

**HỆ ĐIỀU HÀNH**

- Kernel:** chức năng chính đạo, ghi dữ liệu bộ nhớ, tiến hành xử lý các lệnh, xác định cách dữ liệu được nhận và gửi thông qua các thiết bị khác
- API:** giao diện lập trình ứng dụng cho phép các lập trình viên phát triển ứng dụng có thể viết mã module
- Giao diện người dùng (User Interface):** cho phép hệ thống tương tác với người sử dụng bằng một command line hoặc các graphical icons và một desktop
- 3 HOẠT ĐỘNG CHÍNH CỦA OPERATING SYSTEM:** Quản lý tiến trình, Luồng và quản lý đồng bộ và Cơ chế lập lịch
  - +Vai trò:** 1. Cung cấp giao diện ảo trừu tượng 2. Quản lý, cấp phát tài nguyên và một cách cóch đồng bộ, hiệu quả và an toàn 3. Tạo điều kiện thuận lợi và đơn giản hóa việc lập trình

**+PHÂN LOẠI:**

- Batch OS(tho lệ):** Chương trình sẽ nằm trên các thẻ và được đưa vào máy để > **tuần tự**.
- (+) Tốc độ nhanh hơn, (-): CPU rảnh trong quá trình nhập/xuất, thiếu sự tương tác với user

- Spooling – Queue:** Simultaneous Peripheral Operation On-Line Tốc độ CPU >> đầu đọc thẻ, máy in
- Ở cùng ra đời >> đầu đọc thẻ

- Multiprogramming(đa chương):** nạp đồng thời nhiều chương trình. Khi program hiện tại đang chờ I/O đợi xử lý thì OS sẽ đưa chương trình khác vào xử lý.
- (+): tăng năng suất. (-): OS phải xử lý các vấn đề lập lịch, cơ chế bảo vệ bộ nhớ

- Timesharing OS(Multitasking OS – đa nhiệm):** Thông qua **time slice** để trao quyền lần lượt sử dụng CPU cho các tiến trình với thời gian chuyển đổi diễn ra rất nhanh. Phức tạp hơn đa chương, có thêm chức năng: Quản trị và bảo vệ bộ nhớ, sử dụng bộ nhớ ảo.
- (+): Tốc độ chuyển đổi nhanh nên tạo cảm giác mỗi người dùng có một máy ảo riêng. (-): CPU scheduling phức tạp => các hđh ngày nay sử dụng cơ chế này

- Parallel OS(RTOS):** nhiều bộ xử lý (CPUs), chia thành các cviệc nhỏ hơn, chia sẻ bus truyền dữ liệu, đồng bộ, bộ nhớ và các thiết bị ngoại vi.
- (+): Tăng suất mạnh và độ tin cậy (Sử hàng hóa của 1 bộ xử lý để không làm "sụp" đến toàn bộ hệ thống). (-): Yêu cầu cấu bộ phần cứng & khó về lập trình

- Real-time OS (RTOS):** đòi hỏi khác که về thời gian ở kết quả đầu ra. • hard real-time system: ràng buộc thời gian cứng. VD: Quản lý hoạt động công nghiệp, đèn giao thông. • soft real-time system: Livestream, media

- Distributed OS:** Tương tự timesharing nhưng các CPU số hữu bộ nhớ, đồng bộ **riêng**. Hđh phải hợp nhiều máy tính khác nhau -> chia sẻ tài nguyên, liên liên lạc. (+): bền vững 1 máy hư thì vẫn bth, tăng tốc độ.

