

**ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**-Khoa Công nghệ thông tin-**



**BÀI TẬP LỚN**

**MÔN: NGUYÊN LÝ HỆ ĐIỀU HÀNH**

**Đề tài: Nguyên cứu và tìm hiểu về quản lý bộ nhớ ngoài trong HĐH Windows**

**Giáo viên hướng dẫn: Nguyễn Thanh Hải**

**Sinh viên thực hiện: 1. Nguyễn Thị Điệp**

**2. Phan Thùy Linh**

**3. Nguyễn Thị Hồng Nhung**

**4. Trịnh Thu Phương**

**5. Trần Đặng Phương Thảo**

**6. Phạm Huyền Thương**

**PHIẾU HỌC TẬP CÁ NHÂN/NHÓM**

**I. Thông tin chung**

1. Tên lớp: 2021IT6025005

2. Tên nhóm: 5

3. Họ và tên thành viên trong nhóm: Nguyễn Thị Điệp, Phan Thùy Linh, Nguyễn Thị Hồng Nhung, Trịnh Thu Phương, Trần Đặng Phương Thảo, Phạm Huyền Thương.

**II. Nội dung học tập**

1. Tên chủ đề: Nghiên cứu tìm hiểu về quản lý Bộ nhớ Ngoài trong HĐH Windows.

2. Nội Dung và yêu cầu.

Nắm được những kiến thức về Nguyên lý của việc quản lý Bộ nhớ ngoài (các phương pháp quản lý không gian nhớ tự do, cấp phát không gian nhớ, lập lịch cho đĩa …) từ đó gắn với HĐH Windows.

**III. Nhiệm vụ học tập**

1. Hoàn thành báo cáo bài tập lớn theo đúng thời gian quy định

2. Báo cáo sản phẩm nghiên cứu được giao trước giảng viên và các sinh viên khác.

**IV. Học liệu sử dụng cho bài tập lớn**

1. Tài liệu học tập:

* *Giáo trình nguyên lý hệ điều hành* của nhà xuất bản Hà Nội.
* *Giáo trình nguyên lý hệ điều hành* – Trường Đại học Công Nghiệp Hà Nội, Khoa Công Nghệ Thông Tin.
* Và một số trang web trực tuyến khác

**KẾ HOẠCH THỰC HIỆN TIỂU LUẬN, BÀI TẬP LỚN**

Tên lớp: 2021IT6025005 Khóa: 16

Tên nhóm: 5 Họ tên thành viên của nhóm: Nguyễn Thị Điệp, Phan Thùy Linh, Nguyễn Hồng Nhung, Trịnh Thu Phương, Trần Đặng Phương Thảo, Phạm Huyền Thương.

|  |  |  |  |
| --- | --- | --- | --- |
| Tuần | Người thực hiện | Nội dung công việc | Phương pháp thực hiện |
| 5 - 6 | Cả nhóm | Phân chia, đảm nhận công việc |  |
| 7 - 13 | Nguyễn Thị Điệp |  | Đọc tài liệu trên mạng, giáo trình |
| 7 - 13 | Phan Thùy Linh |  | Đọc tài liệu trên mạng, giáo trình |
| 7 - 13 | Nguyễn Thị Hồng Nhung |  | Đọc tài liệu trên mạng, giáo trình |
| 7 - 13 | Trịnh Thu Phương |  | Đọc tài liệu trên mạng, giáo trình |
| 7-13 | Trần Đặng Phương Thảo |  | Đọc tài liệu trên mạng, giáo trình |
| 7-13 | Phạm Huyền Thương |  | Đọc tài liệu trên mạng, giáo trình |
| 13 -14 | Cả nhóm |  |  |

**Tên chủ đề:** Nghiên cứu tìm hiểu về quản lý Bộ nhớ Ngoài trong HĐH Windows.

Giảng viên

*Nguyễn Thanh Hải*

# LỜI NÓI ĐẦU

Windows là phần mềm hệ điều hành của hãng Microsoft. Với giao diện đồ họa thông qua hệ thống thực đơn và các hộp hội thoại đa dạng, hệ điều hành Windows dễ sử dụng và tương đối dễ học. Các hệ thống máy vi tính hiện nay thường dùng hệ điều hành Windows (Việt Nam là một trường hợp). Đến nay, hãng Microsoft đã phát triển các hệ điều hành Windows3x, Windows9x, Windows 2000, Windows ME, Windows XP và Windows 7, Windows 8 và Windows 10. Windows là hệ điều hành đa tác vụ, nghĩa là có thể thực hiện đồng thời nhiều công việc. Ví dụ, vừa nghe nhạc, vừa sử dụng phần mềm Excel để tính toán, hay phần mềm Winword để soạn thảo văn bản. Chính vì sự phổ biến của hệ điều hành windows nên việc quản lý bộ nhớ là một thành phần rất quan trọng của hệ điều hành. Trong tài liệu này sẽ đề cập đến bộ nhớ ngoài mà cụ thể là ổ đĩa cứng và các phương pháp tổ chức cũng như quản lý bộ nhớ ngoài.

Nhóm sinh viên thực hiện!

# MỤC LỤC

[**LỜI NÓI ĐẦU 3**](#_heading=h.1fob9te)

[**MỤC LỤC 4**](#_heading=h.mq4v2nbdgn60)

[**CHƯƠNG 1. MỞ ĐẦU 6**](#_heading=h.oxqrczzajlx)

[1.1. Hệ điều hành Windows 6](#_heading=h.29vugk7epecg)

[1.1.1. Hệ điều hành Window là gì? 6](#_heading=h.rkjsa4bnhen9)

[1.1.2. Lịch sử hệ điều hành Windows 6](#_heading=h.wofho5ed368k)

[1.1.3. Các hệ điều hành Window hiện nay 7](#_heading=h.5swhxeng9tff)

[1.1.3.1. Windows Home 7](#_heading=h.7kgphqxrp3x2)

[1.1.3.2. Windows Professional 7](#_heading=h.5b0d0u9j2g0g)

[1.1.4. Tính năng nổi bật của hệ điều hành Window là gì? 7](#_heading=h.sasj37h770b9)

[1.1.4.1. Bảng điều khiển 7](#_heading=h.gs4yxctjmfrj)

[1.1.4.2. Cortana 7](#_heading=h.nhs8i86cjwyi)

[1.1.4.3. File Explorer 8](#_heading=h.1kwyff87qbms)

[1.1.4.4. Trình duyệt Internet 8](#_heading=h.hxj79vhe0eq0)

[1.1.4.5. Microsoft Paint 8](#_heading=h.mddf9bgfe1cz)

[1.1.4.6. Thao tác vụ 8](#_heading=h.9nrzdrbtc24i)

[1.1.4.7. Start Menu 8](#_heading=h.5nw3tof4emdy)

[1.1.4.8. Trình quản lý tác vụ 9](#_heading=h.cgw66v1eujq6)

[1.1.4.9. Disk Cleanup 9](#_heading=h.v166r6xpedsc)

[1.2. Tổng quan về quản lý bộ nhớ ngoài 9](#_heading=h.fe63i9c3t8hf)

[**CHƯƠNG 2. CẤU TRÚC VẬT LÝ-THƯ MỤC THIẾT BỊ 10**](#_heading=h.lab4s46k05e1)

[2.1. Cấu trúc vật lý 10](#_heading=h.nfobmgknih7n)

[2.1.1. Đĩa từ (Platter) 10](#_heading=h.5ezr86xqf1wg)

[2.1.2. Các rãnh từ (Track) 10](#_heading=h.8lvhfbkgjym1)

[2.1.3. Sector 11](#_heading=h.8lc4feslyol9)

[2.1.4. Cylinder 11](#_heading=h.hhawkw5xtx5i)

[2.1.5. Đầu đọc/ghi (Read Write Heads) 11](#_heading=h.mr0d6zbqeao)

[2.1.6. Cần di chuyển đầu đọc/ghi (Disk controller) 12](#_heading=h.e0e3ftjexan6)

[2.2. Thư mục thiết bị 12](#_heading=h.yq01gmy11obe)

[**CHƯƠNG 3 :CÁC DẠNG LƯU TRỮ DỮ LIỆU TRÊN HĐH WINDOWS 13**](#_heading=h.2d7gzgu8iqwo)

[3.1. Lưu trữ cơ bản ( Basic storage). 13](#_heading=h.2ydrsjhc6yls)

[3.2. Lưu trữ động (Dynamic storage) 14](#_heading=h.jhdlzoj0my8u)

[3.2.1. Spanned volume 15](#_heading=h.cle1f31qu1w3)

[3.2.2. Simple volume 15](#_heading=h.ucgmtslle5e)

[3.2.3 Striped Volume (RAID-0) 16](#_heading=h.pi1fs2yw6ruh)

[3.2.4 Mirror volume (RAID-1) 16](#_heading=h.c4x0m2tiaad6)

[3.2.5. Volume RAID-5g 17](#_heading=h.e4ssppmnerkx)

[**CHƯƠNG 4. CHƯƠNG TRÌNH QUẢN LÝ BỘ NHỚ NGOÀI DISK MANAGER 19**](#_heading=h.4z9plze7cm9c)

[4.1. Thuộc tính của đĩa 19](#_heading=h.wx6qm7ntl103)

[4.2. Thuộc tính của phân vùng hoặc đĩa cục bộ 19](#_heading=h.d42ossna9xxx)

[**CHƯƠNG 5. QUẢN LÝ KHÔNG GIAN NHỚ TỰ DO TRONG HỆ ĐIỀU HÀNH 21**](#_heading=h.8ulb7cfkqdve)

[5.1. Quản lý bộ nhớ bằng phương pháp dùng bit vector (Bitmap) 21](#_heading=h.pl4frowdvpjb)

[5.2. Quản lý bộ nhớ bằng phương pháp liệt kê( free list) 22](#_heading=h.wlewx1o24dbi)

[5.3. Quản lý bộ nhớ bằng phương pháp lập nhóm(grouping) 22](#_heading=h.3hrfl9d70guk)

[5.4. Quản lý bộ nhớ bằng phương pháp đếm (counting) 23](#_heading=h.6r3w0oknle4o)

[**CHƯƠNG 6: CÁC PHƯƠNG PHÁP CẤP PHÁT KHÔNG GIAN NHỚ TỰ DO 25**](#_heading=h.tyjcwt)

[6.1. Cấp phát liên tục (Contiguous) 25](#_heading=h.3dy6vkm)

[6.2. Cấp phát liên kết (Linked) 26](#_heading=h.1t3h5sf)

[6.3. Cấp phát theo chỉ số (Index) 26](#_heading=h.2s8eyo1)

[**CHƯƠNG 7: LẬP LỊCH CHO ĐĨA TỪ TRONG HỆ ĐIỀU HÀNH WINDOWS 30**](#_heading=)

[7.1. Khái niệm về lập lịch cho đĩa 30](#_heading=h.26in1rg)

[7.2. Các thuật toán lập lịch cho đĩa 31](#_heading=h.35nkun2)

[7.2.1.First come first Served(FCFS) 31](#_heading=h.44sinio)

[7.2.2. Shortest Seek time First 32](#_heading=h.4o3k7tvbxbpu)

[7.2.3. Scan 33](#_heading=h.pyrvxfz0r8x9)

[7.2.4. C-Scan 33](#_heading=h.a204p4oli3ry)

[7.2.5. Look 34](#_heading=h.v6jtfho05rq7)

[7.2.6. C-Look 34](#_heading=h.czyut1oc6l1y)

[7.3. Quản lý lỗi 35](#_heading=h.tqly67yjhj2y)

[7.4. RAM Disks 36](#_heading=h.2c8p3egovnmk)

[7.5. Interleave 37](#_heading=h.srca1g7xu33v)

[**CHƯƠNG 8. FILE EXPLORER 39**](#_heading=h.o4dqb67drd47)

[8.1 Khái niệm file 39](#_heading=h.u7t6seknytnq)

[8.2. Các thao tác cơ bản: 40](#_heading=h.jodqrp5bce5p)

[8.2.1. Tạo một file, thư mục 40](#_heading=h.fs966sw3apyc)

[8.2.2. Xóa một file, thư mục 41](#_heading=h.jymlb8ecxmir)

[8.2.3 Đổi tên một file, thư mục 41](#_heading=h.jujimy1tcdd4)

[8.2.4 Copy, cut một file, thư mục 42](#_heading=h.6ek635nj63lx)

[**CHƯƠNG 9. QUẢN LÝ BỘ NHỚ NGOÀI TRONG HỆ ĐIỀU HÀNH WINDOWS 43**](#_heading=h.sxl74b7mnc6)

[9.1. Phương pháp quản lý không gian nhớ tự do 43](#_heading=h.19dksgclab9z)

[9.2. Phương pháp cấp phát không gian nhớ tự do 48](#_heading=h.ewh4r445y0nd)

[9.3. Lập lịch cho đĩa 52](#_heading=h.19cxuwnr9jmo)

[9.4. Một số hệ thống file trong windows 54](#_heading=h.pjxuu1c5b95j)

[**TÀI LIỆU THAM KHẢO 57**](#_heading=h.xuk2hfau3i6a)

# 

# CHƯƠNG 1. MỞ ĐẦU

## 1.1. Hệ điều hành Windows

### 1.1.1. Hệ điều hành Window là gì?

Hệ điều hành Window là một hệ điều hành đồ họa được phát triển bởi Microsoft. Window cho phép người dùng xem, lưu trữ các tập tin. Hoặc chạy phần mềm, chơi trò chơi, xem video. Đồng thời, nó cũng cung cấp một cách kết nối với internet. Hệ điều hành này được phát hành cho cả máy tính cá nhân và hệ thống PC phục vụ các công việc chuyên nghiệp.

Trong các hệ điều hành được sử dụng hiện nay, Window là hệ điều hành được sử dụng rộng rãi nhất và mang lại hiệu quả rất cao trong công việc. Nó thực sự là một sản phẩm đột phá về công nghệ của thập niên 1990.

### 1.1.2. Lịch sử hệ điều hành Windows

Phiên bản đầu tiên của Window Microsoft, đó là phiên bản 1.0. Ra đời vào ngày 10 tháng 11 năm 1983. Người sáng lập Microsoft ‘Bill Gates’. Khi ấy, hệ điều hành Window 1.0 được phát hành cho cả máy tính gia đình và các chức năng chuyên nghiệp của Windows. Sau đó, phiên bản này cũng được phát hành trên nhiều phiên bản Windows khác cũng như phiên bản hiện tại – Windows 10.

Phiên bản thứ hai (Windows 2.0) được sản xuất vào ngày 9 tháng 12 năm 1987. Và nó được giới thiệu cùng ngày với Window 386. Nó đi kèm với các tính năng mới như: Các cửa sổ mở có thể chồng lên nhau, có thể phóng to và thu nhỏ, thay vì sử dụng ‘phóng to’ và ‘biểu tượng hóa’ tương ứng.

Hệ điều hành Window ra mắt lần đầu tiên năm 1983

Năm 1993, Microsoft cho ra đời phiên bản Window đầu tiên hướng tới doanh nghiệp, với tên gọi Windows NT 3.1. Các phiên bản tiếp theo lần lượt ra mắt là: Windows 3.5, 4/0 và Windows 2000.

Vào năm 2001 khi Windows XP được Microsoft phát hành, Microsoft đã thiết kế các phiên bản khác nhau cho môi trường cá nhân và doanh nghiệp. Phiên bản này được thiết lập dựa trên phần cứng x 86 tiêu chuẩn, tương tự như bộ xử lý Intel và AMD. Theo đó, hệ điều hành có thể chạy trên các thương hiệu phần cứng khác nhau. Chẳng hạn như máy tính Dell, HP, Dell, Sony, bao gồm cả PC sản xuất tại nhà.

### 1.1.3. Các hệ điều hành Window hiện nay

Microsoft đã sản xuất rất nhiều phiên bản Windows khác nhau, bắt đầu từ Windows XP. Các phiên bản này có cùng một hệ điều hành cốt lõi, nhưng một số phiên bản bao gồm các tính năng nâng cao với một khoản chi phí bổ sung. Chúng ta sẽ nhắc tới hai phiên bản Windows phổ biến nhất hiện nay. Đó là Windows Home và Windows Professional

#### 1.1.3.1. Windows Home

Windows Home là phiên bản cơ bản của Window. Hệ điều hành này cung cấp tất cả các chức năng cơ bản của Windows. Chẳng hạn như: duyệt web, kết nối Internet, chơi trò chơi điện tử, sử dụng phần mềm văn phòng, xem video. Hơn nữa, nó ít tốn kém chi phí hơn và được cài đặt sẵn trên hầu hết các máy tính mới.

