**Chapter 4 - Memory**

1. What is the problem with no memory abstraction?

Sự trừu tượng hóa bộ nhớ đơn giản nhất là không có sự trừu tượng nào cả. Điều này có nghĩa là chúng tôi không thể có 2 chương trình chạy trong bộ nhớ chính cùng một lúc, vì điều này sẽ gây ra sự không nhất quán trong dữ liệu. Các quy trình có thể xóa đầu ra / đầu vào bằng văn bản của nhau và gây ra sự không nhất quán nghiêm trọng

2. What is swapping - batch system?

Với hệ thống hàng loạt, việc tổ chức bộ nhớ thành các phân vùng cố định rất đơn giản và hiệu quả. Mỗi công việc được tải vào một phân vùng khi nó đến đầu hàng đợi. Nó vẫn còn trong bộ nhớ cho đến khi nó kết thúc. Miễn là có thể lưu đủ công việc trong bộ nhớ để CPU luôn bận rộn, thì không có lý do gì để sử dụng bất cứ điều gì phức tạp hơn.

Với hệ thống chia sẻ thời gian hoặc máy tính cá nhân định hướng đồ họa, tình hình lại khác. Đôi khi không có đủ bộ nhớ chính để chứa tất cả các tiến trình hiện đang hoạt động, vì vậy các tiến trình thừa phải được lưu trên đĩa và đưa vào để chạy động.

Có thể sử dụng hai cách tiếp cận chung để quản lý bộ nhớ, tùy thuộc (một phần) vào phần cứng có sẵn. Chiến lược đơn giản nhất, được gọi là hoán đổi , bao gồm việc đưa toàn bộ quá trình vào, chạy nó một lúc, sau đó đặt nó trở lại đĩa. Chiến lược khác, được gọi là bộ nhớ ảo , cho phép các chương trình chạy ngay cả khi chúng chỉ nằm một phần trong bộ nhớ chính.

3. What are the two methods of memory management?

Static loading : - tải toàn bộ chương trình vào một địa chỉ cố định. Nó yêu cầu nhiều không gian bộ nhớ hơn.

Dynamic loading : - Toàn bộ chương trình và tất cả dữ liệu của một tiến trình phải nằm trong bộ nhớ vật lý để tiến trình thực thi. Vì vậy, kích thước của một tiến trình được giới hạn ở kích thước của bộ nhớ vật lý. Để sử dụng bộ nhớ thích hợp, tải động được sử dụng. Trong tải động, một quy trình không được tải cho đến khi nó được gọi. Tất cả các quy trình đều nằm trên đĩa ở định dạng tải có thể di chuyển lại được. Một trong những lợi thế của tải động là các quy trình không sử dụng sẽ không bao giờ được tải. Việc tải này rất hữu ích khi cần một lượng lớn mã để xử lý nó một cách hiệu quả.

4. What are the advantages of the linked list method (Section 4.2.1 & 4.2.2)?

Ưu điểm của Linked List:

Cấu trúc dữ liệu động: Danh sách được liên kết là một sự sắp xếp động để nó có thể phát triển và thu nhỏ trong thời gian chạy bằng cách cấp phát và phân bổ bộ nhớ . Vì vậy không cần đưa ra kích thước ban đầu của danh sách liên kết.

Không lãng phí bộ nhớ: Trong danh sách được Liên kết, có thể đạt được việc sử dụng bộ nhớ hiệu quả vì kích thước của danh sách được liên kết tăng hoặc giảm trong thời gian chạy, do đó không lãng phí bộ nhớ và không cần cấp phát trước bộ nhớ.

Thực hiện: Các cấu trúc dữ liệu tuyến tính như ngăn xếp và hàng đợi thường được thực hiện dễ dàng bằng cách sử dụng danh sách liên kết.

Thao tác chèn và xóa : Thao tác chèn và xóa khá dễ dàng hơn trong danh sách liên kết. Không cần thay đổi các phần tử sau khi chèn hoặc xóa một phần tử mà chỉ cần cập nhật địa chỉ có trong con trỏ tiếp theo.

Nhược điểm của Linked List:

Sử dụng bộ nhớ: Cần nhiều bộ nhớ hơn trong danh sách liên kết so với một mảng. Bởi vì trong danh sách liên kết, một con trỏ cũng được yêu cầu để lưu trữ địa chỉ của phần tử tiếp theo và nó yêu cầu thêm bộ nhớ cho chính nó.

Truyền tải: Trong một danh sách được liên kết , việc truyền tải mất nhiều thời gian hơn so với một mảng. Không thể truy cập trực tiếp vào một phần tử trong danh sách liên kết như trong mảng theo chỉ mục. Ví dụ, để truy cập một nút ở vị trí n, người ta phải đi qua tất cả các nút trước nó.

Đảo ngược: Trong một danh sách liên kết đơn không thể duyệt đảo ngược, nhưng trong trường hợp danh sách được liên kết kép , nó có thể thực hiện được vì nó chứa một con trỏ đến các nút được kết nối trước đó với mỗi nút. Để thực hiện bộ nhớ bổ sung này là cần thiết cho con trỏ trở lại, do đó, sẽ có sự lãng phí bộ nhớ.

Truy cập ngẫu nhiên: Không thể truy cập ngẫu nhiên trong danh sách được liên kết do phân bổ bộ nhớ động của nó .

5. Understand algorithms to allocate memory: first fit, next fit, best fit, worst fit (Sectio 4.2.2).

first fit:

là phân bổ phân vùng hoặc lỗ trống đầu tiên đủ lớn để có thể chứa quá trình. Nó kết thúc sau khi tìm thấy phân vùng miễn phí phù hợp đầu tiên.

