|  |  |  |  |
| --- | --- | --- | --- |
| **\*ĐN HĐH**: -Một lớp phần mềm ở giữa phần cứng và các chương trình ứng dụng/người dùng, nó cung cấp một giao diện máy ảo (virtual machine) : dễ dàng và an toàn.  -Một bộ quản lý tài nguyên (resource manager) cho phép các chương trình/người dùng chia sẻ tài nguyên phần cứng: công bằng và hiệu quả.  -Một tập các tiện ích để đơn giản hóa việc phát triển ứng dụng. | -**Cơ chế:** cấu trúc dữ liệu và các thao tác trên một đối tượng nào đó (v.d. buffer cache).  -**Chính sách:** các phương thức đưa ra quyết định khi phải chọn lựa (v.d. chính sách thay thế buffer cache)  -Các hệ điều hành khác nhau có thể cần các chính sách khác nhau | **Tiến trình** là một khái niệm trừu tượng: Tạo ảo giác như là công việc duy nhất trong một hệ thống.  **Cơ chế**: Tạo, hủy, dừng, context switch, báo hiệu,..  **Chính sách**: -Ai có thể tạo/ hủy/ dừng tiến trình?  -Mỗi người dùng có thể chạy bao nhiêu tiến trình đồng thời?  -Làm sao chia sẻ tài nguyên giữa các tiến trình? | **Tiểu trình** đ.trưng cho một bộ xử lý: Tạo ảo giác như là có riêng 1 bộ xử lý cho một thực thi nhất định.  **Cơ chế**: Tạo, hủy , dừng, context switch, báo hiệu, đồng bộ..  **Chính sách**: Làm sao chia sẻ CPU giữa các tiểu trình thuộc các tiến trình khác nhau? -Làm sao chia sẻ CPU giữa các tiểu trình thuộc cùng một tiến trình? -Làm sao để đồng bộ các tiểu trình với nhau? -Làm sao điều khiển sự liên lạc giữa các tiểu trình? |
| **Bộ nhớ logic** đ.cho bộ nhớ: Tạo ảo giác vùng nhớ lớn liên tục, thường là nhiều hơn bộ nhớ vật lý thật sự.  **Cơ chế:** Bộ nhớ lôgic <=> bộ nhớ vật lý, lỗi trang (page-fault),..  **Chính sách:** -L.sao để biến bộ nhớ ảo trông lớn hơn bộ nhớ vật lý thật dựa trên tài nguyên có sẳn? -L.sao cấp phát bộ nhớ vật lý cho các tiến trình đang tranh giành nhau -L.sao điều phối việc chia sẻ bộ nhớ vật lý giữa các tiến trình? | **Hệ thống tập tin** b.trưng cho sự lưu trữ: tạo ảo giác một không gian lưu trữ có cấu trúc.  **Cơ chế:** -Tạo tập tin, xóa, đọc, ghi, ánh xạ file-block-to-disk-block, cache vùng đệm của tập tin, ...  **Chính sách:** -Chia sẻ so với bảo vệ? Cấp phát những block nào? Quản lý cache vùng đệm của tập tin? | **Truyền thông điệp** biểu trưng cho việc **truyền thông**: Tạo ảo giác sự truyền thông tin cậy.  **Cơ chế:** -Gửi, nhận, bộ đệm, truyền lại,…  **Chính sách:**  Điều khiển tắt nghẽn và dẫn đường  Đảm bảo đa nối kết trên cùng một card mạng | **Các chủ đề chính trong thiết kế hệ điều hành:** Giao diện lập trình, Quản lý tài nguyên, Chia sẻ, An toàn, Truyền thông, Cấu trúc, Đồng thời, Thực thi, Tin cậy, Bền vững, Sổ sách, Phân tán, Tính phát triển(scaling). |
| **Cuộn (Spooling)**: Đọc cv 1 từ card vào đĩa. Thực thi cv 1, trong khi đó đọc cv 2 từ cards vào đĩa; lưu kết quả cv 1 vào đĩa. In kết quả công việc 1, trong khi đó thực thi cv 2 và đọc cv 3 từ card vào đĩa. Và tiếp tục như vậy …  -S**pooling**:Simultaneous Peripheral Operation On-Line  *LOAD 5,2: lấy gtri ở ô nhớ 5 gán vào giá trị ô nhớ 2*  LOAD [5], 2: ô nhớ 2 trỏ vào vị trí giá trị ô nhớ 5: giá trị ô nhớ 5 =100, ô nhớ 2 đến vị trí 100 và lấy giá trị ở ô nhớ đó (gián tiếp) | **Đachương(Multiprogramming):** Nạp nhiều chương trình vào đĩa cùng một thời gian (sau này là vào bộ nhớ). Chuyển sang công việc kế tiếp nếu công việc hiện thời đang thực hiện lệnh I/O. Thiết bị ngoại vi thường chậm hơn trên đĩa (hay bộ nhớ). Đồng thời thực hiện I/O của chương trình này và tính toán cho chương trình khácThiết bị ngoại vi phải là bất đồng bộ. Phải biết khi nào công việc I/O xong: ngắt vs.polling->Tăng khả năng phục vụ của hệ thống, có thể tốn nhiều thời gian hơn để phản hồi. | **Chia sẻ thời gian (Time-Sharing):** Sự khác biệt chính giữa đa chương trình và chia sẻ thời gian là đa chương trình là việc sử dụng hiệu quả thời gian của CPU, bằng cách cho phép một số chương trình sử dụng CPU cùng một lúc nhưng chia sẻ thời gian là một số người dùng muốn sử dụng cùng một cơ sở. | **Hệ điều hành song song (Parallel OS):** Vài ứng dụng có các công việc có thể thực hiện đồng thời: Có thể tăng tốc độ bằng cách chạy các công việc trên các bộ xử lý khác nhau song song đồng thời. Cần HĐH và ngôn ngữ lập trình hỗ trợ chia nhỏ công việc thành các hành động song song. Cần HĐH hỗ trợ đồng bộ và truyền thông nhanh.  Nhiều kiến trúc song song khác nhau  Khả năng thực thi và tính mở rộng |