- Embedded OS:** Hệ thống nhúng, không cần tương tác nhiều, được áp dụng trên điện thoại và các thiết bị khác (-): Tài nguyên thiếu, thuật toán phức tạp

**+KIẾN TRÚC HDH:**

- Đơn giản:** Tất cả (ứng dụng, thư viện, nhân HDH) chung không gian địa chỉ.
- (+) Xử lý nhanh, Dễ phát triển, mở rộng. Phù hợp với hệ đơn người dùng
- (-) Không có bảo vệ giữa nhân và ứng dụng. Dễ xảy ra xung đột khi mở rộng

- Monolithic Kernel:** Các đvụ hđh đều được **thực hiện vào kernel** (+) Hiệu năng cao, dễ hiểu với ltvien, có thể giao ứng dụng và nhân

- (-) Không có bảo vệ nội bộ giữa các thành phần trong nhân, khó mở rộng

- Microkernel (Client-Server):** nhân cực nhỏ, chỉ quản lý tối thiểu (giao tiếp, bộ nhớ, CPU), **giao tiếp qua message**
- (+) Dễ mở rộng, độc lập, dễ bảo trì, một dịch vụ lỗi không làm treo máy hệ thống

- (-) Hiệu năng thấp hơn do **giao tiếp thông điệp**, khó triển khai, lập trình phức tạp

**NHÂN CỦA HỆ ĐIỀU HÀNH**

- Là thành phần trung tâm của hầu hết các hđh
- Các module của hệ thống luôn có mặt trong bộ nhớ trong
- Được đặt ở các vùng biên có bộ nhớ trong tại vùng biên dưới
- 2 loại: Nhân bất buộc phải có ở bộ nhớ trong và Nhân rời cấu thiết mới gọi vào

- Các module của nhân: loader, monitor (lựa chọn các bước làm việc), scheduler

**BAO VỆ VÀ BẢO MẬT**

- Nguy cơ (Khách quan) thiên tai, lỗi sử dụng, lỗi phần cứng, phần mềm. (Chủ quan) Tăn công phá hoại (virus, worm, DoS), ăn cắp tài nguyên (trojan horses, trap doors, man-in-the-middle)

- Bảo vệ (protection) và Bảo mật (security): kiểm soát quá trình truy xuất tài nguyên và phòng thủ, chống lại các tấn công.

- Một số cơ chế: Hoạt động ở 2 chế độ (kernel vs. user mode), Sao lưu (Backup), Thao tác người dùng (User Authentication), Phân quyền (Authorization), chính sách bảo mật (Policy), Kiểm soát nhật ký (Audit log).

**+ QUÁ TRÌNH KHỞI ĐỘNG** - 3 giai đoạn

- 1) CPU thực thi lệnh tải địa chỉ boot trước (boot ROM)
  - x86: 0x7FFF0 tới tới BIOS
- 2) Firmware nạp boot loader;
  - Fw là phần chip CMOS
  - x86: ktra cấu hình CMOS (complementary metal oxide semiconductor); nạp trình quy ngắ và các trình khởi thiết bị; khởi tạo thanh ghi và qly nguồn (power); qly ktra phần cứng (POST – power-on self-test); hiện thiết lập hệ thống; xđịnh thiết bị có khả năng khởi động; tiếp tục quá trình khởi động.
- 3) Boot loader nạp HDH.
  - Lúc này HDH vẫn chưa chạy
  - Boot loader hiện đầ nhiều HDH khác, nhiều phiên bản HDH và hỗ trợ dual-boot.

**QUẢN LÝ TẬP TIN**

**1. DIATỤ:**

- 1. Khái niệm:** đầ bằng tổng hợp tín/kim loại có phủ lớp từ để lưu data. Mỗi liên **đọc/ghi ít nhất 1 sector**.

- Truy xuất:** vị trí sector góc (sector, track, head). Chỉ số được đánh như sau: Head (trên xuống 0), Track (ngoài vào, 0), sec (ngược chiều quay, 1)

- Access Time = Seek time + Rotational time + Read time**  
= 1/2 \* (Số Cylindar \* tg di chuyển qua mỗi track + tốc độ quay vòng/s + tổng tg xoay)

- Nhận xét:** truy xuất sector theo từng cylinder sẽ đảm bảo sau khi truy xuất sector K thì truy xuất sector K+1 là nhanh hơn so với tất cả các sector khác

- (\*) **Đĩa mềm: 1.44MB**. Có 2 head, 80 track/head, 18 sector/track.

