- +Kernel: chức năng chính đọc, ghi dữ liệu bộ nhớ, tiến hành xử lý các lệnh, xác định cách dữ liệu được nhận và gửi thông qua các
- +API: giao diên lập trình ứng dung cho phép các lập trình viên phát triển ứng dụng có thể viết mã module
- +Giao diên 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 TRO: 1. Cung cấp giao diện ảo trừu tượng 2. Quản lý, cấp phát tài nguyên máy tính một cách công bằng, hiệu quả và an toàn 3. Tạo điều kiện thuận lợi và đơn giản hóa việc lập trình

+PHÂN LOAI:

-Batch OS(theo lô): Chương trình sẽ nằm trên các thẻ và được đưa vào máy đọc -> tuần tư. (+) tốc đô nhanh hơn, (-): CPU rành trong quá trình nhập/xuất.

Thiếu sư tương tác với user - Spooling - Cuôn: Simultaneous Peripheral Operation On-Line

Tốc độ CPU >> đầu đọc thẻ, máy in

Ô cứng ra đời >> đầu đọc thẻ

-Multiprogramming(đa chương): nạp đồng thời nhiều chương trình. Khi program hiện tại đang chờ I/O dev 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

-Timesharing OS(Multitasking OS - đa nhiệm): Thông qua time slice để trao quyền lần lươt sử dụng CPU cho các tiến trình với thời gian chuyển đổi diễn ra rất nhanh. Phức tạp hơn đa chương, có them chức năng: Quản trị và bảo vê bô nhớ, sử dụng hô nhớ ảo

(+): Tốc độ chuyển đổi nhanh nên tạo cảm giác mỗi người dùng có một máy ảo riêng. (-): CPU scheduling phức tạp => các hdh ngày nay sử dụng cơ chế này

-Parallel OS(Multiprocessing OS - đa xử lý): nhiều bô xử lý (CPUs), chia thành các cviec nhỏ hơn, xchia sẻ bus truyền dữ liêu, đồng hồ hô nhớ và các thiết hị ngoại vị

(+). Tặng sức mạnh và đô tin cây (Sự hỏng học của 1 hộ vừ lý sẽ không làm "sụp" đến toàn bộ hệ thống), (-): Yêu cầu cao về phần cứng & khó về lân trình

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

-Distributed OS: Turong tur timesharing nhưng các CPU sở hữu bộ nhớ, đồng hồ riêng. Hdh phối hợp nhiều máy tính khác nhau -> chia sẻ tài nguyên, ttin liên lac. (+): bền vững 1 máy hư thì vẫn

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

+KIÉN TRÚC HDH:

Đơn giản: Tất cả (ứng dụng, thư viên, nhân HĐH) chung không gian địa chỉ. (+) Xử lý nhanh, Dễ phát triển, mở rộng, Phù hợp với hệ đơn

người dùng (-) Ko có bảo vệ giữa nhân và ứng dụng, Dễ xảy ra xung đột khi

mở rộng Monolithic Kernel: Các dvụ hdh đều dược tích hợp vào kernel

(+) Hiệu năng cao, dễ hiệu với ltvien, có bvệ giữa ứng dụng và (-) Không có bảo vệ nội bộ giữa các thành phần trong nhân, khố

mở rông

Microkernel (Client-Server): nhân cực nhỏ, chi quản lý tối thiều (giao tiếp, bộ nhớ, CPU), giao tiếp qua message

(+) Dễ mở rộng, độc lập, dễ bảo trì, một dịch vụ lỗi không làm treo toàn 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 hdh - Các module của hệ thống luôn có mặt trong bộ nhớ trong
- Được đặt ở các vùng biên của bộ nhớ trong tại vùng biên dưới
- 2 loại: Nhân bắt buộc phải có ở bộ nhớ trong và Nhân khi cầi thiết mới goi vào
- Các module của nhân: loader, monitor (lưa chon các bước làm viêc) scheduler

BẢO VỀ VÀ BẢO MẬT

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

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

- Môt số cơ chế: Hoat đông ở 2 chế đô (kernel vs. user mode). Sao lutu (Backun). Xác thực người dùng (User Authentication). Phân quyền (Authoriz-ation), chính sách bảo mật (Policy), Kiểm soát nhât ký (Audit log).