| **HĐH thời gian thực (Real-Time OS)**: Thực thi các ứng dụng có thời hạn cho trước  **Hard real-time system**: Hệ thống điều khiển bay, các hệ thống điều khiển công nghiệp,…Gây thảm họa nếu ta trễ hạn. Thách thức là làm sao không trễ hạn mà không phung phí nhiều tài nguyên.  **Soft real-time system:** Ứng dụng multimedia. Có thể gây khó chịu nhưng không đến nỗi thảm họa nếu bị quá hạn đôi chút! Thách thức là làm sao không trễ hạn mà không phung phí nhiều tài nguyên  Thử thách ở chỗ là khi hệ thống quá tải | **HĐH phân tán (Distributed OS)**  **Clustering:** Dùng nhiều máy nhỏ để phục vụ các công việc lớn.Rẻ hơn là dùng một máy tính lớn. Độ tin cậy cao hơn, tăng khả năng mở rộng. Hệ thống phân tán trong khu vực rộng. Cho phép sử dụng tài nguyên phân tán. *V.d: sử dụng PC để truy cập Web.* Không cần mang nhiều thông tin cần thiết theo mình. Cần HĐH hỗ trợ truyền thông và chia sẻ tài nguyên phân tán. *V.d., hệ lưu trữ tập tin trên mạng.* Quan tâm tính thực thi (mặc dù tăng tốc không phải là mục tiêu của HĐH này), độ tin cậy cao, sử dụng nguồn tài nguyên đa dạng | **HĐH nhúng (Embedded OS):** Phát triển rộng khắp  Hiện thời, ĐTDĐ và PDAs  Tương lai, các thiết bị tính toán ở mọi nơi. Đặc tính:  -Tài nguyên hạn hẹp: CPU chậm, bộ nhớ nhỏ, không ổ đĩa, v.v.  -Chúng ta cần chạy những ứng dụng mạnh hơn!..  Làm sao ta có thể chạy các chương trình mạnh hơn khi mà phần cứng vẫn như trước đây?  Sử dụng nhiều thiết bị…  Tăng thêm các tiện ích trên thiết bị. HĐH giúp quản lý năng lượng, tính di động, tìm kiếm tài nguyên, ... | **Góc nhìn từ HĐH:** Máy tính là một thiết bị phần cứng cứ thực hiện xoay vòng “lấy lệnh(chậm nhất)-giải mã- thực thi” (fetch-decode-execute)  **Exceptions:** xảy ra khi CPU phát hiện một lệnh mà nó không  thể thi hành được  **Traps:** một lệnh đặc biệt nó gán PC trỏ tới 1 địa chỉ biết trước và  đặt mode thành system mode  **Interrupts:** là một sự kiện bên ngoài làm cho CPU nhảy tới  một địa chỉ biết trước  ---------------------------------------  **Polling**: thiết bị bị động, CPU phải đọc/ghi vào thanh ghi  **Interrup**t: báo hiệu cho CPU bằng interrupt |
| **Thiết bị I/O cơ bản**  Card mạng có 2 thanh ghi:  Một để chứa dữ liệu gửi ra đường truyền.Việc truyền thường được viết tắt **TX** (V.d. thanh ghi TX)  Một để chứa dữ liệu nhận từ đường truyền. Nhận thường được viết như là **RX.**  CPU truy cập các thanh ghi này như thế nào? ánh xạ chúng vào bộ nhớ  Một lệnh truy cập ô nhớ 98 nghĩa là truy cập thanh ghi TX  Một lệnh truy cập ô nhớ 99 nghĩa là truy cập thanh ghi RX  Các thanh ghi này được gọi là “ánh xạ bộ nhớ” (memory-mapped) | **Tại sao cần “ánh xạ bộ nhớ”**  Dùng bộ nhớ cho các thanh ghi thiết bị nhằm 2 tính năng:  Bảo vệ truy cập --- chỉ HĐH mới có thể truy cập thiết bị. (Chương trình ứng dụng phải thông qua HĐH mới truy cập, được thiết bị I/O vì các cơ chế bảo vệ đã thảo luận)  *HĐH điều khiển thiết bị, ghi hay đọc dữ liệu vào thiết bị*  *Không cần sự thay đổi tập lệnh*  *Gọi là programmed I/O*  **Programmed I/O:** CPU phải đọc/ghi trên thiết bị  **DMA**: thiết bị đọc và ghi trên bộ nhớ | **Thanh ghi trạng thái**  Để gửi, HĐH ghi 1 byte vào thanh ghi TX và chuyển bit thứ 0 trong thanh ghi trạng thái thành 1. Khi gửi xong byte này, nó lại gán bit thứ 0 của thanh ghi trạng thái trở về 0.  Khi card mạng nhận 1 byte, đặt byte này vào thanh ghi RX và đặt bit thứ 1 của thanh ghi trạng thái thành 1. Khi HĐH đọc xong dữ liệu, nó đặt bit thứ 1 của thanh ghi trạng thái trở về 0.  **Kích thước của dữ liệu <- truyền trở thành yếu tố quyết định để lựa chọn PIO hay DMA** | **Polling vs. Interrupts**  Polling: chúng ta cứ 10 giây check email 1 lần thử có email mới không  Interrupt: nếu có email mới, trình duyệt sẽ phát âm thanh báo. Nếu tất cả đều dùng polling!  Interrupts yêu cầu chi phí nhiều hơn cho quá trình xử lý.  **Polled IO vs. Truy cập bộ nhớ trực tiếp (DMA)**  -PIO làm ít việc hơn DMA: PIO chỉ kiểm tra thanh ghi trạng thái, rồi gửi hay nhận dữ liệu. DMA phải khởi tạo giá trị base, count, kiểm tra trạng thái và dùng interrupt.  DMA hiệu quả hơn trong việc truyền: PIO trói buộc CPU trong suốt thời gian truyền |
| Cho biết: Việc NẠP tốn 100 CPU “cycles”. GHI tốn 50 cycles. Xử lý một interrupt tốn 2000 lệnh, trung bình 1 lệnh tốn 2 cycles. Để gửi 1 gói tin dùng PIO tốn 1 NẠP(setup) + 1 GHI cho 1 byte. Nếu dùng DMA tốn 4 NẠP(setup) + 1 interrupt: **Giải**  PIO: setup+G\*B = 100 +50B  DMA:setup+interrup=4\*100+4000  100+50B = 4(100)+4000->B = 86 bytes (làm tròn). KL: Khi kích thước gói tin lớn hơn >86 bytes, DMA ít tốn CPU cycles hơn PIO. | **Các thiết bị I/O tiêu biểu**  **Ổ đĩa:** CPU hiểu như một mãng các blocks cùng kích thước.  **Card mạng**: Cho phép CPU gửi và nhận các gói tin thông qua mạng (có dây hay không dây)  Kích thước gói tin thông thường 64-8000 bytes  **Card màn hình**: CPU hiểu là một vùng nhớ mà có thể chuyển thành điểm ảnh trên màn  hình | **Bộ nhớ phụ:** Là bộ lưu trữ bên ngoài bộ nhớ chính. Không cho phép thực thi trực tiếp các instruction hoặc lưu trữ dữ liệu cho các instruction này.  **Đặc điểm: -**Kích thước lơn: ~ GB, TB. -Rẻ. -Dữ liệu được lưu trữ ngay cả khi không có nguồn cung cấp năng lượng. -Chậm: thời gian truy cập khoảng milliseconds | **Seek time**: thời gian di chuyển  đầu đọc tới track cần đọc  **Rotational delay:** thời gian để  tìm sector cần đọc khi mà đầu  đọc đã tìm được track  **Transfer rate:** tốc độ đọc/ghi  dữ liệu  *Thông thường:*  Seek: ~8-10ms  Rotational delay: ~4.15ms  với tốc độ 7200 rpm |
| **Ổ đĩa chậm hơn rất nhiều so với bộ nhớ chính: -**Tốc độ I/O chiếm vị trí quan trọng trong việc nâng cao khả năng thực thi của máy tính  -Thời gian truy cập (seek time+ rotational delay) >> thời gian copy một sector. Vì vậy thứ tự đọc các sector là ảnh hưởng nhiều đến thời gian I/O  **Lập lịch ổ đĩa: -**Thường là dựa trên vị trí của các sector hơn là độ ưu tiên của các tiến trình.  -Thay đổi thứ tự đọc/ghi các sector giúp tăng tốc độ thực thi  ------------------------------------------  **C-LOOK:**Một phiên bản của C-SCAN.Đầu đọc chỉ đi chuyển đến yêu cầu ở xa nhất về một phía, sau đó chuyển hướng, không cần phải d.chuyển về đến đầu cuối của ở đĩa. | **SSTF(shortest seek-time first):** Chọn sector có seek time nhỏ nhất từ vị trí đầu đọc hiện tại.  **SCAN:** Đầu đọc bắt đầu tại một đầu của ổ đĩa, và di chuyển tới đầu còn lại, phục vụ các yêu cầu trong lúc di chuyển, khi tới đầu kia thì quay ngược lại và tiếp tục phục vụ  các yêu cầu.  **C-SCAN:** Cung cấp thời gian đợi trung bình đều hơn so với SCAN.  Đầu đọc bắt đầu tại một đầu của ổ đĩa, và di chuyển tớiđầu còn lại, phục vụ các yêu cầu trong lúc di chuyển, khi tới đầu kia thì quay ngược lại vị trí bắt đầu của ổ đĩa, không phục vụ các yêu cầu trong lúc di chuyển ngược lại. | -SSTF có triết lý tự nhiên. -SCAN và C-SCAN tốt cho các hệ thống thường truy suất khối lượng dữ liệu lớn trên đĩa.  -Tốc độ thực thi còn tùy thuộc vào thứ tự và loại yêu cầu.  -Các yêu cầu từ ổ đĩa chịu ảnh hưởng từ các phương thức cấp phát bộ nhớ để lưu trữ file.  -Thuật toán lập lịch đĩa nên cài đặt riêng biệt với HĐH, cho phép HĐH có thể thay thế thuật toán khi cần thiết.  -Nên chọn thuật toán SSTF hoặc LOOK làm thuật toán mặc định. | **Quản lý ổ đĩa**  -Formatting cấp thấp, hay là Format vật lý — chia ổ đĩa thành  các sector mà disk controller có thể đọc và ghi.  **Để có thể lưu được các tập tin, HĐH cần phải lưu cấu trúc dữ**  **liệu của nó lên ổ đĩa.**  -Partition ổ đĩa thành một hoặc nhiều nhóm cylinders.  -Logical formatting hay “tạo hệ thống tập tin”.  **Boot block dùng để khởi động hệ thống.**  -Bootstrap được lưu trên ROM.  -Bootstrap nạp Bootstrap loader program.  **Các phương thức như là sector dự trữ dùng để xử lý các blocks**  **hư.** |