- (\*) Cylinder: các track có cùng R nằm trên các sides.→ Tổ chức logic: đầu sector được đánh số theo từng cylinder bắt đầu từ 0. Tổ chức logic đánh số từ 0→2879.

- (\*) Quy ước:

**Sector vật lý – Sector logic:**

- Sector logic→Sector vật lý:**  
$$s = t \cdot st \cdot hd + h \cdot st + s - 1$$

- Sector logic→Sector vật lý:**  
$$s = (l \cdot mod \cdot st + 1) \cdot t = l \cdot div \cdot (st \cdot hd) \cdot h = (l \cdot div \cdot st) \cdot mod \cdot hd$$

- l:** sector logic; **s:** sector vật lý; **st:** số sector / track; **t:** track; **h:** head; **hd:** số head (số side)

$$= 36t + 18h + s - 1 \text{ (đĩa mềm)}$$

**Tổ chức hệ thống tập tin trên đĩa từ**

- Master boot record (MBR) thường nằm tại sector logic 0, kích thước 512 bytes, là đoạn trình khởi động hệ thống, có bảng nhân viên.

- Phần vùng (partition) gồm primary và extend, có tối đa 4 phần vùng.

- Boot block + Super block (Boot sector) chứa thông số quan trọng của phần vùng, chứa đoạn trình nạp HDH khi khởi động.

**Quá trình khởi động từ đĩa từ**

- 1) POST (Power-on self-test)
- 2) Tải MBR đọc thông tin bảng phần vùng, tìm phần vùng "active". Nếu k tìm thấy phần vùng "active", MBR có thể tải một boot loader và chuyển khiển cho nó. Boot loader này cho phép chọn HDH trên 1 phần vùng.

- 3) Chuyển quyền khiển về cho đoạn mã chương trình nằm trong Boot Sector của phần vùng được chọn.

**THUẬT TOÁN ĐỌC ĐĨA**

- First Come First Serve – FCFS:** Xử lý các yêu cầu theo đúng thứ tự thời gian đến.

- Ưu điểm:** Dễ cài đặt, công bằng. **Nhược điểm:** Hiệu suất kém nếu yêu cầu bị phân tán (có thể gây nhiều di chuyển đầu đọc).

- Shortest Seek Time First – SSTF:** Xử lý yêu cầu gần đầu đọc hiện tại nhất.

- Các gây starvation**
- SCAN – Thuật toán thang máy:**

- Đầu đọc di chuyển theo một hướng (tăng hoặc giảm track), phục vụ tất cả yêu cầu trên đường đi, đến khi chạm rai thì quay lại.

- Ưu điểm:** Công bằng hơn SSTF, không gây starvation. **Nhược điểm:** Có thể gây delay cho yêu cầu gần nhưng ở chiều ngược.

- C-SCAN:** Giống SCAN nhưng khi chạm rai cuối (max track), đầu đọc quay về đầu (min track) mà không xử lý yêu cầu trên đường về. **Ưu điểm:** Cân bằng hơn khi đến yêu cầu.

- LOOK:** Giống SCAN nhưng không đi đến rai địa, chỉ đi đến yêu cầu xa nhất trong hướng đó rồi quay lại. **Ưu điểm:** Tối ưu hơn SCAN, tránh đi "vô ích" đến cuối đĩa nếu không cần.

- LOOK:** Giống C-SCAN, nhưng chỉ quay lại **yêu cầu nhỏ nhất**, không quay về track 0 nếu không cần. **Ưu điểm:** Hiệu quả và nhanh hơn C-SCAN.