OUÁ TRÌNH KHỞI ĐỘNG - 3 giai đoạn

1) CPU thực thi lệnh tại địa chi biết trước (boot ROM) x86: 0xFFFF0 trò tới BIOS

Firmware nap boot loader;

Fw là pmêm chi đoc.

- x86; ktra cấu hình CMOS (complementary metal oxide semiconductor); nap trình glý ngắt và các trình đkhiển thiết bị; khởi tao thanh ghi và qlý nguồn (power); qlý 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 nap HĐH.

Lúc này HĐH vẫn chưa chay

- Boot loader hiểu đc nhiều HĐH khác, nhiều phiên bản HĐH và hỗ trợ dual-boot.

QUẨN LÍ TẬP TIN

1. Khái niệm: đĩa phẳng bằng thủy tinh/kim loại có phủ lớp từ để uru data. Mỗi lần đọc/ghi ít nhất 1 sector.

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

Access Time = Seek time + Rotational time + Read time 1/2 * (Số Cynlinder * tg di chuyển qua mỗi track + tốc đô 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

*) Dĩa mềm: 1.44MB. Có 2 head, 80 track/head, 18 ector/track

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

(*) Quy đổi:

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

Sector logic→Sector vât lí: $s = (l \mod st) + 1$

> $t = l \operatorname{div} (st \cdot hd)$ $h = (l \operatorname{div} st) \operatorname{mod} hd$

sector logic; s: sector vlý; st: số sector / track; t: track; h: head; hd: số head (số side)

l = 36t + 18h + s - 1 (dĩa 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 ctrình khởi động hệ thống, có băng nhận

Phân vùng (partition) gồm primary và extend, có tối đa 4 phân

Boot block + Super block (Boot sector) chứa thông số quan trọng

của phân vùng, chứa đoạn ctrình nạp HĐH khi khởi động.

Quá trình khởi động từ đĩa từ) POST (Power-on self-test) 2) Tài MBR đọc thông tin bảng phân vùng. Tìm phân vùng

'active". Nếu k tìm thấy phân vùng "active". MBR có thể tại một boot loader và chuyển đkhiển cho nó. Boot loader này cho phép chon HĐH trên 1 phân vùng

 Chuyển quyền đkhiển về cho đoạn mã chương trình nằm trong Boot Sector của phân vùng được chọn.

Tài HĐH tại phân vùng được chọn.

THUẬT TOÁN ĐỘC ĐĨA

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

Ưu điểm: Để 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

Ưu điểm: Giảm di chuyển đầu đọc, tối ưu hơn FCFS. Nhược điểm: Có thể 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 rìa thì quay lại. Uu 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 rìa cuối (max track), đầu đọc quay về đầu (min track) mà không xử lý yêu cầu trên đường về. Ưu điểm: Cân bằng thời gian chờ yêu cầu

 LOOK: Giống SCAN nhưng không đi đến rìa đĩa, chi 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.

1. Tệp 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 tương này thao tác trên đó. Lưu trữ tân hơn 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. Gồm thuộc tính và nội dung. Mỗi hệ thống tập tin có cách tổ chức khác nhau

b. Loai tâp tin: - Ordinary file - Directory file: Thư mục chứa cá thư mục khác - Shortcut file - Special file (Device file): Thư mụ đại diên cho một thiết bị ngoại vị nào đó (USB, Máy in,...)

c. Truy xuất tâp tin: Tuần tự (đọc a→z, có thể quay luirewind). Ngẫu nhiên (có thể seek đến vị trí cần đọc). Indexed

Sequential (Linus/Inode): Mỗi block dữ liêu có 1 index.

d. Thư mục: là loại tệp tin đặc biệt. Nội dung của thư mục: một cấp (tất cả file có cùng thư mục), hai cấp (mỗi người dùng 1 thư muc), cây phân cấp (sử dụng hiện nay). Model of access: R, W,

e. Cấu trúc file: Do HĐH hoặc ctrình quyết định, có thể có hoặc không có cấu trúc.