| **Swap-Space** - sử dụng không gian trên ở đĩa như là phần mở rộng của bộ nhớ chính. Swap-space có thể được lưu trên partition chứa file thông thường, cũng có thể trên một partition riêng biệt.  **Quản lý swap-space:**  **4.3BSD** tạo swap space khi tiến trình bắt đầu; lưu text segment (mã chương trình) và data segment. **Kernel** dùng swap maps để lưu vết sử dụng swap-space. **Solaris** 2 tạo swap space chỉ khi trang bị chép ra khỏi bộ nhớ chính, (không phải lúc các trang được tạo.)  **RAID Lv 4 5:**  **RAID 4: -**Dùng parity strip tương tự như RAID 3 (nhưng RAID 3 dùng parity bit)-Truy cập độc lập – vì vậy nhiều I/O có thể thực hiện song song | **RedundantArray of Inexpensive Disks (RAID):** HĐH sử dụng một bộ các ổ đĩa như là một ổ đĩa logic.-Thay thế ổ đĩa dung lượng lớn bằng nhiều ổ đĩa dung lượng nhỏ để tăng tốc độ I/O-Dữ liệu được phân phối trên các ổ đĩa sao cho có thể truy xuất dữ liệu đồng thời từ nhiều ổ đĩa-Hỗ trợ sao lưu để tăng khả năng đối phó việc mất mát dữ liệu, khi xác suất bị lỗi sẽ tăng khi chúng ta sử dụng nhiều ổ đĩa-Tăng tính sẳn sàng, vì khó có xác suất hệ thống hỏng nhiều đĩa  **Sáu cấp độ RAID đại diện cho các thiết kế khác nhau**  -Truy cập độc lập (riêng lẻ) ➜ write = 2 reads + 2 writes  Parity disk có thể bị vấn đề cổ chai (bottleneck) | **RAID Lv 0:** Không có sao lưu. Dữ liệu được lưu trữ dàn trãi trên các đĩa-Tổng dung lượng của các đĩa được chia thành các strip -Strips được đánh số xoay vòng trên các đĩa (round-robin)- Một tập các strip liên tiếp nhau mà mỗi strip nằm trên một đĩa được gọi là một stripe (một lớp)  **RAID 3:** chỉ dùng một đĩa lưu parity bits: P(i) = X2(i) (+) X1(i) (+) X0(i).Khi một đĩa bị hư, vẫn khôi phục được dữ liệu.Chỉ cho một đĩa hư tại một thời điểm  **RAID 5**: Giống RAID 4 nhưng parity strips được lưu phân phối trên các đĩa | **RAID Lv 1**: Sao lưu bằng cách nhân bản dữ liệu. Mỗi đĩa có một đĩa sao lưu, lưu y hệt dữ liệu của nó-Có thể đọc từ bất cứ đĩa nào trong 2 đĩa này (tăng tốc độ đọc nếu yêu cầu đọc nhiều)-Phải ghi trên cả 2 đĩa, tuy nhiên có thể ghi song song-Khôi phục đơn giản, nhưng chi phí cao  **RAID Lv 2 3:** Truy cập song song: tất cả các đĩa liên quan đến 1 yêu cầu I/O. Kích thước của mỗi lần read/write = số đĩa \* strip size  **RAID 2**: -error correcting code được tính dựa trên các bit tương ứng trên các ổ đĩa dữ liệu và lưu trên parity disks-Hamming code: có thể sửa lỗi 1 bit và nhận dạng 2 bit bị lỗi-Tốn ít chi phí hơn RAID 1 nhưng chi phí vẫn cao – không áp dụng trong thực tế |
| **Đĩa từ** - là những đĩa phẳng bằng thủy tinh hay bằng kim loại cứng được phủ từ để lưu dữ liệu  **Cơ chế đọc đĩa:** Access time = Seek time + Rotational time + Read time | **Cấu trúc vật lý:** -Gồm nhiều lớp hình tròn, mỗi lớp phủ từ 1 hoặc cả 2 mặt (side)-Mỗi mặt có tương ứng 1 đầu đọc (head) để đọc hoặc ghi dữ liệu-Mỗi mặt có nhiều đường tròn đồng tâm (track)-Mỗi đường tròn được chia nhỏ thành các cung tròn (sector), thông thường mỗi cung chứa 4096 điểm từ (~ 4096  bit = 512 byte)-Mỗi lần đọc/ghi ít nhất 1 sector (512 byte) | **Truy xuất mức vật lý:** Để truy xuất 1 sector cần phải chỉ ra vị trí của sector đó. Vị trí sector được thể hiện bằng 3 thông số: chỉ số sector, track và head: Head được đánh số từ trên xuống bắt đầu từ 0.Track được đánh số theo thứ tự từ ngoài vào bắt đầu từ 0.Sector được đánh số bắt | đầu từ 1 theo chiều ngược với chiều quay của đĩa  -Địa chỉ sector vật lý có ký hiệu: (sector, track, head)  -Hàm truy xuất mức vật lý trong C for DOS: int biosdisk (int cmd, int drive, int head, int track, int sector,int nsects, void \*buffer) |