**II. HỆ THỐNG TẬP TIN:**

- 1. **Tập tin:** Là một khái niệm trừu tượng, một đơn vị lưu trữ ở mức logic được hệ điều hành cung cấp để người dùng có thể thông qua các khái niệm trừu tượng này thao tác các tén đồ. Lưu trữ tập các thông tin có liên quan với nhau. Là một đơn vị lưu trữ luận lý che cho chức vật lý của các thiết bị lưu trữ ngoài.

- Gồm thuộc tính và nội dung. Mỗi hệ thống tập tin có cách tổ chức khác nhau.

- h. Loại tập tin:** - Ordinary file - Directory file: Thư mục chứa các thư mục khác - Shortcut file - Special file (Device file): Thư mục đại diện cho một thiết bị ngoại vi nào đó (USB, Máy in,...)

- g. Truy xuất tập tin:** **Tuần tự** (đọc a→x, có thể quay lại sequential). **Ngẫu nhiên** (có thể seek đến vị trí cần đọc). **Indexed Sequential** (Linus/Inode): Mỗi block dữ liệu có 1 index.

- đ. Thu mục:** là loại tập tin đặc biệt. Nội dung của thư mục: **một cấp** (tất cả file có cùng thư mục), **hai cấp** (mỗi người dùng 1 thu mục), **cây phân cấp** (sự dụng hiện nay). Model of access: R, W, X(see).

- g. Các chức file:** Do HDH hoặc crình quyết định, có thể có hoặc không có các chức.

- Dãy các byte/words không có cấu trúc (Window, Linux) - Dãy các bản ghi fixed-length (Mainframes etc.) - Tree of records (Hệ thống lớn cho mục đích thương mại có nhu cầu quản nhóm các dữ liệu tương tự nhau)

- 2. Thu mục**
- Loại tập tin đặc biệt, giúp tổ chức có hệ thống các tập tin. Thuộc tính của thư mục như sau: Head (trên xuống 0), Track (ngoài vào, 0), sec (ngược chiều quay, 1)

- Thao tác trên con nó.
- Tạo, xóa, mở, đóng, liệt kê nội dung, tìm kiếm tập tin, duyệt hệ thống tập tin.

- Yêu cầu, mở, đóng, liệt kê nội dung, tìm kiếm tập tin, duyệt hệ thống tập tin.

**VĂN ĐỀ TỔ CHỨC HỆ THỐNG**

- Tổ chức thư mục**
- Thường được tổ chức thành một bảng các phần tử (directory entry) gọi là bảng thư mục. Tổ chức 2 cách: entry chứa tên và thuộc tính, hoặc entry chứa tên và pointer tới object.

- Vấn đề tên đại: Tên file ==> Pointer tới heap chứa tên file.

- Tổ chức tập tin**
- Mỗi tập tin lưu nội dung trên một số block của thiết bị lưu trữ. Phương pháp cấp phát mã là cách thực cấp phát các block cho các tập tin, gồm:

- Cấp phát liên tục:** Mỗi tập tin chiếm các block liên tục trên đĩa → Chi block bất đầu và chiều dài. Hỗ trợ **truy xuất tuần tự và trực tiếp**. Vấn đề external fragmentation và việc thực tế lớn

- Cấp phát theo extent → tập các block liên tục : cấp phát theo tính ext, mỗi tập tin có thể chứa các discrete ext, sizeof(ext) có thể khác nhau.

- Quản lý các thông tin: block bắt đầu, số block, con trỏ tới block đầu tiên của ext kế tiếp.

- Vấn đề internal và external fragment.

- Cấp phát linked list:** Mỗi tập tin chiếm một tập các block theo kiểu LL, các block sẽ nằm rải ra trên đĩa, **chỉ hỗ trợ truy xuất tuần tự**.

- Chỉ cần vị trí bắt đầu, không bị external fragmentation, tồn chỉ phí lưu địa chỉ trong LL, một node như → hư hết. **Enhanced ver:** tất cả con trỏ của LL lưu vào File-Allocation Table (FAT).