- Dãy các bytes/words không có cấu trúc (Window, Linux) - Dãy các bản ghi fixed-length (Mainframes cũ) - Tree of records (Hệ thống lớn cho mục đích thương mại có nhu cầu gom nhóm các đối tương với nhau)

2. Thu muc

Loại tập tin đặc biệt, giúp tổ chức có hệ thống các tập tin. Thuộc tính của thư mục tương tự tập tin. Ndung của thư mục: qlý các tập tin, thu muc con của nó. Thao tác trên thư mục

Tạo, xóa, mở, đóng, liệt kê nội dung, tìm kiếm tập tin, duyệt hệ thống tập tin.

VÁN 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 dài: 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 mô tả cách thức cấp phát các block cho các tân tin, gồm:

- Cấp phát liên tục: Mỗi tập tin chiếm các block liên tục trên đĩa → Chi block bắt đầu và chiều dài. Hỗ trơ truy xuất tuần tư và trưc tiếp. Vấn đề external fragmentation và kích thước têp lớn

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

Quản lí các thông tin: block bắt đầu, số block, con trở trở 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 rác trên đĩa, chỉ hỗ trợ truy xuất

Chi cần vị trí bắt đầu, không bị external fragmentation, tốn chi phí lưu địa chi 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 liêu. Hỗ trơ truy xuất tuần tư và trực tiếp. Tốn không gian để lưu block chỉ mục, không bị external

fragmentation. Mở rộng: chỉ mục nhiều cấp, chỉ mục kết học linkedlist, Enhanced version: Indexed Allocation (I-nodes) dùng multilevel

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

- Ds liên kết: k tốn kgian đĩa chi nhí duyệt ds cao

Grouping: chứa ds block trống, dễ tìm lương lớn block trống Counting: chứa đia chỉ block trống đầu tiên và số lương 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 trò tới volume tương ứng).

System-wide open-file Table - ds các file đang được mở trên hệ thống. (ptr tập tin, định vi file trên đĩa; quyền truy cập; biến đếm file đang mớ). - Per-process open-file Table - ds các file mà tiến trình đang thao tác (ptr trò tới file đạng 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 HĐH thường phát hiện và tư động kết buộc các hệ thống tập tin tồn tại trên hệ thống (Win mount vào ổ đĩa, Linux mount vào folder

HÈ THỐNG TẬP TIN FAT HĐH MS-DOS và Windows 95

3 Ioai: FAT12 FAT16 và FAT32

2 vùng: vùng hệ thống (boot sector, fat1, fat2, root directory) và vùng data.

Master boot secto

 Nơi lưu trữ thông tin về các partition trong ổ đĩa 0x0: Đoạn mã khởi động

0x1B8: Dấu hiệu nhân diện đĩa 0x1BC: Không dùng, all 0

0x1BE: Bång phân vùng 0x510, 0x511: Dấu hiệu MBR

đông, 00 - Không khởi đông,

Mỗi 16 bits trong 0x1BE, 1 byte đầu (80 khởi động, 00 k kđộng), 3 byte kế thông số vlý (tsvl), 1 byte kế partition (06, 0B là fat; 07 là ntfs; 83 là linux; OF là extended), 3 byte kế tsvl; 4 byte kế partition; 4 byte cuối kích thước partition (bnhiu sector). 07 – NTFS, 83 – Linux, OF – Extended, 06, 0B – FAT, 80 – Khởi

Boot sector

Trạng thái

Gồm một số sector của partition. Sector đầu tiên chứa các thông số quan trọng của phân vùng và một đoạn ctrình nạp HĐH khi khởi động. Sector còn lại chứa các thông tin hỗ trợ cho xđịnh tổng số cluster trống & tìm kiếm cluster trống hiệu quả, chứa một sector bán sao của boot sector. Boot sector -> FAT1 FAT2 -> RDET -> DATA