**Boot Sector Interpretation**

• 00-02: eb 3c 90 Instructions to jump to boot code

• 03-0a: 4d 53 44 4f 53 35 2e 30

• Name string (MSDOS5.0)

• 0b-0c: 00 02 Bytes/sector (0x0200 = 512)

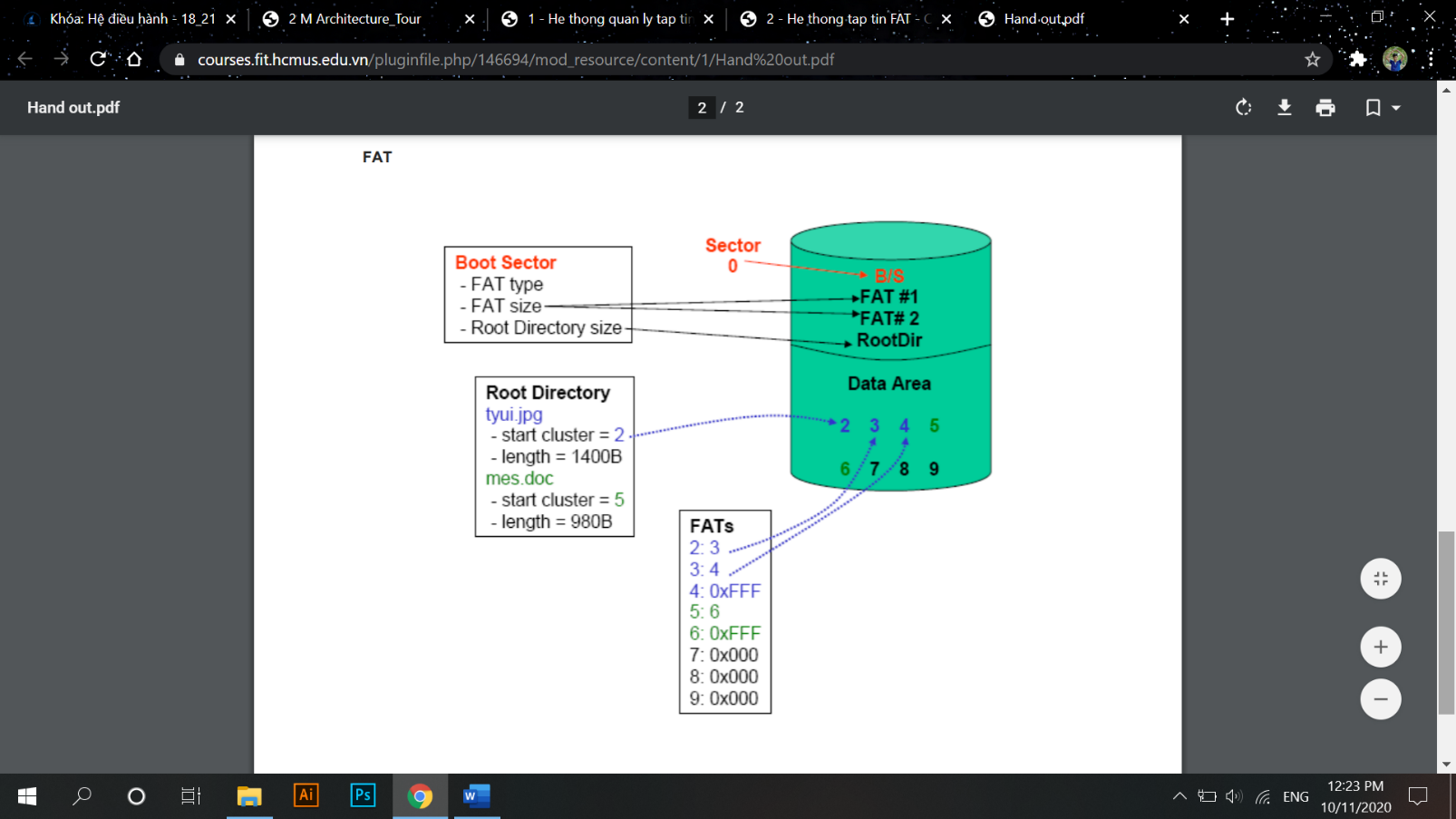
• 0d : 01 Sectors/cluster (1)