Thuận lợi

Thuật toán nhanh nhất vì nó tìm kiếm ít nhất có thể.

Bất lợi

Các vùng nhớ chưa sử dụng còn lại sau khi cấp phát sẽ trở thành lãng phí nếu nó quá nhỏ. Vì vậy, yêu cầu yêu cầu bộ nhớ lớn hơn không thể được thực hiện.

next fit:

là một phiên bản sửa đổi của phù hợp đầu tiên. Nó bắt đầu phù hợp đầu tiên để tìm một phân vùng miễn phí. Khi được gọi vào lần tiếp theo, nó sẽ bắt đầu tìm kiếm từ nơi nó đã dừng lại, không phải từ đầu.

Best Fit:

giải quyết việc phân bổ phân vùng miễn phí nhỏ nhất đáp ứng yêu cầu của quy trình yêu cầu. Thuật toán này đầu tiên tìm kiếm toàn bộ danh sách các phân vùng miễn phí và xem xét lỗ nhỏ nhất là đủ. Sau đó, nó cố gắng tìm một lỗ gần với kích thước quy trình thực tế cần thiết.

Thuận lợi

Việc sử dụng bộ nhớ tốt hơn nhiều so với lần đầu tiên vì nó tìm kiếm phân vùng trống nhỏ nhất có sẵn đầu tiên.

Bất lợi

Nó chậm hơn và thậm chí có thể có xu hướng lấp đầy bộ nhớ với những lỗ nhỏ vô dụng.

Worst fit:

là xác định phần lớn nhất còn lại miễn phí để phần còn lại sẽ đủ lớn để hữu ích. Nó là mặt trái của sự phù hợp tốt nhất.

Thuận lợi

Giảm tỷ lệ sản xuất các khoảng trống nhỏ.

Bất lợi

Nếu một quá trình yêu cầu bộ nhớ lớn hơn đến ở giai đoạn sau thì nó không thể được đáp ứng vì lỗ lớn nhất đã bị chia nhỏ và bị chiếm dụng.

6. What is the unit of virtual memory, and of physical memory?

Máy tính có thể giải quyết nhiều bộ nhớ hơn dung lượng được cài đặt vật lý trên hệ thống. Bộ nhớ bổ sung này được gọi là bộ nhớ ảo và nó là một phần của đĩa cứng được thiết lập để mô phỏng RAM của máy tính. Bộ nhớ ảo là một kỹ thuật phổ biến được sử dụng trong hệ điều hành (OS) của máy tính.

Bộ nhớ ảo sử dụng cả phần cứng và phần mềm để hoạt động. Khi một ứng dụng đang được sử dụng, dữ liệu từ chương trình đó được lưu trữ trong một địa chỉ vật lý sử dụng RAM. Một đơn vị quản lý bộ nhớ (MMU) ánh xạ địa chỉ tới RAM và tự động dịch các địa chỉ.

7. What is the page table mainly for?

page table là cấu trúc dữ liệu được sử dụng bởi hệ thống bộ nhớ ảo trong hệ điều hành máy tính để lưu trữ ánh xạ giữa địa chỉ ảo và địa chỉ vật lý.

8. What is TLB and what is that for?

TLB là một bộ nhớ đệm lưu trữ các bản dịch gần đây của bộ nhớ ảo sang bộ nhớ vật lý. Nó được sử dụng để giảm thời gian truy cập vào vị trí bộ nhớ của người dùng. Nó có thể được gọi là bộ đệm dịch địa chỉ.

9. Differentiate page faults, TLB soft misses and TLB hard misses.

Lỗi trang cứng

Nếu một trang được yêu cầu tồn tại trong tệp trang của đĩa cứng, nó được gọi là "HARD PAGE FAULT" .

Lỗi trang mềm

Nếu một trang được yêu cầu không nằm trên đĩa cứng nhưng được tìm thấy ở một nơi khác trong bộ nhớ, nó được gọi là " SOFT PAGE FAULT " .

Thời gian cần thiết để lấy một trang từ bộ nhớ phụ và khôi phục nó từ bộ nhớ chính sau khi tải trang cần thiết được gọi là "PAGE FAULT TIME"

10. What is the essence of PRAs?

=>> is an experiential learning activity conducted in community, centrally, systematically, semi- structured, over a short period of time

**Question for Lab**

1. What is the page table mainly for?

=>> A page table is the data structure used by a virtual memory system in a computer operating system to store the mapping between virtual addresses and physical addresses.

1. What is TLB and what is that for?

=>> A translation lookaside buffer (TLB) is a memory cache that stores the recent translations of virtual memory to physical memory. It is used to reduce the time taken to access a user memory location. It can be called an address-translation cache. It is a part of the chip's memory-management unit (MMU).

1. Differentiate page faults, TLB soft misses and TLB hard misses.

=>> **TLB Miss**

If we don't find the page frame number inside the TLB, it is called a TLB miss only then we go to the page table to look for the corresponding page frame number.

**TLB Hit**

If we find the page frame number in TLB, its called TLB hit, and we don't need to go to page table.

**Page Fault**

Occurs when the page accessed by a running program is not present in physical memory. It means the page is present in the secondary memory but not yet loaded into a frame of physical memory.

1. A memory free in 4 frames. Which state of the memory after the page 4 is accessed when the requested page as 2 3 2 0 1 5 2 4 5 3 2 5 2 using LRU

=>> answer 7.

1. Assume that the Page Table below is in effect. The number of lines per page is 400. The actual memory location for line 1634 is \_\_34\_\_\_\_ .

|  |  |
| --- | --- |
| Page Number | Page Frame Number |
| 0  1  2  3  4 | 8  10  5  11  0 |