#### 1.1.3.2. Windows Professional

Windows Professional còn được gọi với các tên khác như: Window Pro hoặc Win Pro. Đây là một phiên bản nâng cao của Windows Home. Phiên bản này rất có lợi cho những người dùng đã thành thạo các tác vụ của Window và đối với các doanh nghiệp vừa và nhỏ. Tuy nhiên, chi phí đầu tư cho Windows Professional cũng tốn kém hơn so với phiên bản Home.

Windows 10 - một phiên bản của Windows Professional

### 1.1.4. Tính năng nổi bật của hệ điều hành Window là gì?

Microsoft Windows có rất nhiều tính năng để trợ giúp người dùng trong quá trình sử dụng. Trong đó, 9 tính năng tuyệt vời nhất phải kể đến ở hệ điều này, đó là:

#### 1.1.4.1. Bảng điều khiển

Windows cung cấp tính năng “Bảng điều khiển” với nhiều công cụ để cấu hình và quản lý tài nguyên trên máy tính của người dùng. Bạn có thể thực hiện các thay đổi cài đặt cho âm thanh, video, máy in, chuột, bàn phím, kết nối mạng. Các thay đổi ngày giờ, tài khoản người dùng, tùy chọn tiết kiệm năng lượng, ứng dụng đã cài đặt cũng có thể thực hiện dễ dàng.

#### 1.1.4.2. Cortana

Phiên bản Windows 10 của Window có một tính năng mới, có tên Cortana. Tính năng này có thể chấp nhận lệnh thoại. Có thể thực hiện các tác vụ khác nhau như: tìm kiếm dữ liệu trên máy tính của bạn, có thể trả lời câu hỏi của bạn, mua hàng trực tuyến hay đặt lời nhắc và cuộc hẹn,…

Không chỉ vậy Cortana còn hoạt động giống như các dịch vụ kích hoạt bằng giọng nói khác. Chẳng hạn như: Trợ lý Google, Alexa hoặc Siri, bao gồm một lợi ích khác của việc tìm kiếm thông tin trên máy tính của bạn. Bạn có thể mở Cortana trong Windows 10, hãy nhấn phím Windows + S.

#### 1.1.4.3. File Explorer

Tính năng File Explorer của hệ điều hành Window là gì? Nó còn được gọi là Windows Explorer. Với khả năng hiển thị các tệp và thư mục của bạn trên máy tính. File Explorer cho phép người dùng duyệt dữ liệu trên ổ cứng, SSD và các ổ đĩa di động được chèn khác như ổ bút và CD. Đồng thời, bạn có thể quản lý nội dung theo yêu cầu như: tìm kiếm dữ liệu, chuyển dữ liệu, xóa, đổi tên,…

#### 1.1.4.4. Trình duyệt Internet

Trình duyệt internet rất quan trọng giúp bạn tìm kiếm bất cứ thông tin gì. Lướt web, xem các trang, mua sắm trực tuyến, chơi trò chơi, xem video,… Tính năng này được cài đặt sẵn trong Windows 10, với trình duyệt Internet Edge là trình duyệt mặc định. Giúp cho việc truy cập internet của bạn giờ đây trở nên đơn giản hơn rất nhiều.

#### 1.1.4.5. Microsoft Paint

Kể từ tháng 11 năm 1985, Microsoft Windows được cài đặt sẵn Microsoft Paint. Đây là một phần mềm sử dụng khá đơn giản. Nó cho phép người dùng tạo, xem và chỉnh sửa hình ảnh. Microsoft Paint cung cấp một số công cụ để vẽ hình ảnh, cắt, thay đổi kích thước và lưu hình ảnh với một phần mở rộng tệp khác.

#### 1.1.4.6. Thao tác vụ

Windows luôn có một thanh tác vụ đi kèm. Nó có vai trò hiển thị các chương trình đang mở. Nó cho phép người dùng truy cập vào bất kỳ chương trình cụ thể nào. Ngoài ra, hệ điều hành này còn có một khu vực thông báo ở phía bên phải. Có chức năng hiển thị ngày và giờ, pin, mạng, âm lượng và các ứng dụng chạy nền khác.

#### 1.1.4.7. Start Menu

Microsoft Windows chứa menu bắt đầu ở bên trái của thanh tác vụ. Menu này hiển thị các chương trình và các tiện ích cài đặt trên máy tính của bạn. Start Menu có thể được mở đơn giản bằng cách nhấp vào nút “Start menu”. Hoặc nhấn phím bắt đầu trên bàn phím.

#### 1.1.4.8. Trình quản lý tác vụ

Trình quản lý tác vụ trong hệ điều hành Window là gì? Trình quản lý tác vụ có vai trò cung cấp thông tin chi tiết về các ứng dụng hoặc chương trình đang chạy trên máy tính. Bạn cũng có thể kiểm tra xem mỗi ứng dụng đang sử dụng bao nhiêu tài nguyên hệ thống. Chẳng hạn như RAM, CPU, ổ đĩa I / O.

#### 1.1.4.9. Disk Cleanup

Tính năng này được sử dụng để giải phóng dung lượng ổ đĩa với sự trợ giúp của việc xóa các tệp tạm thời hoặc không cần thiết. Đồng thời nâng cao hiệu suất làm việc của máy tính. Disk Cleanup cũng giúp tăng dung lượng lưu trữ, cho phép tải xuống các chương trình và tài liệu.

## 1.2. Tổng quan về quản lý bộ nhớ ngoài

Khi cần lưu trữ các chương trình, dữ liệu để sử dụng trong thời gian lâu dài

thì bắt buộc cá hệ thống phải sử dụng đến bộ nhớ ngoài (đĩa từ, băng từ, CD,

DVD…). Để cho hệ thống làm việc có hiệu quả và ổn định thì cần thiết phải có cơ

chế quản lý thích hợp để đảm bảo cho quá trình vận hành máy không xảy ra xung đột, ảnh hưởng đến hiệu năng của máy. Muốn làm được điều này thì các hệ điều hành cần đảm bảo được các chức năng sau:

* Quản lý không gian nhớ tự do trên bộ nhớ ngoài (Free space Manage)
* Cấp phát không gian nhớ tự do (Allocation Methods)
* Cung cấp các khả năng định vị bộ nhớ ngoài
* Cơ chế lập lịch cho bộ nhớ ngoài

# CHƯƠNG 2. CẤU TRÚC VẬT LÝ - THƯ MỤC THIẾT BỊ

## 2.1. Cấu trúc vật lý

### 2.1.1. Đĩa từ (Platter)

· Đĩa từ chỉ sử dụng được sau khi đã định dạng, đó là việc tổ chức, sắp xếp các vùng lưu trữ thông tin trên đĩa.

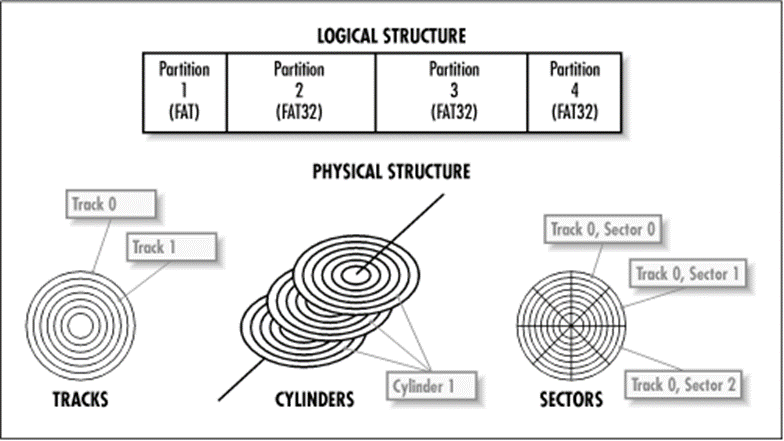
· Về mặt vật lý đĩa từ được chia thành:

· Rãnh từ (track): Là các vùng vòng tròn đồng tâm, có bề dày xác định dùng để ghi từ, các rãnh cách nhau bởi vành hẹp không được từ hóa.

· Cung từ (sector): Mỗi rãnh từ được chia thành các cung (sector), mỗi sector = 512 byte, các sector được đánh số.

· Liên cung (Cluster): Một nhóm các sector liên tiếp, thường 2/4/8 sector.

Từ trụ (Cylinder): Các track có cùng bán kính tạo thành một từ trụ (Cylinder).



*Hình 1.1: Cấu trúc vật lý của đĩa từ*

### 2.1.2. Các rãnh từ (Track)

Trên một mặt làm việc của đĩa từ chia ra nhiều vòng tròn đồng tâm thành các track.

Track có thể được hiểu đơn giản giống các rãnh ghi dữ liệu giống như các [đĩa nhựa](https://vi.wikipedia.org/w/index.php?title=%C4%90%C4%A9a_nh%E1%BB%B1a&action=edit&redlink=1) (ghi âm nhạc trước đây) nhưng sự cách biệt của các rãnh ghi này không có các gờ phân biệt và chúng là các vòng tròn đồng tâm chứ không nối tiếp nhau thành dạng xoắn trôn ốc như [đĩa nhựa](https://vi.wikipedia.org/w/index.php?title=%C4%90%C4%A9a_nh%E1%BB%B1a&action=edit&redlink=1). Track trên ổ đĩa cứng không cố định từ khi sản xuất, chúng có thể thay đổi vị trí khi định dạng cấp thấp ổ đĩa (*low format* ).

Khi một ổ đĩa cứng đã hoạt động quá nhiều năm liên tục, khi kết quả kiểm tra bằng các phần mềm cho thấy xuất hiện nhiều khối hư hỏng (bad block) thì có nghĩa là phần cơ của nó đã rơ rão và làm việc không chính xác như khi mới sản xuất, lúc này thích hợp nhất là [format cấp thấp](https://vi.wikipedia.org/wiki/%E1%BB%94_%C4%91%C4%A9a_c%E1%BB%A9ng#Format_c%E1%BA%A5p_th%E1%BA%A5p) cho nó để tương thích hơn với chế độ làm việc của phần cơ.

### 2.1.3. Sector

Trên *track* chia thành những phần nhỏ bằng các đoạn hướng tâm thành các sector. Các sector là phần nhỏ cuối cùng được chia ra để chứa dữ liệu. Theo chuẩn thông thường thì một sector chứa dung lượng 512 [byte](https://vi.wikipedia.org/wiki/Byte).

Số sector trên các track là khác nhau từ phần rìa đĩa vào đến vùng tâm đĩa, các ổ đĩa cứng đều chia ra hơn 10 vùng mà trong mỗi vùng có số sector/track bằng nhau.

Bảng sau cho thấy các khu vực với các thông số khác nhau và sự ảnh hưởng của chúng đến tốc độ truyền dữ liệu của ổ cứng Các khu vực ghi dữ liệu của ổ đĩa cứng Hitachi Travelstar 7K60 2,5".

### 2.1.4. Cylinder

Tập hợp các track cùng bán kính ở các mặt đĩa khác nhau tạo thành cylinder (trụ). Trên đĩa hai mặt, một cylinder sẽ bao gồm rãnh 1 của mặt trên trên và rãnh 1 của mặt dưới. Trên các đĩa cứng sắp xếp cái này chồng lên cái kia, một cylinder gồm các rãnh trên cả hai mặt của tất cả các đĩa. Trên một ổ đĩa cứng có nhiều cylinder bởi có nhiều track trên mỗi mặt của đĩa từ.

### 2.1.5. Đầu đọc/ghi (Read Write Heads)

Đầu đọc đơn giản được cấu tạo gồm lõi [ferit](https://vi.wikipedia.org/wiki/Ferit) (trước đây là lõi [sắt](https://vi.wikipedia.org/wiki/S%E1%BA%AFt)) và cuộn dây (giống như [nam châm](https://vi.wikipedia.org/wiki/Nam_ch%C3%A2m) điện). Gần đây các [công nghệ](https://vi.wikipedia.org/wiki/C%C3%B4ng_ngh%E1%BB%87) mới hơn giúp cho ổ đĩa cứng hoạt động với mật độ xít chặt hơn như: chuyển các hạt từ sắp xếp theo phương [vuông góc](https://vi.wikipedia.org/wiki/Vu%C3%B4ng_g%C3%B3c) với bề mặt đĩa nên các đầu đọc được thiết kế nhỏ gọn và phát triển theo các ứng dụng công nghệ mới.

Đầu đọc trong đĩa cứng có công dụng đọc [dữ liệu](https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u) dưới dạng từ hoá trên bề mặt đĩa từ hoặc từ hoá lên các mặt đĩa khi ghi dữ liệu.

Số đầu đọc ghi luôn bằng số mặt hoạt động được của các đĩa cứng, có nghĩa chúng nhỏ hơn hoặc bằng hai lần số đĩa (nhỏ hơn trong trường hợp ví dụ hai đĩa nhưng chỉ sử dụng 3 mặt).

### 2.1.6. Cần di chuyển đầu đọc/ghi (Disk controller)

Cần di chuyển đầu đọc/ghi là các thiết bị mà đầu đọc/ghi gắn vào nó. Cần có nhiệm vụ di chuyển theo phương song song với các đĩa từ ở một khoảng cách nhất định, dịch chuyển và định vị chính xác đầu đọc tại các vị trí từ mép đĩa đến vùng phía trong của đĩa (phía trục quay).

Các cần di chuyển đầu đọc được di chuyển đồng thời với nhau do chúng được gắn chung trên một trục quay (đồng trục), có nghĩa rằng khi việc đọc/ghi [dữ liệu](https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u) trên bề mặt (trên và dưới nếu là loại hai mặt) ở một vị trí nào thì chúng cũng hoạt động cùng vị trí tương ứng ở các bề mặt đĩa còn lại.

Sự di chuyển cần có thể thực hiện theo hai phương thức:

* Sử dụng [động cơ bước](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%99ng_c%C6%A1_b%C6%B0%E1%BB%9Bc) để truyền [chuyển động](https://vi.wikipedia.org/wiki/Chuy%E1%BB%83n_%C4%91%E1%BB%99ng).
* Sử dụng [cuộn cảm](https://vi.wikipedia.org/wiki/Cu%E1%BB%99n_c%E1%BA%A3m) để di chuyển cần bằng lực từ.

## 2.2. Thư mục thiết bị

Trên mỗi đĩa thường có một thư mục thiết bị (device Directory) cho biết đĩa

gồm những thông tin gì, độ dài, kiểu, người sở hữu, thời điểm khởi, vị trí, được

phân bổ không gian như thế nào. Thư mục thiết bị nằm trong vùng nhớ đặc biệt.

# CHƯƠNG 3 :CÁC DẠNG LƯU TRỮ DỮ LIỆU TRÊN HĐH WINDOWS

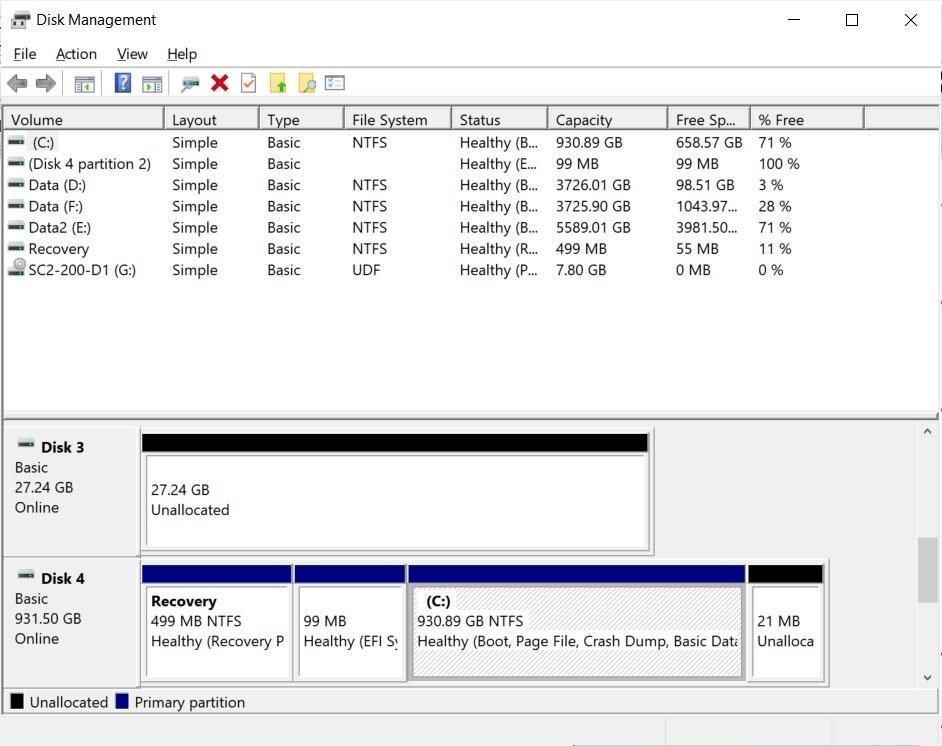
Hệ điều hành Windows hỗ trợ hai loại đĩa lưu trữ chủ yếu là *Basic* (cơ bản) và *Dynamic* (động).