• 0e-0f: 01 00 Size of reserved area (1 sector)

• 10 : 02 Number of FATs (2) • 11-12: e0 00 Max. number of root directory entries (0x00e0 = 224)

• 13-14: 40 0b Total number of sectors (0x0b40 = 2,880)

• 15 : f0 Media type (removable)

• 16-17: 09 00 FAT size (0x0009 = 9 sectors)

• 18-19: 12 00 Sectors/track (0x0012 = 18)

• 1a-1b: 02 00 Number of heads (0x0002 = 2)

• 1c-1f: 00 00 00 00 Number of sector before partition (0)

• 20-23: 00 00 00 00 Total number of sectors (0 because 2B value not equal 0) • 24 : 00 Drive number (0)

• 25 : 00 Unused • 26 : 29 Extended boot signature

• 27-2a: cf cd b1 c4 Volume serial number (C4B1-CDCF)

• 2b-35: 4e 4f 20 4e 41 4d 45 20 20 20 20

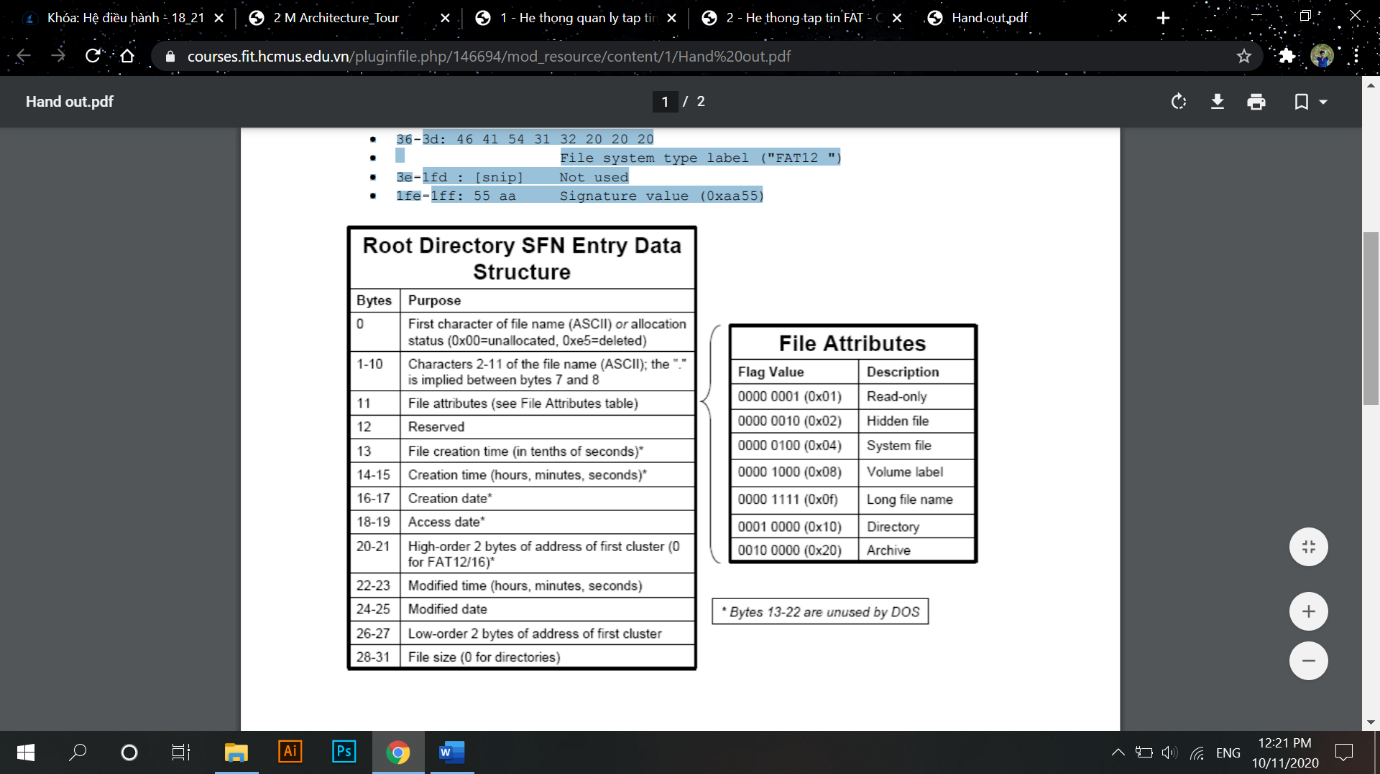
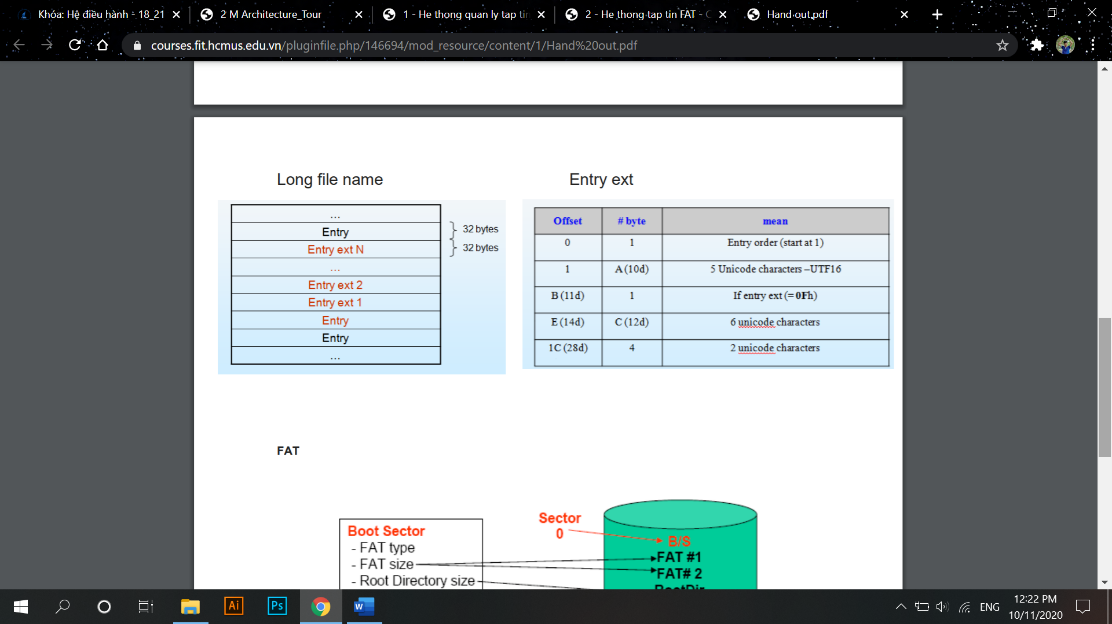
• Volume label ("NO NAME ")