- Cấp phát theo kiểu chỉ mục:** Gồm các block làm bảng chỉ mục chứa địa chỉ các block dữ liệu. Hỗ trợ **truy xuất tuần tự và trực tiếp**. Tồn không gian để lưu block chỉ mục, không bị external fragmentation.

- Mở rộng: chỉ mục nhiều cấp, chỉ mục kết học linkedlist..

- INDEXED ALLOCATION:** Indexed Allocation (I-nodes) dùng multilevel index.

**QUẢN LÝ ĐỊA TRỌNG**

- Bit vector (bit map): Mỗi block biểu diễn 1 bit. Bit[i] = 0 khi block trống và ngược lại. Bit vector tồn không gian đĩa, dùng trong HDH Macintosh.

- Ds liên kết: k tón gian đĩa, chỉ phí duyệt ds cao.

- Grouping: chứa ds block trống, để tìm lượng lớn block trống

- Counting: chứa địa chỉ block trống đầu tiên và số lượng block trống liên tục kế tiếp.

**TỔ CHỨC HỆ THỐNG FILE TRONG BỘ NHỚ**

- Thao tác nhiều tập tin cùng lúc, thao tác cùng 1 tập tin cùng lúc => thông tin cần lưu trữ:

- Mounted Volume Table** – ds các volume được sử dụng trên hệ thống.

- Directory Structure** – Thông tin các folder mới được sử dụng (ptr trỏ tới volume tương ứng).

- System-wide open-file Table** – ds các file đang được mở trên hệ thống. (ptr tập tin, định vị file trên đĩa; quyền truy cập; biến đếm file đang mở).

- Per-process open-file Table** – ds các file mà tiến trình đang thao tác (ptr trỏ tới file đang mở tương ứng trong system-wide open-file table)

- Kết buộc (mount) hệ thống tập tin**
- Một hệ thống tập tin phải được kết buộc (mount) trước khi có thể truy xuất (như file phải được mở trước khi sử dụng). Các HDH thường phân hiện và tự động kết buộc các hệ thống tập tin tồn tại trên hệ thống (Win mount vào ổ đĩa, Linux mount vào folder

**Master boot sector**

- Nội lưu trữ thông tin về các partition trong ổ đĩa

- 0x0: Đoạn mã khởi động
- 0x1B8: Dấu hiệu nhân định đĩa
- 0x1BC: Không dùng, all 0
- 0x1BE: Bảng phân vùng
- 0x510, 0x511: Dấu hiệu MBR

- Mỗi 16 bits trong 0x1BE, 1 byte đầu (80 khởi động, 00 k động), 3 byte kế thông số vly (tsv1), 1 byte kế partition (06, 0B là fat; 07 là ntfs; 83 là linux; 0F là extended), 3 byte kế tsv1; 4 byte kế partition: 4 byte cuối kích thước partition (bnhư sector).

- 07 – NTFS, 83 – Linux, 0F – Extended, 06, 0B – FAT, 80 – Khởi động – Không khởi động.

- Boot sector**
- Gồm một số sector của partition. Sector đầu tiên chứa các thông số quan trọng của phần vùng và một đoạn trình nạp HDH khi khởi động. Sector còn lại chứa các thông tin hỗ trợ cho xđịnh thức số cluster trống & tìm kiếm cluster trống hiệu quả, chứa một sector bản sao của boot sector.

- Boot sector => FAT1 FAT2 → RDET → DATA

Trạng thái	Gửi trên FAT32	Note
Trống	0	= FREE
Hư	0FFFFFFF7	= BAD
Cluster cuối	0FFFFFFF	= EOF
Nội dung file	2.0FFFFFFF	