## 3.1. Lưu trữ cơ bản ( Basic storage).

Gồm các phân vùng cơ bản (Partition Primary), hay còn gọi là phân vùng chính, và phân vùng mở rộng (Extended Partition). Phân vùng tạo ra đầu tiên trên đĩa được gọi là phân vùng chính và toàn bộ không gian cấp cho phân vùng sẽ được sử dụng trọn vẹn. Mỗi ổ đĩa vật lý có thể tạo tối đa bốn phân vùng chính hoặc ba phân vùng chính và một phân vùng mở rộng . Với phân vùng mở rộng, ta có thể tạo ra tùy ý số phân vùng logic khác.

Trên ổ cứng có một vùng nhỏ dùng để ghi bảng phân vùng ổ đĩa (Disk partition table). Đây là nơi hệ điều hành sẽ đọc để theo dõi cách thức phân chia đang tồn tại trên ổ đĩa. Bảng phân vùng ổ đĩa có độ lớn 64 byte chia làm 4 mục, các thông tin về mỗi phân vùng chính được ghi trên một mục chiếm 16 byte, ổ cứng vật lý chỉ có thể chia làm 4 phân vùng cũng là lý do đó. Tại 1 thời điểm chỉ có một phân vùng được nhận quyền khởi động, đó là phân vùng chứa hệ điều hành dùng để khởi động máy.

Một Basic storage có hai cách quản lý phân vùng là MBR disks (Master Boot Record) và GPT disks (GUIDs Partition Table).Đối với MBR disks, chúng ta có thể tạo được nhiều nhất 4 phần vùng chính (Primary Partition), hoặc 3 phân vùng chính và một phân vùng mở rộng (Extended Partion) và trong phân vùng mở rộng (Extended Partion) ta có thể tạo vô hạn các ổ đĩa luận lý (Logical Drive).Đối với GPT disks, chúng ta có thể tạo lên đến 128 phân vùng chính (Primary Partition). Bởi vì GPT disks không giới hạn 4 phân vùng chính nên chúng ta không cần tạo phân vùng mở rộng (Extended Partion) hay các ổ đĩa luận lý (Logical Drive)



*Hình 3.1. Quản lý đĩa cứng trên Windows*

## 3.2. Lưu trữ động (Dynamic storage)

Đĩa lưu trữ động được chia thành các phân vùng động. Phân vùng động không chứa phân vùng hoặc ổ đĩa logic và chỉ có thế truy cập được trên hệ điều hành Windows server 2003 và Windows 2000. Windows server 2003 và Windows 2000 hỗ trợ 5 loại phân vùng động là spanned, simple, striped, mirrored và RAID-5

* + Ưu điểm của công nghệ lưu động so với công nghệ lưu trữ căn bản là:
    - * Cho phép nhiều ổ đĩa vật lý để tạo thành các ổ đĩa logic (Volume).
      * Cho phép ghép nhiều vùng trống không liên tục trên nhiều đĩa cứng vật lý để tạo ổ đĩa logic.
      * Có thể tạo ra các ổ đĩa logic có khả năng dung lỗi cao và tắng tốc truy xuất
  + Nhược điểm của đĩa lưu trữ động là chỉ được hỗ trợ bởi một số hệ điều hành của Microsoft, ngoài ra nó đòi hỏi phải có từ hai ổ cứng trở lên, trong một số trường hợp các ổ cứng phải giống hệt nhau về kích thước… do đó đĩa lưu trữ động chủ yếu chỉ sử dụng trong các hệ thống lớn, trong các máy chủ, hoặc trong các nơi có đòi hỏi cấp thiết về vấn đề an toàn và tốc độ xử lý dữ liệu

### 3.2.1. Spanned volume

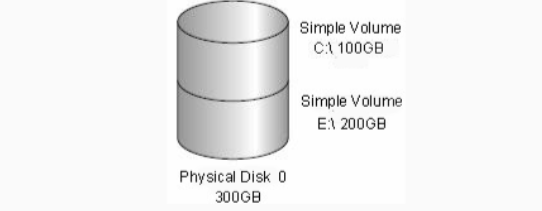
Bao gồm 1 hoặc nhiều đĩa lưu trữ động ( tối đa là 32 đĩa ). Sử dụng khi người dùng muốn tăng kích thước phân vùng. Dữ liệu được ghi lên vùng theo thứ tự và hết đĩa này đến đĩa khác. Thông thường người quản trị sử dụng phân vùng *spanned* khi ổ đĩa đang sử dụng trong phân vùng sắp bị đầy và muốn tăng kích thước của phân vùng bằng cách bổ sung thêm một đĩa khác.

Do dữ liệu được ghi tuần tự nên phân vùng loại này không tăng hiệu năng sử dụng. Nhược điểm chính của phân vùng *spanned* là nếu một đĩa bị hỏng thì toàn bộ dữ liệu trên phân vùng sẽ không thể truy xuất được. Dữ liệu trên *spanned volume* được chép phân bổ trên 2 ổ cứng vật lý trở lên, các ổ cứng không nhất thiết phải giống nhau và chúng được ghép lại thành một Volume duy nhất, tuy vậy chúng không có khả năng đáp ứng vấn đề an toàn dữ liệu (Fault Tolerangcing), và tốc độ xử lý dữ liệu (Load Balancing), vì chưa có sự thay đổi về cơ chế (dữ liệu được chép đầy trên span volume ở disk 1 mới chép sang các disk còn lại).

### 3.2.2. Simple volume

Đây là một “phân vùng” thông thường. Nếu loại đĩa là “Cơ bản”, điều này sẽ tạo ra một phân vùng thực.Phân vùng *simple* không gian lấy từ một đĩa dynamic duy nhất. Không gian đĩa này có thể liên tục hoặc không liên tục. Hình sau minh hoạ một đĩa vật lý được chia thành hai volume đơn giản.

Dữ liệu trên simple volume chỉ được lưu trữ trên 1 ổ cứng vật lý, do đó vấn đề an toàn dữ liệu (Fault Tolerancing), và tăng tốc độ xử lý (Load Balancing) không được đảm bảo, khi ổ cứng vật lý hỏng, thì dữ liệu có nguy cơ bị mất.

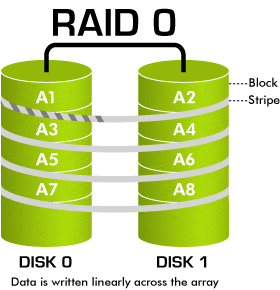


*Hình 3.2. Một đĩa vật lý được chia thành hai phân vùng đơn giản*

### 3.2.3 Striped Volume (RAID-0)

Lưu trữ dữ liệu lên các dãy (strip) bằng nhau trên một hoặc nhiều đĩa vật lý (tối đa là 32). Do dữ liệu được ghi tuần tự lên từng dãy, nên bạn có thể thi hành nhiều tác vụ I/O đồng thời, làm tăng tốc độ truy xuất dữ liệu. Thông thường, người quản trị mạng sử dụng volume striped để kết hợp dung lượng của nhiều ổ đĩa vật lý thành một đĩa logic đồng thời tăng tốc độ truy xuất.

Dữ liệu trên striped volume có thể được trao đổi cùng lúc trên 2 ổ cứng vật lý trở lên, dung lượng trên các ổ cứng vật lý của striped volume phải bằng nhau. Striped Volume có sự thay đổi trong cơ chế hoạt động, dữ liệu khi được chép trên striped được chia ra và chép đều trên các disk, vì thế striped đáp ứng được vấn đề tốc độ xử lý dữ liệu (Load Balancing), tuy nhiên striped không đáp ứng được vấn đề an toàn dữ liệu (Fault Tolerangcing), nếu một ổ đĩa bị hỏng thì dữ liệu trên toàn bộ volume mất giá trị

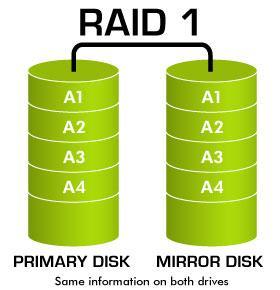


*Hình 3.3. Trao đổi dữ liệu trên các ổ cứng vật lý*

### 3.2.4 Mirror volume (RAID-1)

Là hai bản sao của một volume đơn giản. Bạn dùng một ổ đĩa chính và một ổ đĩa phụ. Dữ liệu khi ghi lên đĩa chính đồng thời cũng sẽ được ghi lên đĩa phụ. Volume dạng này cung cấp khả năng dung lỗi tốt. Nếu một đĩa bị hỏng thì ổ đĩa kia vẫn làm việc và không làm gián đoạn quá trình truy xuất dữ liệu.Nhược điểm của phương pháp này là bộ điều khiển đĩa phải ghi lần lượt lên hai đĩa, làm giảm hiệu năng

Để tăng tốc độ ghi đồng thời cũng tăng khả năng dung lỗi, bạn có thể sử dụng một biến thể của volume mirrored là duplexing. Theo cách này bạn phải sử dụng một bộ điều khiển đĩa khác cho ổ đĩa thứ hai.

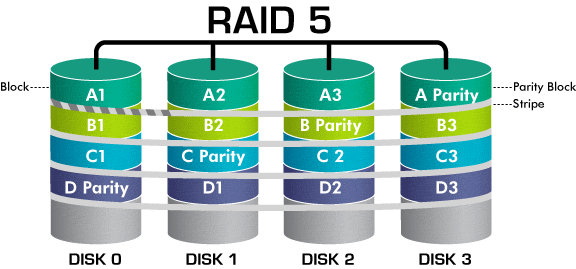


*Hình 3.4. Dữ liệu khi ghi lên đĩa chính đồng thời cũng sẽ được ghi lên đĩa phụ*

Nhược điểm chính của phương pháp này là chi phí cao. Để có một volume 4GB bạn phải tốn đến 8GB cho hai ổ đĩa.

### 3.2.5. Volume RAID-5g

Raid-5 Volume là giải pháp kết hợp các loại volume (Striped Volume RAID-0, Mirror Volume RAID-1)mà ta đã đề cập ở trên. Raid-5 đáp ứng cho chúng ta cả 2 vấn đề an toàn dữ liệu (Fault Tolerangcing), và tăng tốc độ xử lý dữ liệu (Load Balancing). Để đáp ứng 2 vấn đề trên, Raid-5 đòi hỏi phải sử dụng 3 ổ đĩa cứng vật lý, và sử dụng thuật toán Parity (khi 1 trong 3 đĩa bị hỏng, thuật toán Parity sẽ tự chép những bit bị mất). Vì phải chứa thêm bit Parity nên dung lượng của Raid-5 Volume sẽ chỉ bằng 2/3 dung lượng ta cấu hình (1/3 còn lại là để chứa bit Parity). Ưu điểm chính của kỹ thuật này là khả năng dung lỗi cao và tốc độ truy xuất cao bởi sử dụng nhiều kênh I/O.



*Hình 3.5. Hoạt động của Raid-5*

Để setup Raid 5 ta cần tối thiểu 3 ổ cứng. Theo như hình minh họa phương án lưu trữ của Raid 5 như sau. Giả sử có 1 file A thì khi lưu trữ sẽ tách ra 3 phần A1, A2, A3. Ba phần nãy sẽ tương ứng lưu trên ổ đĩa Disk 0, Disk 1, Disk 2, còn ổ đĩa Disk 3 sẽ giữ bản sao lưu backup của 3 phần này. Tương tự các file sau cũng vậy và tùy theo tiến trình thực hiện mà bản sao lưu có thể được lưu ở bất kì 1 trong những ổ trong cụm Raid.

# 

# CHƯƠNG 4. CHƯƠNG TRÌNH QUẢN LÝ BỘ NHỚ NGOÀI DISK MANAGER

## 4.1. Thuộc tính của đĩa

Cấu trúc vật lý

Một đĩa từ gồm một hay nhiều lá đĩa xếp chồng lên nhau

- Mỗi mặt của đĩa được chia thành nhiều rãnh tròn đồng tâm gọi là Track

- Mỗi Track được chia thành các cung tròn bằng nhau gọi là Sector ( Cluster)

- Các Track có cùng số hiệu được gọi là từ trụ (Cylinder)

- Mỗi mặt của đĩa có một đầu từ để đọc/ghi dữ liệu trên mặt đĩa đó, để điều khiển đầu đọc/ghi dữ liệu cần có một trình điều khiển đĩa.

- Thông tin trên đĩa từ bị tham chiếu bởi các thành phần:

+ ổ đĩa (Drive)

+ Mặt đĩa (Side)

+ Rãnh đĩa (Track)

+ Cung tròn trên track (Sector) - Cluster

OS xem đĩa như là một mảng 1 chiều mà thành phần là các khối đĩa (Disk Block). Mỗi khối đĩa ghi các thông tin về Cylinder, track, Sector mà OS có thể định vị trên đĩa đó.

## 4.2. Thuộc tính của phân vùng hoặc đĩa cục bộ

Thư mục thiết bị

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Master Boot | Boot Sector | FAT or None FAT | ROOT Directory | Database |
| Gồm 2 thành phần là Master boot Record và Partition | Là Sector đầu tiên của ổ đĩa có nhiệm vụ khởi động | Bảng định vị và quản lý dữ liệu trên đĩa | Cấu trúc thư mục để tổ chức dữ liệu trên đĩa | Vùng lưu trữ CSDL của người dùng |

Trên mỗi đĩa thông thường có một thư mục thiết bị cho biết đĩa gồm những thông tin gì, độ dài, kiểu, người sở hữu, thời điểm khởi tạo, vị trí, vùng không gian nhớ được cấp phát như thế nào

Thư mục thiết bị được tạo ngay ở trên đĩa tại một vùng nhớ đặc biệt.

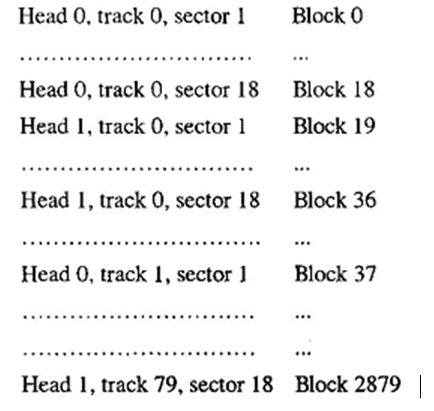
# CHƯƠNG 5. QUẢN LÝ KHÔNG GIAN NHỚ TỰ DO TRONG HỆ ĐIỀU HÀNH

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 sáo cho hiệu quả tối ưu

## 5.1. Quản lý bộ nhớ bằng phương pháp dùng bit vector (Bitmap)

Không gian đĩa được chia thành các khối (block) và được đánh số từ 0…max.

Ví dụ: đĩa mềm 1.44Mb, 2 mặt, 80 track/1 mặt, 18 sector/1 track được đánh số như sau:



Mỗi khối đĩa sử dụng 1 bit để đánh dấu trạng thái. Khối đĩa nào sử dụng thì bit trạng thái có giá trị bằng một, chưa sử dụng thì có giá trị bằng 0. Tập hợp các kí hiệu 0,1 tạo thành một bit vector (bitmap). Đọc thông tin trong bitmap hệ điều hành có thể xác định không gian tự do trên đĩa.

Ví dụ: cho không gian đĩa như hình 4.1, các khối 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, 27 là các khối đĩa tự do. Khi đó bitmap quản lý không gian nhớ tự do như sau:

11000011000000111001111110001111…

Phương pháp bitmap có ưu điểm là cài đặt đơn giản, dễ quản lý, dễ tìm kiếm những khối đĩa tự do liên tục trên đĩa nhưng tốn không gian lưu trữ dành cho bitmap (mỗi khối đĩa sẽ là một bit).