• 36-3d: 46 41 54 31 32 20 20 20

• File system type label ("FAT12 ")

• 3e-1fd : [snip] Not used

• 1fe-1ff: 55 aa Signature value (0xaa55)



|  |  |  |  |
| --- | --- | --- | --- |
| **Tổ chức logic** Do truy xuất mức vật lý phải dùng đến 3 tham số rất bất tiện nên tổ chức logic được đưa ra để dễ hiểu, dễ thao tác, dễ tính toán hơn-Cylinder: là tập các track có cùng bán kính (cùng số hiệu) trên tất cả các mặt. **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 | -Tổ chức logic là một dãy sector được đánh chỉ số theo theo từng cylinder, bắt đầu từ 0 -Mỗi lần truy xuất (đọc/ ghi đĩa) chỉ có thể thực hiện trên N sector liên tiếp (N>=1) n Hàm truy xuất mức logic trong C for DOS: **int absread** (int drive, int nsects, long lsect, void \*buffer). **int abswrite** (int drive, int nsects, long lsect, void \*buffer); | **Sector vật lý « Sector logic**  -Sector vật lý =>Sector logic  l = t\*st\*hd + h\*st + s -1  -Sector logic =>Sector vật lý  s = (l mod st) + 1  t = l div (st \* hd)  h = (l div st) mod hd  Trong đó:  **l**: chỉ số sector logic **st** : số sector /track, **h**: chỉ số head **th**: số track /side (head), **t**: chỉ số track **hd**: tổng số side (head), **s:** chỉ số sector vật lý | **Một số hạn chế của bộ nhớ trong:** Không lưu trữ dữ liệu lâu dài. Không chứa lượng thông tin lớn.=>Cần các thiết bị lưu trữ ngoài(bộ nhớ ngoài) để lưu trữ dữ liệu  **-Tuy nhiên, có nhiều loại thiết bị lưu trữ ngoài (đĩa**  **từ, CD/DVD, USB, thẻ nhớ,…); đa dạng về cấu trúc,khả năng lưu trữ, phương thức truy xuất, tốc độ truy xuất** |
| **Tập tin:-**Lưu trữ tập hợp các thông tin có liên quan với nhau-Là một đơn vị lưu trữ luận lý che tổ chức vật lý của các thiết bị lưu trữ ngoài-Thường bao gồm 2 thành phần:Thuộc tính, Nội dung-Mỗi hệ thống tập tin có cách thức tổ chức tập tin khác nhau. | **Một số thao tác cơ bản trên tập tin:** Tạo, Xóa, Đọc, Ghi, Định vị (seek), Xóa nội dung (truncate), Mở, Đóng, khác: sao chép, di chuyển, đổi tên,..  +Hai cấp: mỗi người dùng có 1 thư mục riêng+Cây phân cấp: được sử dụng phổ biến hiện nay | **Thư mục** là một loại tập tin đặc biệt, giúp tổ chức có hệ  thống các tập tin trên hệ thống lưu trữ ngoài;-Thuộc tính của thư mục tương tự của tập tin-Nội dung của thư mục: quản lý các tập tin,thư mục con của nó: + Một cấp: đơn giản nhất, tất cả tập tin trên hệ thống cùng thư mục | **Một số thao tác trên thư mục:** Tạo, Xóa, Mở, Đóng, Liệt kê nội dung thư mục, Tìm kiếm tập tin, Duyệt hệ thống tập tin |
| **Tổ chức tập tin:** -Mỗi tập tin lưu nội dung trên một số block (khối lưu trữ) của thiết bị lưu trữ-Phương pháp cấp phát mô tả cách thức cấp phát các block cho các tập tin- Có 3 phương pháp cấp phát chính: Cấp phát liên tục  Cấp phát theo kiểu danh sách liên kết Cấp phát theo kiểu chỉ mục | **Quản lý không gian đĩa trống**  -**Bit vector (Bit map):** Mỗi block được biểu diễn bằng 1 bit. Bit vector tốn không gian đĩa. Ví dụ: kích thước 1 block = 212 bytes, kích thước đĩa = 230 bytes (1 gigabyte), n = 230/212 = 218 bits (or 32K bytes)  -HĐH Macintosh | **Danh sách liên kết** Chi phí duyệt danh sách cao. Không tốn không gian đĩa  **Grouping** Chứa danh sách các block trống. Dễ tìm một lượng lớn các block trống **Counting:** Chứa địa chỉ block trống đầu tiên và số lượng các block trống liên tục tiếp theo | **Hệ thống tập tin trên Unix /Linux Cấu trúc I-node**  Gián tiếp cấp 1: cấp này trỏ tới 256 địa chỉ. Tổng 256KB  - Gián tiếp cấp 2:  256\*256 = 65 MB  - Gián tiếp cấp 3:256\*256\*256 =16GB  **Hệ thống tập tin trên UNIX V7** |