#	s	mean
0	3	lệnh nhảy đến đầu đoạn mã boot
3	8	tên công ty / version của hđh
B	2	số byte của sector (512)
D	1	số sector của cluster $S_C$
E	2	số sector của boot sector $S_B$
10	1	số lượng bảng fat $N_F$ (2)
11	2	số entry của rdet $S_r$ (fat16: 512)
13	2	số sector của volume $S_V$ (0 if > 65535)
15	1	ki hiệu lưu volume
16	2	số sector của 1 bảng fat $S_F$
18	2	số sector của track
1A	2	số lượng đầu đọc (side)
1C	4	k/cách từ nơi mà ta vol đến đầu vol
20	4	số sector của volume (if #13 > 65535)
24	1	k/hiệu vly của đĩa (0: mềm, 80h: cứng)
25	1	danh riêng
26	1	ký hiệu nhân diện hđh
27	4	serialnumber của volume
2B	b	volume label
36	8	loại fat, là chuỗi "fat12" hoặc "fat16"
3E	1cf	đoạn trình boot nạp tiếp hđh

- FAT12 quản lý tối đa 4078 (FEEH) cluster
- FAT16 quản lý tối đa 65518 (FEEH) cluster

- Nếu cluster quá > dùng FAT32

#	s	mean
0	3	lệnh nhảy đến đầu đoạn mã boot
3	8	tên công ty / version của hđh
B	2	số byte của sector (512)
D	1	số sector của cluster $S_C$
E	2	số sector của boot sector $S_B$
10	1	số lượng bảng fat $N_F$ (2)
11	2	Không dùng
13	2	Không dùng
15	1	ki hiệu lưu volume
16	2	Không dùng
18	2	số sector của track
1A	2	số lượng đầu đọc (side)
1C	4	k/cách từ nơi mà ta vol đến đầu vol
20	4	số sector của volume (if #13 > 65535)
24	4	số sector trong 1 bảng FAT
28	2	Bit 8 bật: chỉ ghi vào bảng FAT active
2A	2	Ver của FAT32
2C	4	Cluster bắt đầu của RDET
30	2	Sector chứa thông tin phụ
32	2	Sector chứa bản đầu của BS
34	C	danh riêng
40	1	ki hiệu vật lý của đĩa chứa volume
41	1	danh riêng
42	1	ki hiệu nhân diện của HDH
43	4	Serialnumber của vol
47	B	Vol label
52	B	Loại FAT, là chuỗi "FAT32"
5A	1A4	Đoạn trình khởi đ
1FE	2	Dấu hiệu kết thúc BS (AA55h)

**Bảng thư mục gốc**

- RDET – Root Directory Entry Table**
- Nằm trên vùng hệ thống (FAT12 & FAT16) hoặc nằm trên vùng dữ liệu (FAT32). Gồm một dãy các phần tử (gọi là entry), mỗi entry có kích thước 32 bytes chứa các thông tin của 1 file hoặc folder.

- Số sector của RDET:

số entry × 32		
$S_{RDET}$ = số byte của sector (512)		
Byte đầu mỗi entry:	entry trống (0), tập tin ở chấm entry đã xóa (E5), thông tin file / folder (giá trị còn lại).	
Entry chứa thông tin của tập tin		
#	s	mean
0	8	Tên chính
8	3	Tên mở rộng
B	1	Thuộc tính trạng thái
C	1	Danh riêng
D	3	Giới tạo
10	2	Ngày tạo
12	2	Ngày truy cập gần nhất
14	2	Cluster bắt đầu – phần word cao
16	2	Giới sửa gần nhất
18	2	Ngày cập nhật gần nhất
1A	2	Cluster bắt đầu – phần word thấp
1C	4	Kích thước của phần nội dung tập tin

- 0: readonly, 1:hidden, 2: system, 3: vollabel, 4: directory, 5: archive

Entry phụ chỉ chứa tên tập tin			
#	s	mean	
0	1	Thứ tự của entry	
1	A	5 kí tự Unicode	
B	1	Dấu hiệu nhận biết (luôn là 0F)	
E	C	6 kí tự tiếp t theo	
1C	4	2 kí tự tiếp theo	

**Vùng dữ liệu**

- Mỗi phần tử gọi là cluster, có kích thước  $S_C = 2^n$  sector, tùy vào người dùng format. **Đánh stừ 2.**
- Cluster thứ k sẽ bắt đầu tại sector:

$$S_B + S_F \cdot N_F + S_{RDET} + (k - 2) S_C$$

**Bảng thư mục con**

- SDET – Sub Directory Entry Table**
- Nằm trên vùng dữ liệu, có cấu trúc hoàn toàn giống bảng thư mục gốc.