## 5.2. Quản lý bộ nhớ bằng phương pháp liệt kê( free list)

Trong phương pháp này hệ thống sử dụng một danh sách móc nối để liệt kê các Block tự do. Con trỏ đầu trong danh sách chỉ tới block đĩa tự do đầu tiên, mỗi block đĩa có một con trỏ để trỏ tới khối kế tiếp.

Ưu điểm : Tiết kiệm không gian đĩa

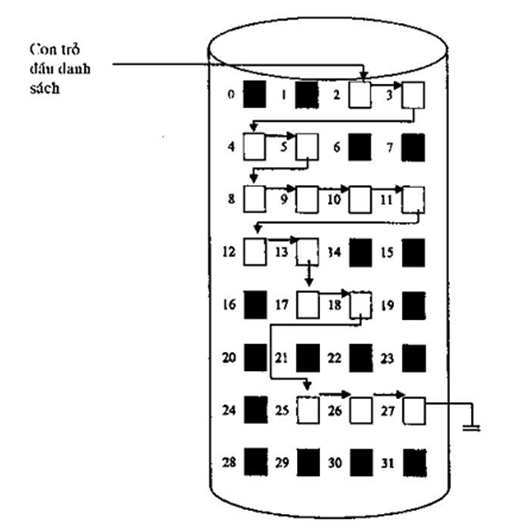
Nhược điểm : Tăng thời gian truy nhập trên đĩa

Ví dụ : Giả sử Block 3, 7, 9 tự do có thể phân bổ được

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

## 5.3. Quản lý bộ nhớ bằng phương pháp lập nhóm(grouping)

Nhóm các Block tự do liên tiếp thành một nhóm, block tự do đầu tiên trong nhóm ghi địa chỉ của các block còn lại trong nhóm, block tự do cuối cùng trong nhóm ghi địa chỉ của block tự do đầu tiên của nhóm tiếp theo.



*Hình 5.2 Mô tả không gian đĩa từ*

Ví dụ: Theo hình 5.2, ta có bảng quản lý không gian nhớ tự do như sau:

|  |  |  |
| --- | --- | --- |
| Nhóm | Block đầu | Block cuối |
| 1 | 2(2,3,4,5) | 5(8) |
| 2 | 8(8,9,10,11,12,13) | 13(17) |
| 3 | 17(17,18) | 18(25) |
| 4 | 25(25,26,27) | 27(...) |

## 5.4. Quản lý bộ nhớ bằng phương pháp đếm (counting)

Phương pháp đếm là sự biến đổi của phương pháp lập nhóm. Trong phương pháp này hệ thống lập danh sách quản lý địa chỉ của các block đĩa tự do đầu tiên và số lượng các block tự do liên tục kế tiếp các block đĩa đó.

Ví dụ: Theo hình trên ta có danh sách quản lý không gian nhớ tự do như sau:

|  |  |
| --- | --- |
| Danh sách | Số lượng |
| 2 | 4 |
| 8 | 6 |
| 17 | 2 |
| 25 | 3 |

# CHƯƠNG 6: CÁC PHƯƠNG PHÁP CẤP PHÁT KHÔNG GIAN NHỚ TỰ DO

## 6.1. Cấp phát liên tục (Contiguous)

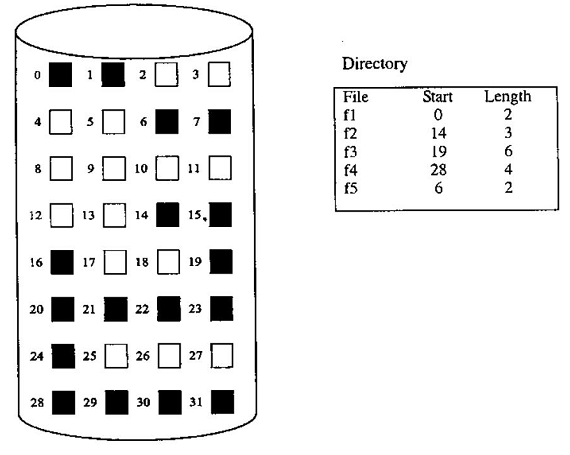
Để phân bố không gian nhớ cho một file, hệ thống chọn một đoạn liên tục các khối đĩa tự do để cấp phát cho file đó. Với phương pháp này, để định vị file hệ thống chỉ cần biết địa chỉ của khối đĩa tự do đầu tiên và số lượng block đã dùng.

Ưu điểm của cấp phát liên tục là hỗ trợ phương pháp truy nhập tuần tự và truy nhập trực tiếp, nhưng tồn tại 3 nhược điểm chính:

- Phải chọn được thuật toán tối ưu để tìm các vùng không gian tự do cấp phát cho file (First – Fit, Best – Fit hoặc Worst – Fit)

- Có thể xảy ra trường hợp không đủ số khối đĩa tự do liên tiếp cần thiết để cấp phát cho file ( kích thước file lớn hơn vùng các khối đĩa liên tục lớn nhất).

- Trường hợp các khối đĩa tự do nằm tản mạn sẽ không được sử dụng, sẽ gây lãng phí không gian nhớ.

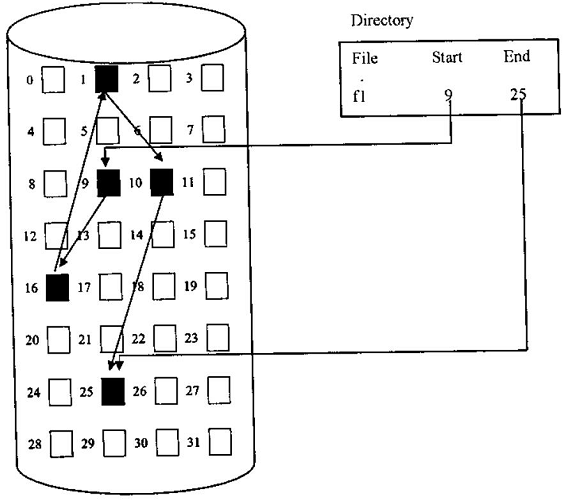


*Hình 6.1. không gian đĩa được cấp phát kề*

## 6.2. Cấp phát liên kết (Linked)

Trong phương pháp này, mỗi file được định vị trong thư mục thiết bị bằng 2 con trỏ, một cái trỏ tới khối đĩa đầu tiên, một cái trỏ tới khối đĩa cuối cùng đã cấp phát cho file. Trong mỗi khối đĩa đã cấp phát cũng có một con trỏ để trỏ tới khối đĩa kế tiếp.

Ví dụ: file f1 được cấp phát 5 khối đĩa có số hiệu 9, 16, 1, 11, 25; khối đầu là 9, khối cuối là 25.

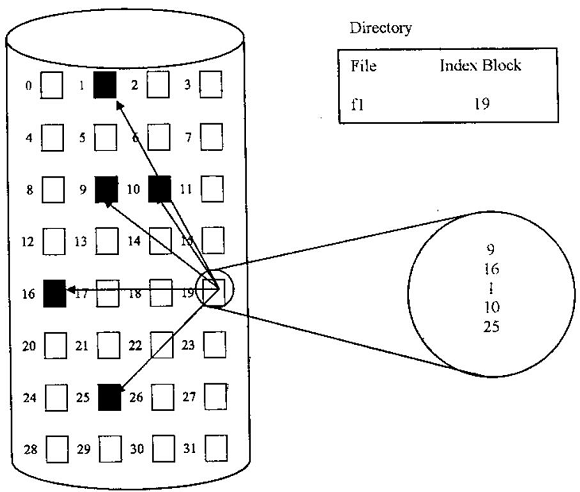


*Hình 6.2. cấp phát không gian đĩa liên kết*

Cấp phát liên kết có ưu điểm là sử dụng được các khối đĩa tự do nằm tản mạn nhưng chỉ hỗ trợ truy nhập tuần tự, không hỗ trợ truy nhập trực tiếp, độ tin cậy không đảm bảo nếu mất các con trỏ liên kết. Mặt khác phương pháp này tốn không gian nhớ để lưu trữ các con trỏ ( khoảng 0,38% không gian đĩa).

## 6.3. Cấp phát theo chỉ số (Index)

Phương pháp này, để cấp phát không gian nhớ cho một file, hệ thống sử dụng một khối đĩa đặc biệt gọi là khối đĩa chỉ số (Index block) cho mỗi file. Trong khối đĩa chỉ số chứa địa chỉ các khối đĩa đã cấp phát cho file, trong thu mục thiết bị địa chỉ của các khối đĩa chỉ số. Khi một khối đĩa được cấp phát cho file thì hệ thống loại bỏ địa chỉ của khối đĩa này khỏi danh sách của cấc khối đĩa tự do và cập nhật vào khối chỉ số của file.



*Hình 6.3. cấp phát không gian đĩa theo chỉ số*

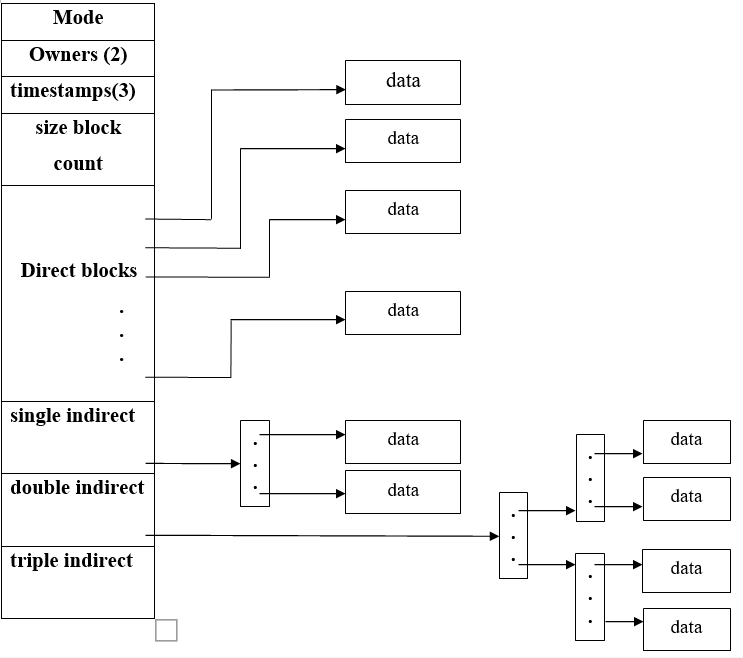
Phương pháp cấp phát theo chỉ số hỗ trợ truy nhập trực tiếp nhưng lãng phí không gian nhớ dành cho khối đĩa chỉ số.

Điểm này sinh ra câu hỏi: Khối chỉ số nên lớn bao nhiêu? Tuy nhiên, nếu khối chỉ số quá nhỏ nó không thể quản lý đủ các con trỏ cho một tập tin lớn, cần có một cơ chế giải quyết vấn đề này:

+ *Cơ chế liên kết (linked scheme):* một khối chỉ số thường là một đĩa. Do đó, nó có thể được đọc và viết trực tiếp bởi chính nó. Để cho phép đối với các tập tin lớn, chúng ta có thể liên kết nhiều khối chỉ số với nhau. Thí dụ, một khối chỉ số có thể chứa một header nhỏ cho tên tập tin và một tập hợp của các địa chỉ 100 khối đĩa đầu tiên. Địa chỉ tiếp theo (từ cuối cùng trong khối chỉ số ) là nil (đối với một tập tin nhỏ ) hay một con trỏ tới khối chỉ số khác (cho một tập tin lớn).

+ *Chỉ số nhiều cấp (multilevel index):* một biến dạng của biểu diễn liên kết là dùng khối chỉ số cấp 1 để chỉ tới khối chỉ số cấp 2. Khối chỉ số cấp 2 chỉ tới các khối tập tin. Để truy xuất một khối, hệ điều hành dùng chỉ số cấp 1 để tìm một khối chỉ số cấp 2 và khối đó tìm khối dữ liệu mong muốn. Tiếp cận này có thể được tiếp tục tới cấp 3 hay cấp 4, tùy thuộc vào kích thước tập tin lớn nhất được mong muốn. Với khối có kích thước 4,096 bytes, chúng ta có thể lưu 1,024 con trỏ 4 bytes trong một khối chỉ số. Chỉ số hai cấp cho phép 1,048,576 khối dữ liệu, cho phép tập tin có kích thước tới 4GB.

+ *Cơ chế kết hợp (combined scheme):* một biến dạng khác được dùng trong UFS là giữ 15 con trỏ đầu tiên của khối chỉ số trong inode của tập tin. 12 con trỏ đầu tiên của 15 con trỏ này chỉ tới khối trực tiếp (direct blocks), nghĩa là chúng chứa các địa chỉ của khối mà chứa dữ liệu của tập tin. Do đó, dữ liệu đối với các tập tin nhỏ (không lớn hơn 12 khối) không cần một khối chỉ số riêng. Nếu kích thước khối là 4KB, thì 48KB dữ liệu có thể truy xuất trực tiếp. 3 con trỏ tiếp theo chỉ tới các khối gián tiếp (indirect blocks). Con trỏ khối gián tiếp thứ nhất là địa chỉ của khối gián tiếp đơn (single indirect blocks). Khối gián tiếp đơn là một khối chỉ số không chứa dữ liệu nhưng chứa địa chỉ của các khối dữ liệu. Sau đó, có con trỏ khối gián tiếp đôi (double indirect block) chứa địa chỉ của một khối mà khối này chứa địa chỉ của các khối chứa con trỏ chỉ tới khối dữ liệu thật sự. Con trỏ cuối cùng chứa địa chỉ của khối gián tiếp ba (triple indirect block). Với phương pháp này, số khối có thể cấp phát tới một tập tin vượt quá hạn lượng không gian có thể đánh địa chỉ bởi các con trỏ tập tin 4 bytes hay 4 GB. Nhiều cài đặt UNIX gồm Solaris và AIX của IBM hỗ trợ tới 64 bit con trỏ tập tin. Các con trỏ có kích thước này cho phép các tập tin và hệ thống tập tin có kích thước tới terabytes.



*Hình 6.4. Inode của UNIX*

Cơ chế cấp phát lập chỉ số gặp một số vấn đề khó khăn về năng lực như cấp phát liên kết. Đặc biệt các khối chỉ số có thể được lưu trữ (cache) trong bộ nhớ,nhưng các khối dữ liệu có thể trải rộng khắp phân khu.

# CHƯƠNG 7: LẬP LỊCH CHO ĐĨA TỪ TRONG HỆ ĐIỀU HÀNH WINDOWS

## 7.1. Khái niệm về lập lịch cho đĩa

Thời gian truy nhập đĩa phụ thuộc vào ba yếu tố: thời gian di chuyển đầu từ đọc/ghi đến track hoặc cylinder cần thiết (seek-time), thời gian định vị đầu từ đọc/ghi tại khối đĩa cần truy nhập (latency-time) và thời gian truy nhập dữ liệu (transfer-time). Thời gian định vị đầu từ đọc/ghi và thời gian truy nhập dữ liệu thông thường cố định và phụ thuộc cấu trúc kỹ thuật của ổ đĩa. Do đó để tăng tốc độ truy nhập đĩa, các hệ điều hành thường quan tâm tới thời gian di chuyển đầu từ đọc/ghi.

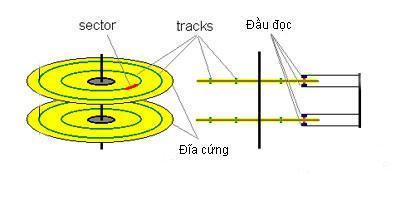
Lập lịch cho đĩa là xây dựng các thuật toán dịch chuyển đầu từ đọc ghi sao cho thời gian truy nhập đĩa là tối ưu nhất

Thời gian truy nhập đĩa

- Thời gian di chuyển đầu từ đọc ghi đến strack thích hợp(seek-time)

- Thời gian chờ cho khối cần thiết dưới đầu đọc(latency -time)

- Thời gian vận chuyển dữ liệu giữa đĩa và bộ nhớ chính(transfer-time)



Tất cả mọi công việc đều phụ thuộc vào việc nạp chương trình và nhập xuất tập tin, do đó điều quan trọng là dịch vụ đĩa phải càng nhanh càng tốt. Hệ điều hành có thể tổ chức dịch vụ truy xuất đĩa tốt hơn bằng cách lập lịch yêu cầu truy xuất đĩa.