| **Tổ chức hệ thống tập tin trên đĩa từ**  Master Boot Record (MBR): thường nằm tại sector logic 0, kích thước 512 bytes  **- Phân vùng (Partition):** Primary, Extended  Tối đa 4 phân vùng  **-Boot block + Super block (Boot sector)**  -Chứa các thông số quan trọng của phân vùng  -Chứa một đoạn chương trình nhỏ để nạp HĐH khi khởi động máy | **Master Boot Record**  -Đoạn chương trình để giúp khởi động hệ thống  -Bảng mô tả thông tin các phân vùng logic  +TYPE-ID=0x07 :Windows  +TYPE-ID = 0x83 : Linux  +TYPE-ID = 0x00 : Không sử dụng.  -Thông tin nhận diện MBR  3. Chuyển quyền điều khiển về cho đoạn mã chương trình nằm  trong Boot Sector của phân vùng được chọn 4. Tải HĐH tại phân vùng được chọn | **Quá trình khởi động hệ thống từ đĩa từ**  1. POST (Power-On Self-Test) 2. Tải MBR để đọc thông tin bảng phân vùng.  Tìm phân vùng “active”.  Nếu không tìm thấy phân vùng “active”, MBR có thể tải một boot loader và chuyển điều khiển cho nó. Boot loader này sẽ cho phép  chọn HĐH trên một phân vùng | FAT là hệ thống tập tin được sử dụng trên HĐH MS-DOS và Windows 9x (trên Windows họ NT có thêm hệ thống NTFS). Có 3 loại FAT  FAT12 FAT16 FAT32  - Tổ chức thành 2 vùng  + Vùng hệ thống  + Vùng Boot Sector  + Bảng FAT  + Bảng thư mục gốc (có thể nằm trên vùng dữ liệu) |
| **Xóa tập tin (DELETE**) Xác định entry chính trong bảng thư mục (RDET/ SDET)  chứa thông tin của tập tin dựa vào phần tên và phần mở rộng (lưu ý trường hợp tên dài) Đặt giá trị E5h vào byte đầu tiên của entry chính và tất cảcác entry phụ của tập tin (nếu có)Từ entry chính tìm được, ta có được chỉ số cluster/ phần tử  FAT đầu tiên. Vào bảng FAT, xác định được các phần tửcòn lại của tập tin. Đặt tất cả các phần tử FAT của tập tin về giá trị 0. Lưu ý, hoàn toàn không thay đổi gì phần nội dung của tập tin | **Xóa thư mục (RD**): Thực hiện xóa đệ qui tất cả các tập tin và thư mục con từ cấp sâu nhất ra. Xóa thư mục rỗng tương tự như xóa tậptin  **Di chuyển tập tin (MOVE)**  -Tìm đủ số entry trống liên tiếp nhau trên bảng thư mục (RDET/ SDET) để chứa thông tin của tập tin đích (lưu ý trường hợp tên dài)  - Copy thông tin (các entry) của tập tin nguồn sang  các entry tìm được của tập tin đích  - Xóa thông tin của tập tin nguồn | **Sao chép tập tin (COPY)**  Tìm đủ số entry trống liên tiếp nhau trên bảng thư mục  (RDET/ SDET) để chứa thông tin của tập tin đích (lưu ý trường hợp tên dài)  -Kiểm tra trên bảng FAT xem còn đủ số cluster trống để chứa nội dung của tập tin đích không-Copy thông tin (các entry) của tập tin nguồn sang các entry tìm được của tập tin đích-Ghi giá trị vào các phần tử FAT trống tìm được theo dạng danh sách liên kết, đồng thời copy các sector nội dung tậptin nguồn vào các sector nội dung tương ứng tìm được của  tập tin đích | **Đổi tên tập tin/ thư mục (REN)** Xác định entry chính trong bảng thư mục (RDET/ SDET) chứa thông tin của tập tin/ thư mục dựa vào phần tên và phần mở rộng (lưu ý trường hợp tên dài)  - Nếu tên tập tin không cần thêm các entry 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(RDET/ SDET) để chứa thông tin của tập tin đích (lưu ýtrường hợp tên dài)- Copy thông tin (các entry) của tập tin nguồn sang các entrytìm được của tập tin đích. |