi địa chỉ thường bao gồm các bước sau:
  + Phân trang: Bộ nhớ vật lý và địa chỉ luận lý được chia thành các trang có kích thước cố định.
  + Bảng trang (Page table): Hệ thống duy trì bảng trang để lưu trữ ánh xạ giữa trang vật lý và trang ảo tương ứng của chúng.
  + Dịch địa chỉ: Khi một quá trình cần truy cập dữ liệu, địa chỉ luận lý được chuyển đổi thành địa chỉ vật lý thông qua bảng trang. Hệ điều hành sử dụng bảng trang để xác định trang vật lý tương ứng với trang ảo và truy cập đến vị trí thích hợp trong bộ nhớ vật lý.

1. Bảng trang được cài đặt và tổ chức như thế nào? Thời gian truy xuất hiệu dụng (effective access time) được xác định như thế nào?

* Bảng trang (Page table) là một cấu trúc dữ liệu quan trọng trong hệ thống phân trang của máy tính, được sử dụng để quản lý việc ánh xạ địa chỉ luận lý (địa chỉ ảo) của chương trình thành địa chỉ vật lý tương ứng trên bộ nhớ thực (địa chỉ thực).
* Cách tổ chức và triển khai bảng trang có thể thay đổi theo kiến trúc cụ thể của hệ thống, nhưng một cách tổ chức thông thường bao gồm: bảng trang đơn giản, bảng trang đa cấp….
* Thời gian truy xuất hiệu dụng (effective access time) trong hệ thống máy tính có thể được xác định bằng cách tính tổng hợp thời gian của các phần khác nhau mà một hệ thống phải dùng để thực hiện một truy xuất bộ nhớ. Đây có thể bao gồm:
  + Thời gian truy xuất bảng trang: Đây là thời gian cần thiết để hệ thống truy cập vào bảng trang để tìm kiếm địa chỉ vật lý tương ứng với địa chỉ ảo. Thời gian này phụ thuộc vào cấu trúc và kích thước của bảng trang cũng như thuật toán tìm kiếm được sử dụng.
  + Thời gian truy xuất bộ nhớ vật lý: Sau khi xác định được địa chỉ vật lý từ bảng trang, thời gian để truy cập đến vị trí cụ thể trong bộ nhớ vật lý. Thời gian này có thể bao gồm việc truy cập vào bộ nhớ RAM, bộ nhớ cache, hoặc có thể đến cả ổ đĩa cứng nếu dữ liệu không có sẵn trong bộ nhớ chính.
  + Thời gian xử lý ngoại lệ (page fault handling time): Trong trường hợp trang mà chương trình cần không có sẵn trong bộ nhớ và cần được nạp từ đĩa cứng (page fault), thời gian xử lý ngoại lệ sẽ được tính vào thời gian truy xuất hiệu dụng. Điều này có thể tăng đáng kể thời gian truy xuất tổng cộng.

1. Cơ chế hoán vị là gì?

* Một process có thể tạm thời bị swap ra khỏi bộ nhớ chính và lưu trên một hệ thống lưu trữ phụ. Sau đó, process có thể được nạp lại vào bộ nhớ để tiếp tục quá trình thực thi.

1. (Bài tập mẫu)
2. (Bài tập mẫu)
3. Bài tập mẫu)

* Số bit offset: Được xác định bởi số bit còn lại sau khi dành cho bảng trang cấp 1 và cấp 2.
  + Số bit offset = 32 - (9 + 11) = 32 - 20 = 12 bit.
* Kích thước trang: Kích thước trang được xác định bởi số bit offset.
  + Kích thước trang = 2^số bit offset = 2^12 = 4KB.
* Số lượng trang: Số lượng trang được tính bằng cách tính toán số bit dành cho mỗi bảng trang và kết hợp chúng.
  + Số lượng trang cho bảng trang cấp 1 = 2^số bit cho bảng trang cấp 1 = 2^9 = 512 trang.
  + Số lượng trang cho bảng trang cấp 2 = 2^số bit cho bảng trang cấp 2 = 2^11 = 2048 trang.
* Số lượng trang ảo trong hệ thống sẽ là tích của số trang của bảng trang cấp 1 và bảng trang cấp 2:
  + Số lượng trang ảo = Số trang bảng trang cấp 1 \* Số trang bảng trang cấp 2 = 512 \* 2048 = 1048576 trang.
* Vậy, trong hệ thống này, kích thước của một trang là 4KB và tổng số trang ảo là 1048576 trang.
* Trong trường hợp địa chỉ ảo 32-bit được phân tách thành 4 trường a, b, c, d với 3 trường đầu tiên được dùng cho bảng trang 3 cấp và trường thứ 4 dành cho offset, số lượng trang không phụ thuộc vào kích thước của tất cả 4 trường.
* Số lượng trang được xác định chủ yếu bởi số bit được sử dụng cho các trường của bảng trang, không phụ thuộc trực tiếp vào số bit dành cho offset. Trong trường hợp này:
  + Các trường a, b, c (3 trường đầu) đều được sử dụng cho bảng trang. Số lượng trang sẽ phụ thuộc vào số bit được dành cho mỗi trường này.
  + Trường d (offset) không ảnh hưởng trực tiếp đến số lượng trang mà chỉ xác định kích thước của từng trang. Kích thước trang phụ thuộc vào số bit của trường offset.

1. (Bài tập mẫu)
2. 2 \* 124 ns=248 ns
3. Nếu TLB hit -> 34+124 = 158 ns

Nếu TLB miss-> 34+124\*2 = 282 ns

Thời gian truy xuất bộ nhớ hiệu dụng: 95%\*158+5%\*282=166.2 ns

* Xét:

0.87\*(t+24)+0.13\*(t\*2+24)=175 ns

t = 133.633 ns

* Xét:

a\*(250+26)+(1-a)\*(250\*2+26)=182 ns

a = 1.376(không thỏa mãn vì >100%)