Tốc độ đĩa bao gồm ba phần. Để truy xuất các khối trên đĩa, trước tiên phải di chuyển đầu đọc đến track hay cylinder thích hợp, thao tác này gọi là seek và thời gian để hoàn tất gọi là *seek time*. Một khi đã đến đúng track, còn phải chờ cho đến khi khối cần thiết đến dưới đầu đọc. Thời gian chờ này gọi là *latency time*. Cuối cùng là vận chuyển dữ liệu giữa đĩa và bộ nhớ chính gọi là *transfer time*. Tổng thời gian cho dịch vụ đĩa chính là tổng của ba khoảng thời gian trên. Trong đó *seek time* và *latency time* là mất nhiều thời gian nhất, do đó để giảm thiểu thời gian truy xuất hệ điều hành đưa ra các thuật toán lập lịch truy xuất.

## 7.2. Các thuật toán lập lịch cho đĩa

- First come first Served(FCFS)

- Shortest seek time first(SSTF)

- Scan

- C-Scan

- Look

- C-Look

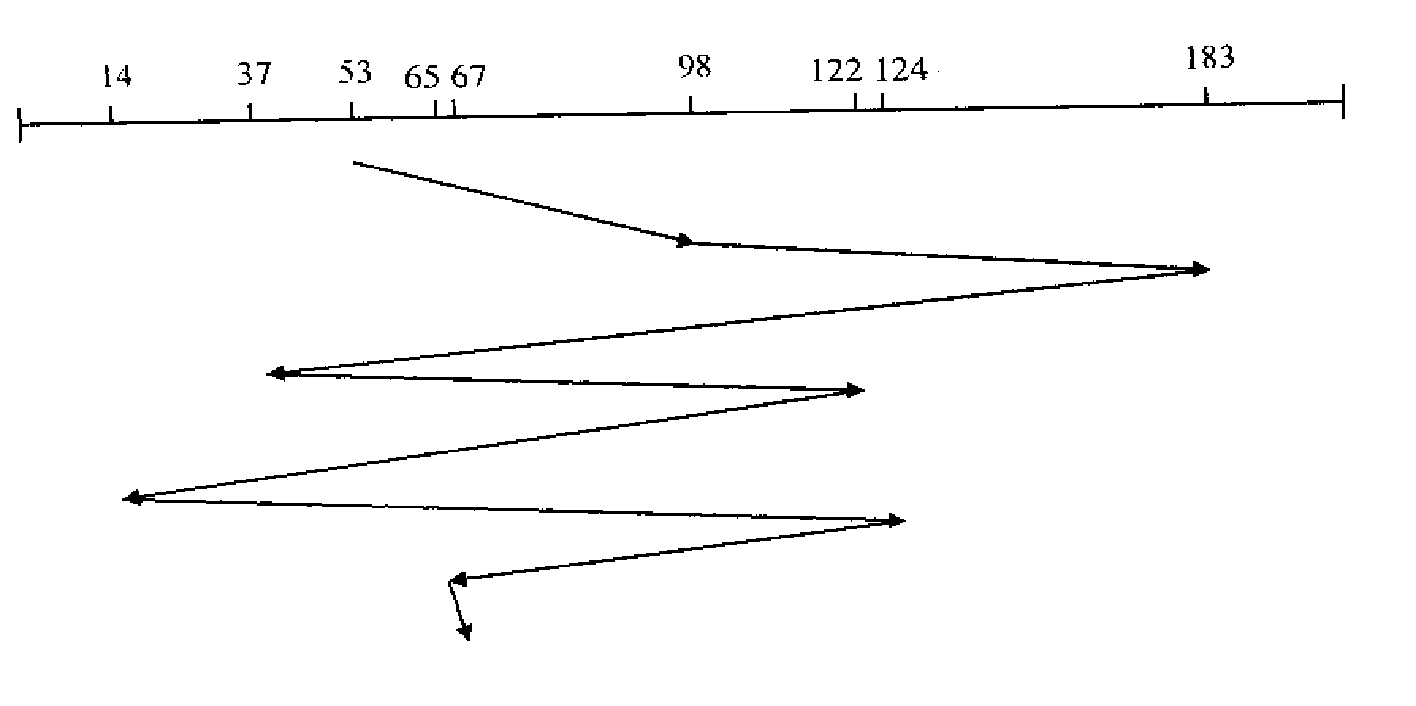
### 7.2.1.First come first Served(FCFS)

-Để truy nhập tới 1 file, hệ thống sẽ tổ chức một hàng đợi các yêu cầu phục vụ của các track(lưu trữ dữ liệu của file cần truy nhập)

-Nội dung: track nào có yêu cầu phục vụ trước thì đầu đọc ghi sẽ dịch chuyển tới đó trước

-Ví dụ:File F1 được phân bổ lần lượt các track số thứ tự :98,183,37,122,14,124,65,67  
giả sử đầu đọc đang ở track 53

-Sơ đồ dịch chuyển theo FCFS(640 bước dịch chuyển)



Ưu,nhược điểm

-Ưu điểm:

+Dễ lập trình

+Các track cần truy xuất là liên tục

-Nhược điểm

+Số track mà đầu đọc phải di chuyển là nhiều

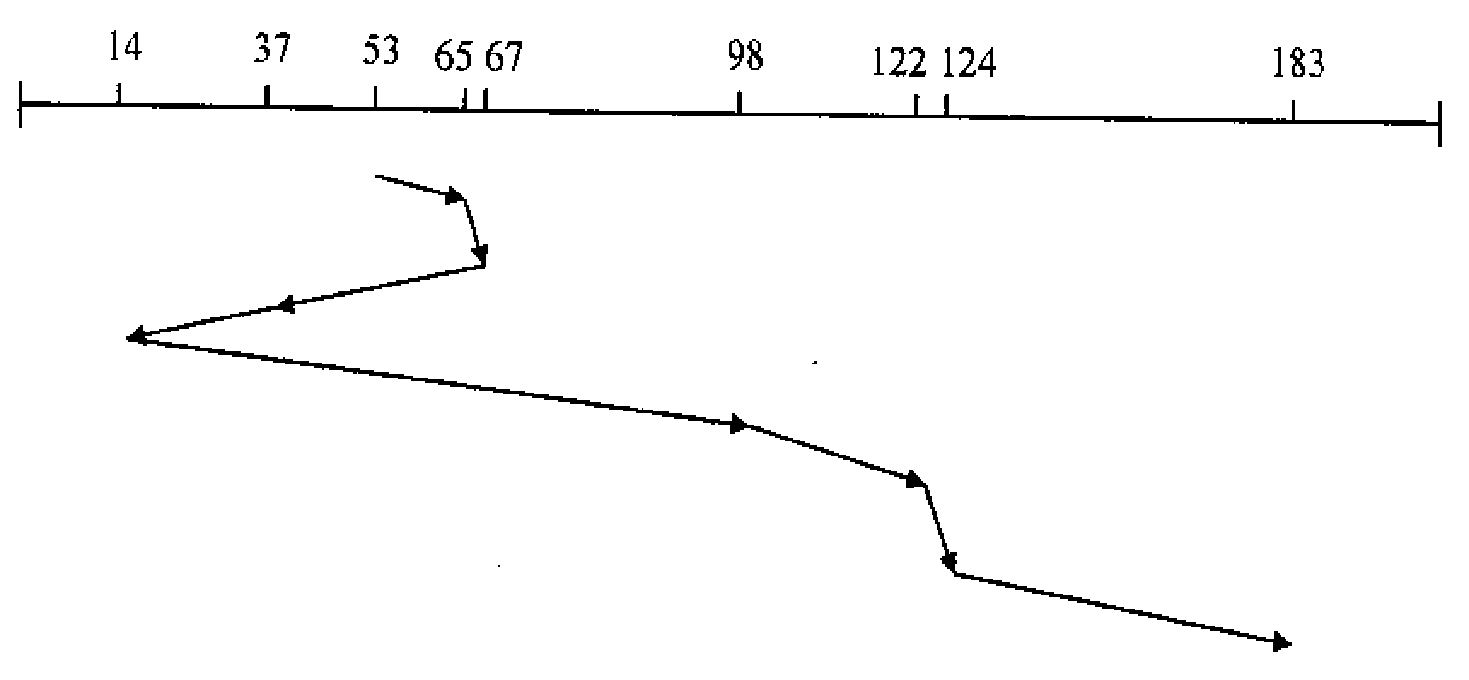
+Hiệu quả của thuật toán phụ thuộc vào thứ tự của các track trong hàng đợi

### 7.2.2. Shortest Seek time First

-Nội dung: track nào có thời gian di chuyển đầu từ đọc ghi ngắn nhất thì phục vụ trước

-Ví dụ:File F1 được phân bổ lần lượt các track số thứ tự :98,183,37,122,14,124,65,67  
giả sử đầu đọc đang ở track 53

-Sơ đồ dịch chuyển theo SSTF(Tổng quãng đường dịch chuyển là 360)



Ưu, nhược điểm SSTF

-Ưu điểm:

+Số track mà đầu đọc phải đi chuyển giảm

-Nhược điểm

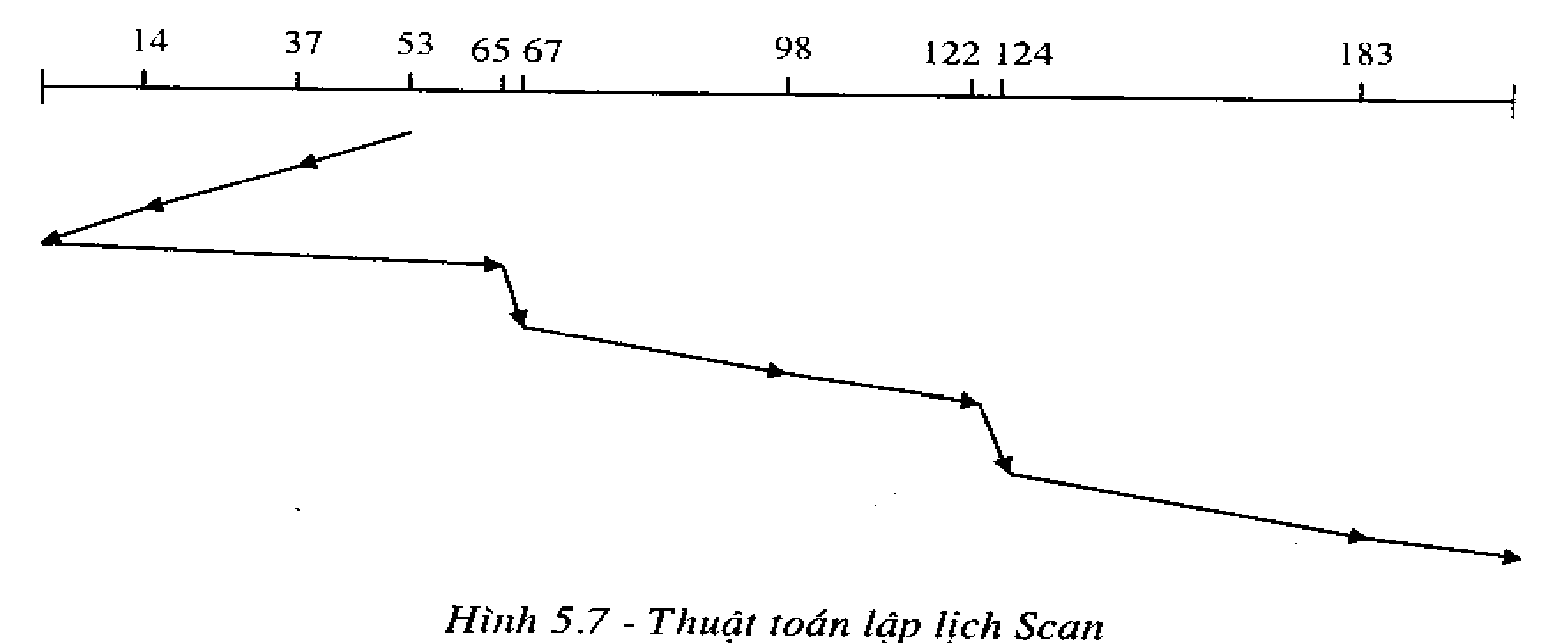
+Có thể gây ra 1 số yêu cầu không bao giờ được phục vụ

### 7.2.3. Scan

-Nội dung: Đầu đọc của đĩa di chuyển từ một phía (ví dụ bên ngoài hoặc bên trong đĩa) sang phía kia để phục vụ các yêu cầu đọc, sau đó di chuyển ngược lại... quá trình này lặp đi lặp lại

-Ví dụ:File F1 được phân bổ lần lượt các track số thứ tự :98,183,37,122,14,124,65,67  
giả sử đầu đọc đang ở track 53

-Tổng quãng đường dịch chuyển là 252



Đặc điểm:

+ Phương thức h/đ như thang máy

+ Số bước đầu đọc phải di chuyển giảm

### 7.2.4. C-Scan

-Nội dung: Đầu đọc chuyển từ một phía (trong/ngoài)sang phía kia và phục vụ các yêu cầu. Khi sang đến phía kia, đầu đọc quay trở lại nhưng trong khi quay trở lại không phục vụ yêu cầu nào.(quét 1 chiều)

-Ví dụ: File F1 được phân bổ lần lượt các track số thứ tự :98,183,37,122,14,124,65,67  
giả sử đầu đọc đang ở track 53

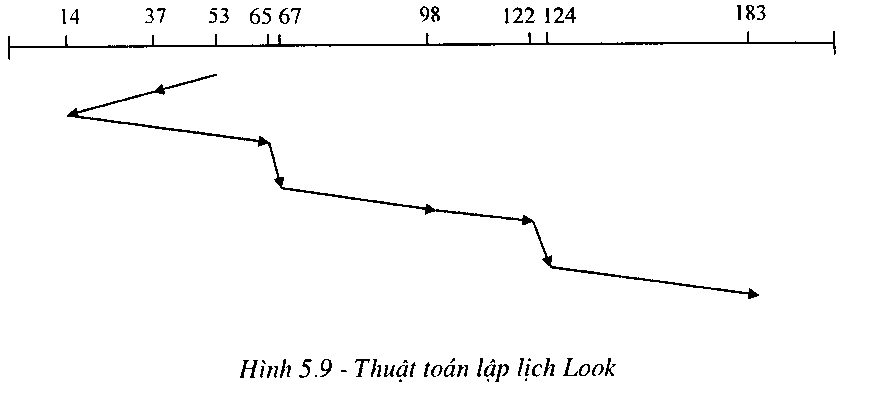
-Số bước dịch chuyển của đầu từ 252

### 7.2.5. Look

-Nội dung: tương tự như Scan nhưng trong thuật toán này đầu đọc ghi chỉ quét trong phạm vi các track có nhu cầu phục vụ không quét tới track đầu tiên hoặc cuối cùng

-Ví dụ: File F1 được phân bổ lần lượt các track số thứ tự :98,183,37,122,14,124,65,67  
giả sử đầu đọc đang ở track 53

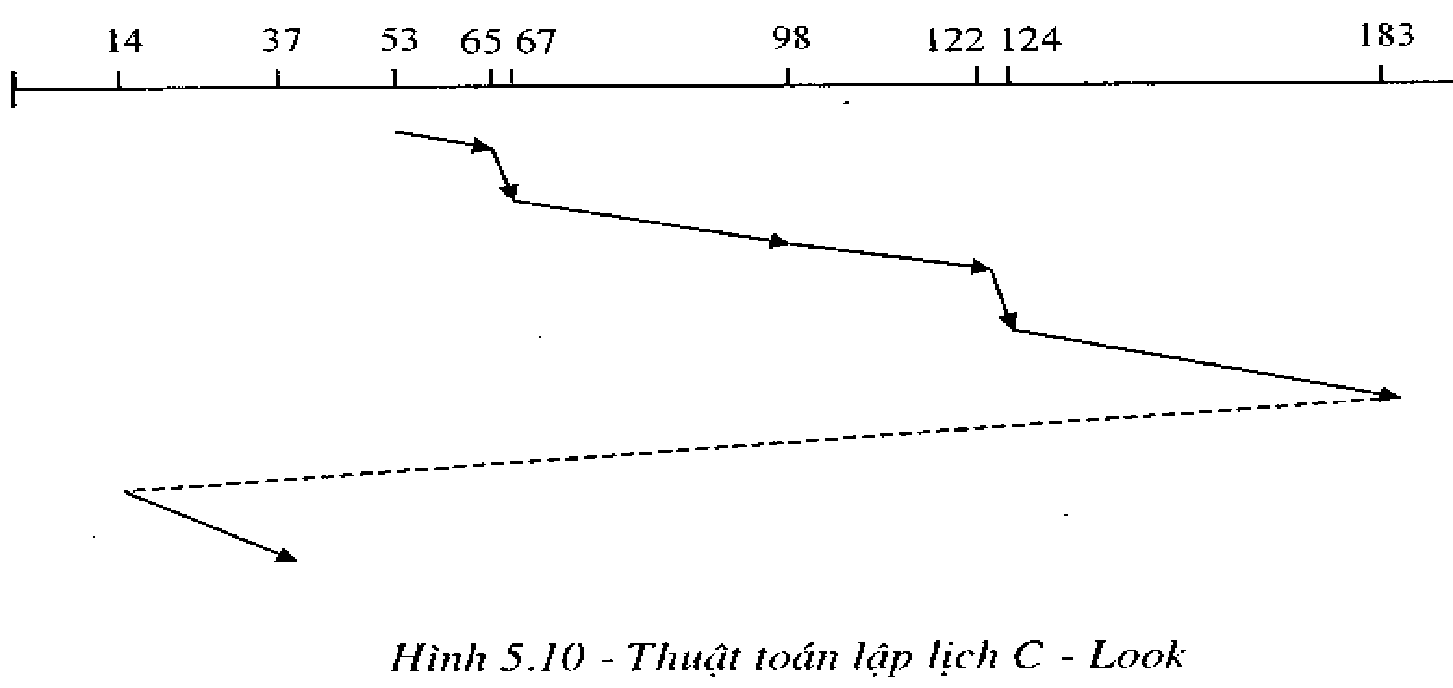
-Số bước dịch chuyển của đầu từ 208



### 7.2.6. C-Look

-Nội dung: Tương tự như Look nhưng đầu đọc ghi không phục vụ đường về

-Ví dụ: File F1 được phân bổ lần lượt các track số thứ tự :98,183,37,122,14,124,65,67  
giả sử đầu đọc đang ở track 53



Lựa chọn một giải thuật lập lịch đĩa:

-FCFS là thuật toán phù hợp khi các track cần truy xuất là liên tục

-SSTF phổ biến và có hiệu quả tốt.