- Mỗi SDET luôn có 2 entry. v và .. đầu bảng mô tả về chính thư mục này và thư mục cha của nó.

**Thao tác trên FAT**

- Đọc nội dung (type):** Xđịnh entry chính chứa t/in thư mục dựa vào phần tên và phần mở rộng. Từ entry chính được cluster bắt đầu. Dựa vào bảng fat tìm các phần tử còn lại. Đọc các sector nội dung của bảng fat tìm các phần tử còn lại.

- Liệt kê nội dung (dir):** Xđịnh entry chính chứa t/in thư mục dựa vào phần tên. Từ entry chính tìm được cluster bắt đầu. Dựa vào bảng fat tìm các phần tử còn lại. Đọc các sector nội dung tìm được tổng entry (32 bytes) và hiển thị tin.

- Tạo tập tin (copy con):** Tìm đủ số entry trống liên tiếp trên bảng thư mục. Ktra bảng fat còn đủ cluster trống k. Lưu vào các entry trống tìm dc. Ghi giá trị vào các phần tử fat trống, lưu nội dung tập tin vào các cluster tương ứng.

- Tạo thư mục (MD):** Tìm đủ số entry trống liên tiếp trên bảng thư mục. Ktra bảng fat còn đủ cluster trống k. Lưu vào các entry trống tìm dc. Ghi giá trị vào các phần tử fat trống, tạo thư mục. v và .. chiếm 2 entry đầu tiên.

- Xóa tập tin (delete):** Xđịnh entry chính trong bảng thư mục chứa t/in tập tin dựa vào phần tên và phần mở rộng. Đặt giá trị E5h vào byte đầu tiên của entry chính và tất cả các entry phụ. Từ entry chính được cluster bắt đầu. Dựa vào bảng fat tìm các phần tử còn lại. Đặt tất cả các phần tử FAT của tập tin về giá trị 0.

- Xóa thư mục (rd):** Xóa đệ quy tất cả thư mục và tập tin con. Thư mục rỗng xóa như xóa tập tin.

- Sao chép tập tin (copy):** Tìm đủ số entry trống liên tiếp trên bảng thư mục. Ktra bảng fat còn đủ cluster trống k. Copy t/in entry của tập nguồn sang entry tập đích. Ghi giá trị vào các phần tử fat trống, copy các sector nội dung tập nguồn vào các sector nội dung tập đích.

- Di chuyển tập tin (move):** Tìm đủ số entry trống liên tiếp trên bảng thư mục. Copy t/in entry của tập nguồn sang entry tập đích. Xóa thông tin của tập tin nguồn.

- Đổi tên tập tin (ren):** Xđịnh entry chính trong bảng thư mục chứa t/in tập tin dựa vào phần tên và phần mở rộng. Nếu tên tập tin không cần thêm các phụ phụ, cập nhật lại phần tên và phần mở rộng. Nếu tên tập tin cần thêm các entry phụ, tìm đủ số entry trống liên tiếp nhau trên bảng thư mục để chứa t/in đích, copy t/in entry của tập nguồn sang entry tập đích.

- Quick format:** Giữ lại các thông số cũ của phần vùng. Cập nhật lại trạng thái các cluster đang chứa dữ liệu thành trống và cho tất cả entry trên bảng thư mục mức gốc về trạng thái trống. Tương đương với việc xóa tất cả mọi tập tin & thư mục nhưng nhanh hơn xóa một tập tin.