Gtrị trên FAT32 Note

Irong		0	= FREE		
Hu		0FFFFFFF7	= BAD		
Cluster	cuối	0FFFFFFFF	= EOF		
Nội dun	g file	20FFFFFEF			
#	S	mean			
0	3	lênh nhày đến đầu đoạn	mã boot		
3	8	tên công ty / version của hđh			
В	2	số byte của sector (512)			
D	1	số sector của cluster S _C			
E	2	số sector của boot sector S _R			
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	kí hiệu loại 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 m tả vol đến đầu vol			
20	4	số sector của volume (if #13 > 65535)			
24	1	k/hiệu vlý của đĩa (0: mềm, 80h: cứng)			
25	1	dành riêng			
26	1	ký hiệu nhận diện hđh			
27	4	serialnumber của volume			
2B	b	volume label			
36	8	loại fat, là chuỗi "fat12" hoặc "fat16"			
3E	lcf	đoạn ctrình boot nạp tiếp hđh			

FAT16 quản lý tối đa 65518 (FFEEh) 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			
В	2 1	số byte của sector (512)			
D	1	số sector của cluster S_C			
E	2 1	số sector của boot sector S_B			
10	1	số lượng bảng fat N_F (2)			
11	2 2 1 2 2 2 4	Không dùng			
13	2	Không dùng			
15	1	kí hiệu loại volume			
16	2	Không dùng			
18	2	số sector của track			
1A	2	số lượng đầu đọc (side)			
1C		k/cách từ nơi m tả 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 lưu của BS			
34	4 2 2 4 2 2 C	dành riêng			
40	1	kí hiệu vật lí của đĩa chứa volume			
41	1	dành riêng			
42	1	kí hiệu nhận diện của HDH			
43	4	Serialnumber của vol			
47	В	Vol label			
52		Loại FAT, là chuỗi "FAT32"			
5A	1A4	Đoạn ctrinh khở tạo			
1FE	2	Dấu hiệu kết túc BS (AA55h)			
- 41					

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 số byte của sector (512)

Byte đầu mỗi entry: entry trống (0), tập tin ở chiếm entry đã xós (E5), thông tin file / folder (giá trị còn lại).

Entry chính chứa thông tin của tập tin s

 $S_{pner} =$

0	8	Tên chính		
8	3	Tên mở rộng		
В	1	Thuộc tính trạng thái		
C	1	Dành riêng		
D	3	Giờ tạo		
10	2	Ngày tạo		
12	2	Ngày truy cập gần nhất		
14	2	Cluster bắt đầu – phần word cao		
16	2	Giờ 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		
readonl	y, 1:hid	den, 2: system, 3: vollabel, 4: directory,		

Entry phu chi chứa tên tân tin

#	s	mean
0	1	Thứ tự của entry
1	A	5 kí tự Unicode
В	1	Dấu hiệu nhận biết (luôn là 0F)
E	C	6 kí tự tiếp t heo
1C	4	2 kí tự tiếp theo

Vùng dữ liệu

archive

Mỗi phần tử gọi là cluster, có kích thước $S_C = 2^n$ sector, tùy vào người dùng format. Đánh stt từ 2.

Cluster thứ k sẽ bắt đầu tại sector: $S_B + S_F \cdot N_F + S_{RDET} + (k-2)S_C$

Bảng thự mục con

SDET - Sub Directory Entry Table

Nằm trên vùng dữ liệu, có cấu trúc hoàn toàn giống bảng thư mụ

Mỗi SDET luôn có 2 entry . và .. ở đầu bảng mô tả về chính thư muc này và thư muc cha của nó.

Thao tác trên FAT

- Đọc nội dung (type): Xđịnh entry chính chứa t/tin thư mục dựa vào phần tên và phần mrông. Từ entry chính tìm được cluster bắt đầu. Dưa vào bảng fat tìm các phần tử còn lai. Đọc các sector nổi dung của tân tin.

- Liệt kê nội dung (dir): Xđịnh entry chính chứa t/tin 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 theo từng entry (32 bytes) và hiện thi ttin.

- Tao tâp tin (copy con): Tìm đủ số entry trống liên tiếp trên bảng thư mục. Ktra bảng fạt còn đủ cluster trống k. Lưu vào các entry trống tìm đc. 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 đc. Ghi giá trị vào các phần từ fat trống, tạo thư mục và .. chiếm 2 entry đầu tiên.