-SCAN và LOOK thích hợp cho những hệ thống phải truy xuất dữ liệu lớn

## 7.3. Quản lý lỗi

Đĩa là đối tượng mà khi truy xuất có thể gây nhiều lỗi. Một trong số các lỗi thường gặp là :

*Lỗi lập trình* : yêu cầu đọc các sector không tồn tại.

Lỗi lập trình xảy ra khi yêu cầu bộ điều khiển tìm kiếm cylinder không tồn tại, đọc sector không tồn tại, dùng đầu đọc không tồn tại, hoặc vận chuyển vào và ra bộ nhớ không tồn tại. Hầu hết các bộ điều khiển kiểm tra các tham số và sẽ báo lỗi nếu không thích hợp.

*Lỗi checksum tạm thời* : gây ra bởi bụi trên đầu đọc.

Bụi tồn tại giữa đầu đọc và bề mặt đĩa sẽ gây ra lỗi đọc. Nếu lỗi tồn tại, khối có thể bị đánh dấu hỏng bởi phần mềm.

*Lỗi checksum thường trực* : đĩa bị hư vật lý trên các khối.

*Lỗi tìm kiếm* : ví dụ đầu đọc đến cylinder 7 trong khi đó phải đọc 6.

*Lỗi điều khiển* : bộ điều khiển từ chối thi hành lệnh.

## 7.4. RAM Disks

RAM disk là một phần bộ nhớ RAM được chuyển đổi để sử dụng như ổ lưu trữ. Vì tốc độ của RAM cực kỳ nhanh, thậm chí nhanh hơn so với SSD truyền thống, nên các ứng dụng/tác vụ sẽ đạt được hiệu suất lớn hơn khi chạy từ RAM disk.

-Ưu điểm của RAM disk:

Tốc độ đọc/ghi ổ đĩa RAM nhanh hơn tốc độ của những ổ cứng truyền thống, nên giúp các ứng dụng, thao tác nhanh, mượt mà hơn.

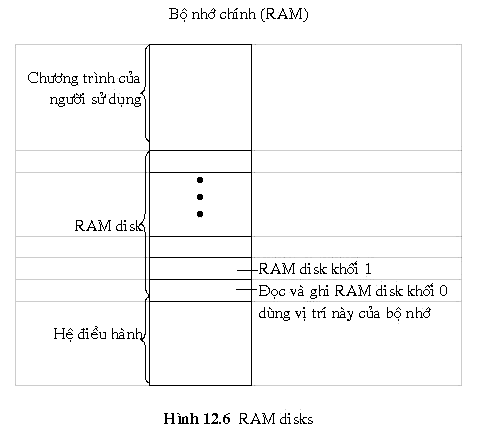
-Nhược điểm của RAM disk:

RAM đắt, dung lượng nhỏ. RAM thông thường chỉ từ 4GB-16GB, trên một số máy cao cấp hơn thì có thể có đến 64GB RAM. Nếu RAM của bạn dưới 8GB thì chỉ nên thử cho biết thôi.

Việc tạo một ổ RAM đòi hỏi phần mềm chuyên dụng.

Phần RAM bị biến thành RAM disk không thể phục vụ cho các tác vụ ghi nhớ thông thường nữa.

Tính không ổn định cao do đặc tính cố hữu của bộ nhớ truy cập tạm thời, chúng sẽ mất dữ liệu mỗi lần PC sập nguồn/mất điện/khởi động lại, nên không lưu các file quan trọng trên ổ này.



*Hình 6.5. Mô tả ý tưởng của RAM disk*

## 7.5. Interleave

Bộ điều khiển đọc ghi đĩa phải thực hiện hai chức năng là đọc/ghi dữ liệu và chuyển dữ liệu vào hệ thống. Để thực hiện được đồng bộ hai chức năng này, bộ điều khiển đọc đĩa cung cấp chức năng interleave. Trên đĩa các sector số hiệu liên tiếp nhau không nằm kế bên nhau mà có một khoảng cách nhất định, khoảng cách này được xác định bởi quá trình format đĩa. Ví dụ : giả sử hệ thống chỉ có 17 sector, và interleave được chọn là 4 thì các sector được bố trí theo thứ tự như sau :

1, 14, 10, 6, 2, 15, 11, 7, 3, 16, 12, 8, 4, 17, 13, 9, 5

Cách đọc lần lượt như sau :

 Lần 1:

*1*, 14, 10, 6, *2*, 15, 11, 7, *3*, 16, 12, 8, *4*, 17, 13, 9, *5*

Lần 2:

1, 14, 10, *6*, 2, 15, 11, *7*, 3, 16, 12, *8*, 4, 17, 13, *9*, 5

Lần 3:

1, 14, *10*, 6, 2, 15, *11*, 7, 3, 16, *12*, 8, 4, 17, *13*, 9, 5

Lần 4:

1, *14*, 10, 6, 2, *15*, 11, 7, 3, *16*, 12, 8, 4, *17*, 13, 9, 5

Như vậy sau bốn lần thứ tự các sector đọc được vẫn là từ 1 đến 17

# CHƯƠNG 8. FILE EXPLORER

## 8.1 Khái niệm file

* Thông tin lưu trữ trên nhiều phương tiện/thiết bị lưu trữ khác nhau

Ví dụ: Đĩa từ, băng từ, đĩa quang...

Thiết bị lưu trữ được mô hình như một mảng của các khối nhớ



* File là tập thông tin ghi trên thiết bị lưu trữ.

File là đơn vị lưu trữ của hệ điều hành trên bộ nhớ ngoài

File bao gồm dãy các bits, bytes, dòng, bản ghi,... mang ý nghĩa

được định nghĩa bởi người tạo ra

* Cấu trúc của file được định nghĩa theo loại file

File văn bản: Chuỗi ký tự tổ chức thành dòng

File đối tượng: Bytes được tổ chức thành khối để chương trình

liên kết (linker) hiểu được

File thực thi: Chuỗi các mã lệnh có thể thực hiện trong bộ nhớ

* Các thuộc tính file
* Tên file (Name): Chuỗi ký tự (hello.c)

Thông tin lưu dưới dạng người dùng có thể đọc được

Có thể phân biệt chữ hoa/chữ thường

Đảm bảo tính độc lập của file với tiến trình, người dùng...

A tạo file hello.c bằng notepad trên hệ Windows

B dùng emacs trên linux sửa lại file bởi xác định tên hello.c

* Định danh (Identifier): Thẻ xác định duy nhất một file
* Kiểu (Type): Dùng cho hệ thống hỗ trợ nhiều kiểu file

Có thể xác định kiểu file dựa trên một phần của tên file

Ví dụ: .exe, .com/ .doc, .txt/ .c, .jav, .pas/ .pdf, .jpg,...

Dựa trên kiểu, HĐH sẽ thao tác trên tập tin phù hợp

Thực hiện file thực thi mà file nguồn đã sửa ⇒ Dịch lại

Nháy đúp vào một file văn bản (\*.doc)⇒ Gọi word processor

* Vị trí (Position): Trỏ tới thiết bị và vị trí của file trên đó
* Kích thước (Size): Kích thước hiện thời/ tối đa của file
* Bảo vệ (Protection): Điều khiển truy nhập: Ai có thể đọc/ghi..
* Thời gian (Time): Thời điểm tạo, sửa đổi, sử dụng cuối ...
* Thuộc tính file được lưu trong cấu trúc dữ liệu: Bản ghi file

Có thể chỉ chứa tên file và định danh file; định danh file xác định

các thông tin còn lại

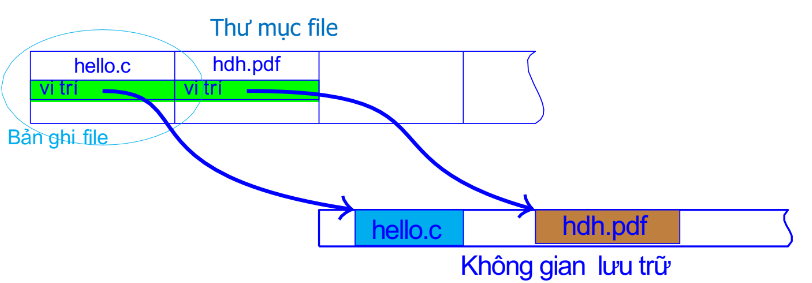
Kích thước từ vài bytes lên tới kilobytes

* Các bản ghi file được lưu giữ trong Thư mục file

Kích thước có thể đạt tới Megabytes

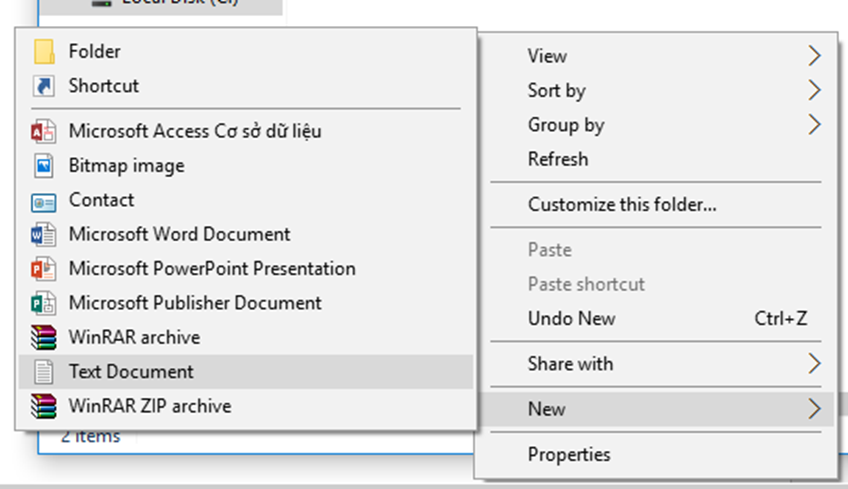
Thường được lưu trữ trên thiết bị nhớ ngoài

Được đưa từng phần vào bộ nhớ khi cần thiết



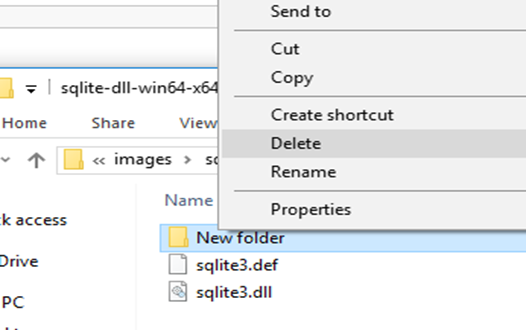
## 8.2. Các thao tác cơ bản:

### 8.2.1. Tạo một file, thư mục



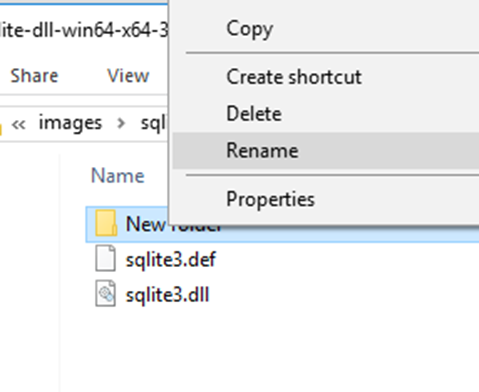
*Hình 8.1. Tạo một file*

### 8.2.2. Xóa một file, thư mục



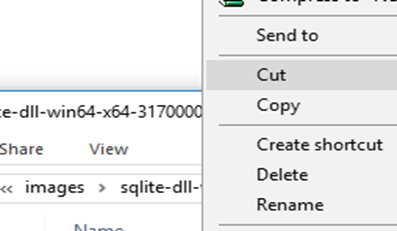
*Hình 8.2. Xóa 1 thư mục*

### 8.2.3 Đổi tên một file, thư mục



*Hình 8.3. Đổi tên thư mục*

### 8.2.4 Copy, cut một file, thư mục



*Hình 8.4. Cut một thư mục, tập tin*

# 

# CHƯƠNG 9. QUẢN LÝ BỘ NHỚ NGOÀI TRONG HỆ ĐIỀU HÀNH WINDOWS

## 9.1. Phương pháp quản lý không gian nhớ tự do

Phương pháp liệt kê (Free list):

+ Trong Windows phương pháp liệt kê được sử dụng chính gồm có 4 loại cơ

bản : Active List,Inactive-dirty list ,Inactive-clean list,Free list.

Phương pháp liệt kê sử dụng một danh sách (free list) để lưu trữ các vùng nhớ tự do và các vùng nhớ đã được giải phóng. Free list chứa các mục (entries) biểu diễn các vùng nhớ tự do và thông tin liên quan. Mỗi mục thường bao gồm địa chỉ bắt đầu và kích thước của vùng nhớ. Khi có yêu cầu cấp phát, hệ điều hành sẽ duyệt qua danh sách này để tìm vùng nhớ phù hợp.

Phương pháp liệt kê (Free list) được sử dụng trong hệ điều hành Windows để quản lý bộ nhớ. Khi một chương trình yêu cầu sử dụng bộ nhớ, hệ điều hành sẽ tìm kiếm các khối bộ nhớ trống trên free list và cấp phát cho chương trình đó sử dụng.

Các bước thực hiện phương pháp liệt kê trong hệ điều hành Windows như sau:

1. Khởi tạo free list: Đầu tiên, cần khởi tạo một danh sách liên kết rỗng để đại diện cho free list. Danh sách này sẽ lưu trữ thông tin về các vùng nhớ tự do có sẵn để cấp phát.

2. Cấp phát bộ nhớ: Khi một quá trình hoặc ứng dụng yêu cầu cấp phát bộ nhớ, hệ thống quản lý bộ nhớ thực hiện các bước sau đây:

-Duyệt qua danh sách liên kết để tìm kiếm một vùng nhớ tự do có đủ kích thước để phục vụ yêu cầu.

-Nếu tìm thấy một vùng nhớ tự do phù hợp, hệ thống quản lý bộ nhớ sẽ cắt ra một phần của vùng nhớ đó và cấp phát cho quá trình hoặc ứng dụng yêu cầu.

-Cập nhật danh sách liên kết với thông tin về vùng nhớ còn lại.

3. Quản lý việc giải phóng vùng nhớ: Khi một vùng nhớ được giải phóng, hệ điều hành sẽ thực hiện các bước sau đây:

-Hợp nhất vùng nhớ giải phóng vào danh sách liên kết.

-Kiểm tra xem các vùng nhớ liền kề có thể được kết hợp thành một vùng nhớ tự do lớn hơn không.

