Phương pháp phân phối đĩa liên tục:

* Phương pháp phân phối đĩa liên tục yêu cầu mỗi tệp phải chiếm một tập hợp địa chỉ liền kề trên đĩa. Địa chỉ đĩa xác định thự tự tuyến tính trên đĩa. Khi một tệp phải được lưu trữ trên đĩa, hệ thống sẽ tìm kiếm tập hợp khối liền kề theo yêu cầu của kích thước tệp, tức là hệ thống sẽ đợi cho đến khi tìm thấy số khối bộ nhớ cần thiết theo trình tự. Khi có dung lượng, hệ thống sẽ lưu tệp trong đĩa và tao một mục nhập trong thư mục.
* Thự mục nhập của tệp với phương pháp phân phối đĩa liên tục phải bao gồm:
  + Địa chỉ của khối bắt đầu.
  + Độ dài của khối phân phối.
* Với thự tự này, truy cập khối b + 1 sau khối b thường không yêu cầu chuyển động đầu đọc. việc phân phối liên tục của một tệp được xác định bởi địa chỉ đĩa và độ dài của khối đầu tiên. Nếu tệp dài n khối và bắt đàu ở vị trí b, thì tệp sẽ chiếm các khối b, b +1, b + 2,…, b + n – 1. Thư mục nhập cho mỗi tệp cho biết địa chỉ của khối bắt đầu và độ dài của vùng được phân phối cho tệp này.

Truy suất một trang được đánh dấu bất hợp lệ sẽ làm phát sinh lỗi trang, hệ điều hành sẽ thực hiện:

**Bước 1:** Kiểm tra việc truy xuất tới bộ nhớ có hợp lệ hay không. Nếu là hợp lệ, chuyển sang bước 2; ngược lại kết thúc chương trình.

**Bước 2:** Tìm vị trí chứa trang cần truy xuất trên đĩa từ.

**Bước 3:** Tìm 1 trang vật lý trống trong bộ nhớ chính. Nếu tìm thấy, chuyển sang bước 4; ngược lại thực hiện thay thế trang.

**Bước 4:** Thực hiện nạp trang (chuyển trang muốn truy xuất từ bộ nhớ ngoài vào bộ nhớ trong, nạp trang này vào trang vật lý trống, cập nhật nội dung bảng quản lý trang).

**Bước 5:** Tái kích hoạt chương trình.

Các thuật toán thay thế trang:

* Thay thế trang có thời gian tồn tại trong bộ nhớ lâu nhất (FIFO – First In First Out)
* Thay thế trang có lần sử dụng cuối cùng, cách thời điểm hiện tại lâu nhất (LRU – Last Recently Used).
* Thay thế trang có tần suất sử dụng thấp nhất (Less Frequently Used)
* Thay thế trang có tần suất sử dụng nhiều nhất (MFU – Most Frequently Used)

Nguyên tắc:

* **FIFO:** ghi nhận thời điểm trang vào bộ nhớ chính. Khi cần thay thế trang, trang ở trong bộ nhớ lâu nhất sẽ được chọn.
* **LRU:** Với mỗi trang, ghi nhận thời điểm cuối cùng trang được truy cập, trang được chọn để thay thế sẽ là trang lâu nhất chưa được truy xuất.
* **LFU:** Thay thế trang có giá trị biến đếm nhỏ nhất, nghĩa là trang ít được sử dụng nhất.
* **MFU:** Thay thế trang có giá trị biến đếm lớn nhất, nghĩa là trang được sử dụng nhiều nhất.

Để minh họa các thuật toán thay thế trang sẽ trình bày, chuỗi truy xuất dược sử dụng là:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

**Thuật toán FIFO:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 7 | 7 |
|  | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
|  |  | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 |
| \* | \* | \* | \* |  | \* | \* | \* | \* | \* | \* |  |  | \* | \* |  |  | \* | \* | \* |

Ghi chú: \* có lỗi trang.

**Đánh giá:** Để áp dụng thuật toán FIFO, thực tế không nhất thiết phải ghi nhận thời điểm mỗi trang được nạp vào bộ nhớ, mà chỉ cần tổ chức quản lý các trang trong bộ nhớ trong một danh sách FIFO, khi đó trang đầu danh sách sẽ được chọn để thay thế. Thuật toán thay thế trang FIFO dễ hiểu, dễ cài đặt. Tuy nhiên khi thực hiện không phải lúc nào cũng có kết quả tốt : trang được chọn để thay thế có thể là trang chức nhiều dữ liệu cần thiết, thường xuyên được sử dụng nên được nạp sớm, do vậy khi bị chuyển ra bộ nhớ phụ sẽ nhanh chóng gây ra lỗi trang. 114 Số lượng lỗi trang xảy ra sẽ tăng lên khi số lượng khung trang sử dụng tăng. Hiện tượng này gọi là nghịch lý Belady

**Thuật toán LRU:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
| 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 |
|  |  | 1 | 1 | 1 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 |
| \* | \* | \* | \* |  | \* |  | \* | \* | \* | \* |  |  | \* |  | \* |  | \* |  |  |

**Đánh giá:** Thuật toán FIFO sử dụng thời điểm nạp để chọn trang thay thế, thuật toán tối ưu lại dùng thời điểm trang sẽ được sử dụng, vì thời điểm này không thể xác định trước nên thuật toán LRU phải dùng thời điểm cuối cùng trang được truy xuất – dùng quá khứ gần để dự đoán tương lai. Thuật toán này đòi hỏi phải được cơ chế phần cứng hỗ trợ để xác định một thứ tự cho các trang theo thời điểm truy xuất cuối cùng. Có thể cài đặt theo một trong hai cách