-Cập nhật danh sách liên kết với thông tin về vùng nhớ mới được giải phóng.

4. Tối ưu hóa và quản lý: Hệ thống quản lý bộ nhớ Windows áp dụng các biện pháp tối ưu hóa và quản lý để cải thiện hiệu suất và sử dụng không gian nhớ hiệu quả. Các biện pháp này bao gồm:

-Phân chia free list thành các phân đoạn (segments) để giảm thời gian tìm kiếm.

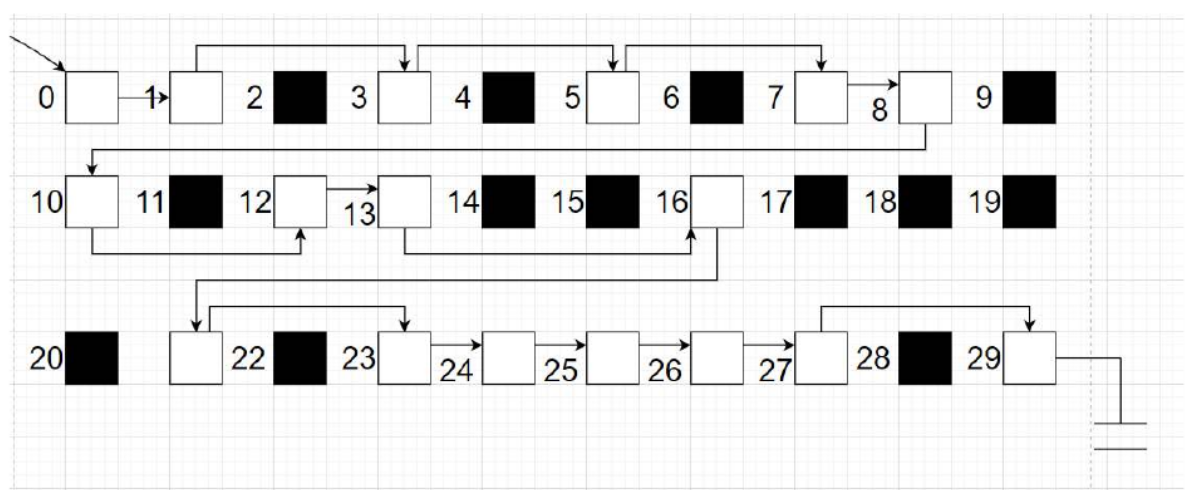
-Sử dụng các cấu trúc dữ liệu khác như cây nhị phân cân bằng (AVL tree) hoặc bảng băm (hash table) để tăng tốc độ tìm kiếm.

-Áp dụng các thuật toán phân mảnh như first-fit, best-fit hoặc worst-fit để quản lý vùng nhớ.

Ví dụ : Cho không gian đĩa từ có các block

0,1,3,5,7,8,10,12,13,16,21,23,24,25,26,27,29 là các block tự do. Khi đó ta có thể

quản lý không gian nhớ tự do sẽ như sau:

free list được sử dụng trong các hệ điều hành windows nafo

Một số ưu điểm và nhược điểm của phương pháp liệt kê (free list)

* Ưu điểm :

-Đơn giản và dễ hiểu: Phương pháp liệt kê sử dụng một danh sách liên kết đơn giản để theo dõi các vùng nhớ tự do. Cấu trúc dữ liệu này dễ hiểu và triển khai, không yêu cầu nhiều tài nguyên hoặc kiến thức phức tạp.

-Linh hoạt và tiết kiệm không gian: Phương pháp liệt kê cho phép linh hoạt trong việc cấp phát và giải phóng bộ nhớ. Nó có thể làm việc với các kích thước vùng nhớ khác nhau và không yêu cầu sự liên tục về không gian nhớ. Điều này giúp tối ưu hóa sử dụng không gian nhớ và giảm sự lãng phí.

-Quản lý phân mảnh tốt: Phương pháp liệt kê có khả năng gộp các vùng nhớ tự do liền kề để tạo ra các vùng nhớ lớn hơn. Điều này giúp giảm phân mảnh và tối ưu hóa việc sử dụng không gian nhớ.

-Tính đồng bộ và đa nhiệm: Phương pháp liệt kê có thể dễ dàng được đồng bộ hóa và sử dụng trong môi trường đa nhiệm. Bằng cách sử dụng các cơ chế đồng bộ hóa như khóa (lock) hoặc semaphore, có thể đảm bảo tính nhất quán và tránh xung đột khi truy cập free list từ nhiều tiến trình hoặc luồng.

-Hiệu quả trong việc quản lý các vùng nhớ nhỏ: Phương pháp liệt kê cho phép quản lý các vùng nhớ nhỏ một cách hiệu quả. Bằng cách chia các vùng nhớ tự do thành các nhóm kích thước, có thể tối ưu hóa quá trình tìm kiếm và cập nhật free list cho các yêu cầu cấp phát nhỏ.

-Cấp phát linh hoạt: Phương pháp liệt kê cho phép cấp phát vùng nhớ có kích thước tùy ý, không bị ràng buộc bởi các khối cố định. Điều này giúp tối ưu hóa sử dụng không gian nhớ bằng cách phân chia vùng nhớ theo nhu cầu thực tế của ứng dụng.

-Chi phí thấp: Phương pháp liệt kê không đòi hỏi nhiều tài nguyên phức tạp để triển khai và quản lý. Nó chỉ yêu cầu một danh sách liên kết đơn giản và các thao tác đơn giản như chèn, xóa và cập nhật. Do đó, nó có chi phí thấp trong việc triển khai và sử dụng.

-Khả năng mở rộng: Phương pháp liệt kê có thể mở rộng để đáp ứng yêu cầu quản lý không gian nhớ lớn hơn. Các vùng nhớ tự do mới có thể được thêm vào danh sách liên kết theo nhu cầu, cho phép quản lý không gian nhớ mở rộng linh hoạt.

-Tính linh hoạt cao: Phương pháp liệt kê cho phép quản lý không gian nhớ tự do trong nhiều hình thức khác nhau. Nó có thể được áp dụng cho cả quản lý tĩnh và động, và có thể tương thích với nhiều thuật toán và chiến lược quản lý không gian nhớ khác nhau.

* Nhược điểm:

-Phân mảnh ngoại tuyến: Phương pháp liệt kê không giải quyết được vấn đề phân mảnh ngoại tuyến. Khi một vùng nhớ tự do được giải phóng, nó có thể tạo ra các khoảng trống không liền kề trong không gian nhớ. Điều này dẫn đến phân mảnh không thể giải quyết trong quá trình cấp phát bộ nhớ.

-Hiệu suất tìm kiếm: Trong phương pháp liệt kê, việc tìm kiếm vùng nhớ tự do phụ thuộc vào việc duyệt qua danh sách liên kết. Điều này có thể làm giảm hiệu suất tìm kiếm, đặc biệt khi danh sách trở nên lớn. Quá trình tìm kiếm có thể tốn nhiều thời gian và tài nguyên tính toán.

-Chi phí cập nhật: Khi một vùng nhớ tự do được cấp phát hoặc giải phóng, free list phải được cập nhật tương ứng. Quá trình này có thể tốn thời gian và tài nguyên. Nếu không được cập nhật chính xác, free list có thể trở nên không nhất quán và gây ra lỗi trong quá trình cấp phát và giải phóng.

-Không hiệu quả cho các vùng nhớ nhỏ: Phương pháp liệt kê không hiệu quả trong việc quản lý các vùng nhớ nhỏ. Khi kích thước yêu cầu không phù hợp với bất kỳ vùng nhớ tự do cụ thể nào trong free list, sẽ xảy ra sự lãng phí không gian nhớ và tạo ra hiệu ứng tương tự như phân mảnh.

-Không đồng bộ: Nếu có nhiều tiến trình hoặc luồng cố gắng truy cập và cập nhật free list cùng một lúc, phương pháp liệt kê có thể gặp vấn đề về đồng bộ hóa. Điều này có thể dẫn đến xung đột và sự không đồng nhất trong quá trình quản lý bộ nhớ.

-Phân mảnh nội tuyến: Trong phương pháp liệt kê, khi các vùng nhớ tự do được cấp phát và giải phóng không theo thứ tự, có thể xảy ra phân mảnh nội tuyến. Điều này làm cho việc cấp phát các vùng nhớ lớn trở nên khó khăn hơn do sự phân mảnh của free list.

-Hiệu suất không ổn định: Hiệu suất của phương pháp liệt kê có thể không ổn định khi free list trở nên quá lớn. Với số lượng lớn các vùng nhớ tự do trong danh sách liên kết, việc tìm kiếm và cập nhật có thể trở nên chậm hơn và tiêu tốn nhiều thời gian.

-Chi phí bổ sung cho tìm kiếm vùng nhớ: Một số biến thể của phương pháp liệt kê, như liệt kê theo kích thước hoặc liệt kê theo thời gian cấp phát, có thể đòi hỏi chi phí bổ sung cho việc duy trì thông tin kích thước hoặc thời gian cấp phát của các vùng nhớ tự do. Điều này có thể làm gia tăng tài nguyên và tốn thời gian cho các thao tác cập nhật và tìm kiếm.

-Không xử lý được tình huống hụt nguồn: Phương pháp liệt kê không đưa ra giải pháp cho tình huống hụt nguồn (out of memory). Khi không còn đủ không gian nhớ tự do để cấp phát, phương pháp liệt kê không thể tự động mở rộng hoặc thu hồi thêm không gian nhớ.

-Sự lãng phí không gian: Phương pháp liệt kê có thể gây ra sự lãng phí không gian khi một số vùng nhớ tự do nhỏ hoặc phân mảnh được giữ lại trong free list. Điều này làm tăng yêu cầu về không gian lưu trữ và có thể ảnh hưởng đến hiệu quả sử dụng bộ nhớ.

-Làm tăng thời gian truy nhập dữ liệu: Khi sử dụng phương pháp liệt kê, việc tìm kiếm vùng nhớ tự do trong free list thông qua duyệt danh sách liên kết có thể tốn thời gian hơn so với các phương pháp khác như bảng băm (hash table). Điều này đặc biệt đúng khi danh sách liên kết trở nên lớn.

* Đề xuất cải tiến:

Để cải tiến phương pháp liệt kê, có thể áp dụng các biện pháp sau:

-Phân chia danh sách tự do thành các phân đoạn nhỏ hơn để giảm thời gian tìm kiếm. Thay vì duyệt toàn bộ danh sách, hệ điều hành chỉ cần tìm kiếm trong phân đoạn chứa kích thước phù hợp với yêu cầu cấp phát.

-Sử dụng các thuật toán tối ưu hóa phân mảnh để giảm hiện tượng phân mảnh không gian như thu gom (compaction) hoặc sắp xếp lại vùng nhớ.

-Kết hợp phương pháp liệt kê với các phương pháp khác như cấp phát liên tục để tận dụng các ưu điểm của cả hai phương pháp.

## 9.2. Phương pháp cấp phát không gian nhớ tự do

Cấp phát liên tục (Contiguous)

Có hai loại cấp phát liên tục chính là cấp phát liên tục tĩnh (Static Contiguous Allocation) và cấp phát liên tục động (Dynamic Contiguous Allocation).

-Cấp phát liên tục tĩnh (Static Contiguous Allocation): Trong phương pháp này, các phân vùng bộ nhớ được cấp phát cho mỗi quá trình theo cách cố định và không thay đổi. Điều này có nghĩa là các quá trình có kích thước cố định và không thể mở rộng hoặc thu hồi không gian bộ nhớ của mình.

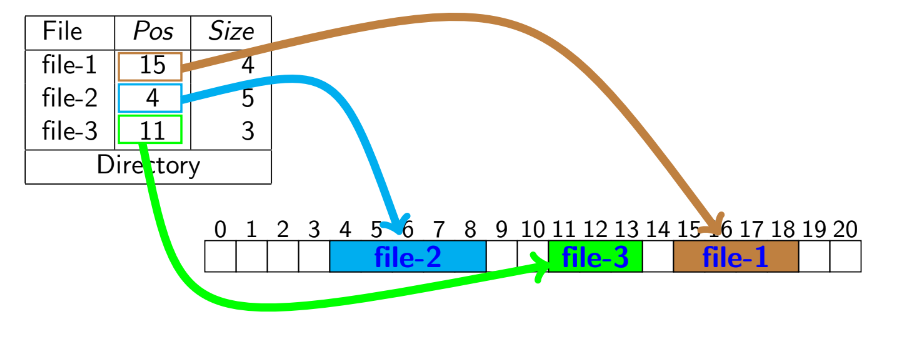
-Cấp phát liên tục động (Dynamic Contiguous Allocation): Trong phương pháp này, không gian bộ nhớ được quản lý một cách linh hoạt và các quá trình có thể yêu cầu và giải phóng vùng nhớ theo nhu cầu. Quá trình có thể mở rộng hoặc thu hồi không gian bộ nhớ của mình tùy thuộc vào yêu cầu và sự thay đổi của hệ thống.

Để phân bổ không gian nhớ cho một file, hệ thống chọn một đoạn liên tục các khối đĩa tự do để cấp phát cho file đó. Với phương pháp này, để định vị file hệ thống chỉ cần biết địa chỉ của khối đĩa tự do đầu tiên và số lượng khối (block) đã dùng.

Nguyên tắc hoạt động của cấp phát liên tục (Contiguous Allocation) trong quản lý bộ nhớ ngoài trong hệ điều hành Windows có các bước sau:

* Khởi tạo không gian bộ nhớ ngoài: Trước khi quản lý bộ nhớ ngoài, hệ điều hành Windows phải khởi tạo không gian bộ nhớ ngoài, ví dụ như ổ cứng, ổ SSD hoặc ổ USB. Đây là không gian lưu trữ vật lý mà các tệp và dữ liệu sẽ được lưu trữ.
* Quá trình yêu cầu cấp phát bộ nhớ: Khi một ứng dụng hoặc quá trình yêu cầu cấp phát bộ nhớ ngoài, hệ điều hành Windows sẽ xử lý yêu cầu này và tìm kiếm một vùng nhớ liên tục trống đủ lớn để đáp ứng yêu cầu.
* Kiểm tra tính khả dụng của vùng nhớ: Hệ điều hành Windows sẽ kiểm tra tính khả dụng của vùng nhớ bằng cách xem xét trạng thái của các khối vùng nhớ trong không gian bộ nhớ ngoài. Các vùng nhớ đã được cấp phát và sử dụng sẽ được đánh dấu là "đã sử dụng", trong khi các vùng nhớ chưa được cấp phát hoặc đã được giải phóng sẽ được đánh dấu là "trống".
* Tìm kiếm vùng nhớ phù hợp: Hệ điều hành Windows tìm kiếm một vùng nhớ liên tục trống đủ lớn để cấp phát cho quá trình yêu cầu. Thông thường, các thuật toán tìm kiếm như First Fit (tìm vùng nhớ đầu tiên phù hợp), Best Fit (tìm vùng nhớ nhỏ nhất phù hợp) hoặc Worst Fit (tìm vùng nhớ lớn nhất phù hợp) có thể được sử dụng để tìm kiếm vùng nhớ thích hợp.
* Cấp phát và đánh dấu vùng nhớ: Nếu hệ điều hành tìm thấy một vùng nhớ phù hợp, nó sẽ cấp phát vùng nhớ này cho quá trình yêu cầu và đánh dấu nó là "đã sử dụng" trong bảng điều khiển bộ nhớ ngoài. Quá trình sẽ nhận được địa chỉ bắt đầu của vùng nhớ đã được cấp phát.
* Giải phóng vùng nhớ: Khi quá trình không cần sử dụng vùng nhớ đã được cấp phát nữa, nó sẽ thông báo cho hệ điều hành và yêu cầu giải phóng vùng nhớ. Hệ điều hành Windows sẽ cập nhật lại bảng điều khiển bộ nhớ ngoài và đánh dấu vùng nhớ đã được giải phóng là "trống" để có thể sử dụng lại cho các yêu cầu cấp phát sau này.
* Quản lý phân mảnh: Một vấn đề phổ biến trong cấp phát liên tục là phân mảnh không gian bộ nhớ. Khi các vùng nhớ được cấp phát và giải phóng không liên tục, có thể xảy ra phân mảnh bộ nhớ, khiến việc tìm kiếm và cấp phát vùng nhớ phù hợp trở nên khó khăn. Để giải quyết vấn đề này, hệ điều hành Windows có thể sử dụng các kỹ thuật như gộp các vùng nhớ trống liền kề hoặc sắp xếp lại các vùng nhớ để tạo ra các vùng nhớ liên tục lớn hơn.
* Quản lý an toàn và bảo mật: Hệ điều hành Windows áp dụng các biện pháp bảo mật để đảm bảo an toàn của quá trình và dữ liệu trong quá trình cấp phát liên tục. Nó sử dụng các cơ chế bảo vệ như quyền truy cập (access rights) và quản lý quyền truy cập (access control) để đảm bảo rằng quá trình chỉ có quyền truy cập và sử dụng vùng nhớ mà nó đã được cấp phát.
* Hiệu suất và tối ưu hóa: Hệ điều hành Windows thường thực hiện các cải tiến và tối ưu hóa để cải thiện hiệu suất của cấp phát liên tục. Điều này có thể bao gồm việc sử dụng bộ nhớ đệm (cache) để giảm thời gian truy cập vào bảng điều khiển bộ nhớ ngoài và tối ưu hóa các thuật toán tìm kiếm và cấp phát vùng nhớ để đạt được hiệu suất tốt nhất.
* Trên cơ bản, cấp phát liên tục trong hệ điều hành Windows hoạt động bằng cách tìm kiếm và cấp phát các vùng nhớ liên tục cho các quá trình, sử dụng bảng điều khiển bộ nhớ để theo dõi trạng thái và tính khả dụng của các vùng nhớ. Điều này giúp quản lý và tối ưu hóa việc sử dụng bộ nhớ và đảm bảo sự an toàn và hiệu suất trong hệ thống.

Ví dụ:(GEEKSFORGEEKS.ORG)



-Chúng ta có thể thấy trong sơ đồ đã cho, có một tệp. Tên của tệp là ‘mail.’ Tệp bắt đầu từ khối thứ 15 và độ dài của tệp là 4. Vì vậy, tệp chiếm 4 khối theo cách liền kề. Như vậy, nó sẽ chứa các khối 15, 16, 17, 18.

* Ưu điểm

-Hỗ trợ tốt cho phương pháp:

-Truy nhập tuần tự.

-Truy nhập trực tiếp.

* Nhược điểm

-Phải chọn được thuật toán tối ưu để tìm các vùng không gian nhớ tự do cấp phát cho file

+Một trong 3 thuật toán:

+First-fit

+Best-fit

+Worst-fit

- Có thể xảy ra trương hợp không đủ số khối đĩa tự do liên tiếp chưa sử dụng để cấp phát cho file chẳng hạn như là kích thước file lớn hơn cùng các khối đĩa tự do liên tục lớn nhất trong bộ nhớ còn trống).

- Trong trường hợp các khối đĩa tự do nằm tản mạn sẽ không sử dụng được gây tình trạng lãng phí bộ nhớ trong khi đó dung lượng không đủ để cấp phát quyền truy xuất cho file có kích thước dung lượng lớn.

+Các thuật toán tối ưu được sử dụng để tìm không gian nhớ tự do thích hợp:

- First –fit : Cấp phát vùng nhớ tự do đầu tiên đủ lớn tìm được cho file.Việc tìm kiếm có thể bắt đầu tại đầu tập hợp các vùng nhớ trống hay tại điểm kết thúc của tìm kiếm first-fit trước đó đã kết thúc. Thuật toán dừng khi tìm kiếm được vùng nhớ đầu tiên đủ điều kiện cho file.Do đó thuật toán này thời gian cấp phát quyền sử dụng ô nhớ nhanh chóng nhưng cũng gây lãng phí bộ nhớ.

- Best-fit : Cấp phát vùng nhớ tự do nhỏ nhất tuy nhiên đủ lớn tìm được. Chúng ta phải tìm toàn bộ danh sách trừ khi danh sách đó đã được xếp thứ tự theo kích thước, Thuật toán này tạo ra lỗ trống nhỏ nhất còn lại dư thừa.Do đó có thể tận dụng tối ưu không gian nhớ của đĩa. Nhược điểm là thời gian chờ cấp phát tương đối dài,hiệu năng hoạt động chung của hệ thống giảm khi có nhiều chương trình thực thi đòi hỏi quyền cấp phát bộ nhớ.

- Worst-fit: Cấp phát vùng nhớ tự do có kích thước lớn nhất cho file. Thuật toán này phải tìm danh sách trừ khi nó được xếp theo thứ tự kích thước. Thuật toán này tạo ra vùng nhớ trống còn lại lớn nhất cái mà có thể có ích hơn lỗ trống nhỏ từ thuật toán best-fit.

- Trong hệ điều hành Windows, Windows trải ra tất cả các tập tin của hệ thống trên toàn bộ bề mặt đĩa (các tập tin không ghi sát với nhau như ở hệ điều hành

Linux) và dữ liệu của tập tin khi ghi xuống đĩa được hệ thống cấp phát dung lượng lớn hơn kích thước của tập tin cần ghi, do đó khi kích thước thay đổi phần dữ liệu mới sẽ được thêm vào phần trống (đã được chừa sẵn từ trước khi tạo tập tin này), trong trường hợp vùng trống này không đủ thì hệ thống sẽ tìm một vị trí khác còn trống có thể chứa được tập tin này và dời tập tin này đến vị trí đó.

Sự phân mảnh chi xảy ra khi dung lượng của đĩa không còn đủ để chứa tập tin mà không cần phải chia nhỏ tập tin này ra. Trường hợp này chỉ xảy ra khi dung lượng của đĩa đã sử dụng quá 80%.

Với cách thực hiện như trên như vậy Windows đã chống phân mảnh ngay từ khi thực hiện ghi tập tin trên đĩa, sự phân mảnh chỉ xảy ra khi dung lượng đĩa còn ít hơn 20%.

## 9.3. Lập lịch cho đĩa

Shortest Seek Time First (SSTF)

Thuật toán Shortest Seek Time First (SSTF) là một thuật toán lập lịch truy cập đĩa được sử dụng trong hệ điều hành Windows để tối ưu hóa việc đọc/ghi dữ liệu từ đĩa cứng. Mục tiêu chính của thuật toán SSTF là giảm thiểu thời gian truy cập đĩa bằng cách chọn ra yêu cầu truy cập có khoảng cách tìm kiếm (thời gian tìm kiếm) gần nhất với vị trí hiện tại của đầu đọc/ghi đĩa.

Thuật toán SSTF trong hệ điều hành Windows hoạt động theo các bước sau:

Bước 1: Khởi tạo - Khi một yêu cầu truy cập vào đĩa cứng được tạo ra (như đọc/ghi dữ liệu từ tập tin), hệ điều hành đưa yêu cầu này vào I/O queue (hàng đợi yêu cầu I/O) .

Bước 2: Xác định vị trí hiện tại - Hệ điều hành xác định vị trí hiện tại của đầu đọc/ghi đĩa.

Bước 3: Chọn yêu cầu tối ưu - Hệ điều hành lựa chọn yêu cầu truy cập có khoảng cách tìm kiếm rút ngắn nhất từ vị trí hiện tại của đầu đọc/ghi đĩa. Để làm điều này, nó so sánh khoảng cách tìm kiếm giữa yêu cầu hiện tại và tất cả các yêu cầu trong hàng đợi I/O và chọn yêu cầu có khoảng cách tìm kiếm ngắn nhất.

Bước 4: Truy cập đĩa - Hệ điều hành thực hiện truy cập vào đĩa cứng để đọc/ghi dữ liệu cho yêu cầu được chọn. Sau khi hoàn thành yêu cầu này, nó đã bị xóa khỏi hàng đợi I/O.

Bước 5: Quay lại bước 2 - Sau khi xử lý xong một yêu cầu, hệ điều hành quay lại bước 2 để xác định vị trí hiện tại của đầu đọc/ghi đĩa và tiếp tục từ bước 3 để chọn yêu cầu truy cập tiếp theo.

Có một số ưu điểm quan trọng của thuật toán Shortest Seek Time First (SSTF) trong hệ điều hành Windows:

* Giảm thiểu thời gian truy cập: SSTF ưu tiên xử lý các yêu cầu truy cập có khoảng cách tìm kiếm gần nhất từ vị trí hiện tại của đầu đọc/đĩa ghi. Điều này giúp giảm thiểu thời gian truy cập ổ cứng do không phải di chuyển xa để trả lời các yêu cầu.
* Tăng hiệu suất: Giảm thiểu thời gian truy cập, thuật toán SSTF có khả năng cung cấp hiệu suất cao hơn so với các thuật toán lập lịch truy cập đĩa cứng khác, chẳng hạn như FCFS (First-Come, First-Served) hay SCAN .
* Độ phản hồi nhanh: Với SSTF, các yêu cầu truy cập có thể được xử lý nhanh chóng do ưu tiên xử lý các yêu cầu có khoảng cách tìm kiếm ngắn nhất. Điều này dẫn đến phản hồi tốt và thời gian đáp ứng nhanh hơn cho các yêu cầu từ người dùng.
* Đơn giản và dễ triển khai: Thuật toán SSTF có tính đơn giản và dễ triển khai trong hệ điều hành Windows và các hệ thống lưu trữ khác. Không yêu cầu quá nhiều tài nguyên tính toán hoặc địa chỉ phức tạp thông tin.
* Tối ưu hóa đường đi: SSTF cho phép tối ưu hóa đường đi của đầu đọc/ghi đĩa cứng bằng cách lựa chọn yêu cầu truy cập có khoảng cách tìm kiếm gần nhất. Khi các yêu cầu được xử lý theo thứ tự gần nhất, đầu đọc/ghi di chuyển ít hơn, giảm thời gian chờ tối thiểu và tăng hiệu suất.
* Phản hồi đồng thời: SSTF cung cấp khả năng phản hồi đồng thời cho các yêu cầu truy cập. Khi một yêu cầu được xử lý, thuật toán có thể tiếp tục tìm kiếm yêu cầu tiếp theo mà không cần đợi đợi hoàn thành. Điều này giúp tối ưu hóa việc sử dụng tài nguyên và đáp ứng tốt hơn cho nhiều yêu cầu cùng một lúc.
* Dễ dàng triển khai và phân tích hợp nhất: SSTF là một thuật toán đơn giản và dễ dàng triển khai trong hệ điều hành Windows. Nó có thể được tích hợp vào các cơ chế quản lý và lập lịch I/O hiện tại một cách dễ dàng, không yêu cầu thay đổi lớn trong hệ thống.
* Hiệu quả cho các ứng dụng có truy cập ngẫu nhiên: Trong một số trường hợp, các ứng dụng có yêu cầu truy cập ngẫu nhiên vào đĩa, không chấp nhận một mẫu truy cập đều đặn. SSTF làm việc tốt cho những vấn đề này bởi vì nó tập trung vào tối ưu hóa khoảng cách tìm kiếm và không yêu cầu các yêu cầu truy vấn một trình tự nhất định.

Thuật toán Shortest Seek Time First (SSTF) trong hệ điều hành Windows cũng có một số nhược điểm cần lưu ý:

* Hiện tượng trì hoãn (bỏ đói): Nếu có một số yêu cầu tập trung ở một vị trí cụ thể trên đĩa cứng, SSTF có thể gây ra hiện tượng trì hoãn cho các yêu cầu khác. Các yêu cầu có khoảng cách tìm kiếm xa hơn có thể bị bỏ qua và phải chờ đợi lâu hơn để được xử lý. Điều này có thể làm giảm hiệu suất và tăng thời gian phản hồi của hệ thống.
* Không cân nhắc đối với việc phân bố công việc: SSTF ưu tiên xử lý các yêu cầu có khoảng cách tìm kiếm gần nhất. Điều này có thể dẫn đến việc một số vùng của ổ cứng không thể truy cập được trong thời gian dài. Yêu cầu tệp trung ở một khu vực nhất định có thể khiến các vùng khác bị loại bỏ và không được xử lý, gây ra sự mất cân đối trong quá trình phân tích công việc trên đĩa cứng.
* Cần cập nhật liên tục vị trí hiện tại: Để thực hiện thuật toán SSTF, hệ điều hành cần cập nhật liên tục vị trí hiện tại của đầu đọc/ghi đĩa. This task tài nguyên thời gian và tài nguyên tính toán. Nếu không cập nhật đúng thời hạn, thuật toán có thể không hoạt động hiệu quả và không đưa ra quyết định tối ưu cho việc chọn yêu cầu truy cập.
* Khả năng xảy ra bế tắc: Trong một số trường hợp, SSTF có thể gây ra bế tắc (tình huống mà các yêu cầu không thể hoàn thành do xung đột tài nguyên). Nếu các yêu cầu tệp trung ở hai vị trí của đối tượng trên đĩa, SSTF có thể rơi vào tình trạng xung đột và không thể tiếp tục xử lý các yêu cầu.
* Không bảo đảm công bằng: SSTF không bảo đảm công bằng giữa các yêu cầu truy cập. Các yêu cầu có khoảng cách tìm kiếm lớn hơn có thể bị bỏ qua hoặc chờ đợi lâu hơn so với các yêu cầu có khoảng cách tìm kiếm nhỏ hơn. Điều này có thể tạo ra một sự cố không công bằng trong quá trình xử lý yêu cầu truy cập của người dùng.

## 9.4. Một số hệ thống file trong windows

Hệ điều hành Windows hỗ trợ và tương thích với nhiều hệ thống tệp để quản lý bộ nhớ bên ngoài như ổ đĩa cứng di động, thẻ nhớ, ổ đĩa flash USB và ổ đĩa SSD bên ngoài. Dưới đây là một số hệ thống tập tin ứng dụng được sử dụng trong Windows để quản lý bộ nhớ bên ngoài:

* NTFS (New Technology File System): NTFS là hệ thống tập tin chính của Windows và hỗ trợ quản lý bộ nhớ ngoài. Với NTFS, bạn có thể định dạng và sử dụng ổ cứng di động, ổ đĩa SSD bên ngoài và các thiết bị lưu trữ khác. NTFS cung cấp các tính năng bảo mật, quản lý không gian đĩa và hỗ trợ các tập tin lớn.
* FAT32 (Bảng phân bổ tệp 32): FAT32 là hệ thống tệp tương thích rộng rãi và được hỗ trợ trên nhiều hệ điều hành, bao gồm cả Windows. FAT32 thích hợp cho các thiết bị lưu trữ di động như thẻ nhớ và ổ đĩa flash USB. Tuy nhiên, FAT32 có giới hạn về kích thước tập tin tối đa là 4GB và kích thước ổ đĩa tối đa là 2TB.
* exFAT (Bảng phân bổ tệp mở rộng): exFAT là một hệ thống tệp mở rộng của FAT32 và được thiết kế để hỗ trợ các thiết bị lưu trữ lớn hơn và tệp có kích thước lớn hơn. exFAT thường được sử dụng trên các thiết bị lưu trữ di động như ổ đĩa cứng di động, thẻ nhớ và ổ đĩa flash USB. exFAT không có giới hạn về kích thước tệp và kích thước ổ đĩa.
* FAT16 (Bảng phân bổ tệp 16): FAT16 là một hệ thống tệp cũ hơn và ít được sử dụng hơn trong các thiết bị lưu trữ hiện đại. Nó có giới hạn về kích thước tệp tối đa là 2GB và kích thước ổ đĩa tối đa là 4GB.

**Kết Luận**

Trên đây là những nghiên cứu của nhóm 5 về quản lý bộ nhớ ngoài trong Window. Qua bài tập lớn này đã phần nào chỉ ra được cấu tạo của bộ nhớ ngoài cũng các phương thức lập lịch cho bộ nhớ ngoài. Hi vọng là từ đó chúng ta có thể lựa chọn cách lập lịch cũng như quản lý bộ nhớ ngoài sao cho tối ưu nhất. Chúng em xin chân thành cảm ơn thầy Nguyễn Thanh Hải đã giúp đỡ chúng em trong quá trình làm bài tập lớn này. Chúng em rất mong nhận được những nhận xét và được chỉ ra những thiếu sót, những chỗ mà chúng em chưa làm tốt. Nhóm em xin cảm ơn!

# 

# TÀI LIỆU THAM KHẢO

**Tài liệu tiếng việt**

Nguyễn Thanh Hải, Nguyễn Tuấn Tú, Trần Thanh Huân, *giáo trình nguyên lý hệ điều hành* – Trường đại học Công Nghiệp Hà Nội

**Tài liệu tiếng anh**

Abraham Silberschatz (1978), University of Texas at Austin and James L. Peterson, *Microelectronics and computer technology Corporation*, Operating System Concepts.