- Xóa tập tin (delete): Xđịnh entry chính trong bảng thư mục chứa t/tin tận tin dựa vào phần tên và phần mrông. Đặt giá trị E5h vào byte đầu tiên của entry chính và tất cả các entry phụ. Từ entry chính tìm được cluster bắt đầu. Dựa vào bảng fạt tìm các phần tử còn lai. Đặt tất cả các phần từ FAT của tập tin về giá tri 0.

Xóa thư mục (rd): Xóa để 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/tin entry của tập nguồn sang entry tập đích. Ghi giá tri vào các phần từ fạt trống, copy các sector nội dung tập nguồn vào các sector nôi dung tập đích.

- Di chuyển tập tin (move): Tìm đủ số entry trống liên tiếp trên bảng thư mục. Copy t/tin 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/tin tập tin dựa vào phần tên và phần mrông. Nếu tên tập tin không cần thêm các entry phụ, cập nhật lại phần tên và phần mrông. Nếu tên tập tin cần thêm các entry phụ, tìm đủ số entry trống liên tiếp nhau trên bảng thư mục để chứa t/tin đích, copy t/tin entry của tập nguồn sang entry tập đích.

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

- Full format: Các thông số của từng thành phần trên phân vùng sẽ được xác định lại. Để tạo ra những dạng thức mới phù hợp hơn cho phân vùng. Chức năng này đi nhiên cũng được dùng cho những phân vùng chưa được định dang.

Fragmentation của FAT

Các phần từ FAT của 1 tập tin không liên tiếp nhau, hoặc, các phần từ FAT của các tập tin không liên tiếp nhau. Dẫn đến truy xuất châm cần đefram

QUẨN LÝ TIỂN TRÌNH (THREAD)

Môt lớp phần mềm ở giữa phần cứng và các ctrình ứng

dụng/người dùng. Thread gom: Program counter, Stack và Registers (Code, Data, Heap, Files là tài nguyên được chia sẻ giữa các thread trong cùng

môt tiến trình (process)) + Single-threaded process (Tiến trình đơn tiểu trình): Chi có l luồng thực thi duy nhất

+ Multi-threaded process (Tiến trình đa tiểu trình): Chia thành nhiều luồng, mỗi luồng là 1 tiểu trình. Thực thị các công việc khác + User-level thread: Lâp trình viên điều phối, + Không biết tiểu trình, cho nên phải thực hiện hết các tiểu trình của tiến trình A thì

mới thực hiện qua các tiểu trình của tiến trình B, chứ không thể điều phối từ tiểu trình A1 qua B2 được -> <u>User thread:</u> code, data, heap, files (riêng: pc, stack, reg)

+ Kernel-level thread: Hê điều hành điều phối + Có bảng tiểu trình nên giao được CPU trực tiến cho các tiểu trình "Cần" các phần của A và B có thể thực hiện luôn phiên song song với nhau.

1. Phân biệt process & thread: tiến trình là chương trình đang được thực thi, nó cần sử dụng các tài nguyên máy tính. Một tiến trình có nhiều tiểu trình, các tiểu trình chia sẻ cùng 1 không gian bô nhớ

2. Tính năng

Trang thái:

+ New: process đang được tạo lập

+ Ready: process chờ được cấp phát CPU để xử lý

+ Running: các chỉ thị của process đang được xử lý

Blocked: process chờ được cấp phát tài nguyên hay chờ sự kiện xáv ra hoặc chờ nhập xuất + Waiting: process phải dừng vì thiếu tài nguyên hoặc chờ một sự

kiên nào đó + Halt: process đã kết thúc việc thực hiện, vẫn chưa bi xóa

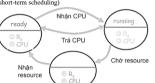
new -> ready: long term scheduling.

swap in-swap out: medium-term scheduling

Sẵn sàng -> chay: sau khi hđh điều đô CPU, process được HDH cấp phát CPU và chuyển sang trạng thái chạy

Chạy -> sẵn sàng: HĐH cấp phát CPU cho process khác, do kết quả điều đô/do ngắt xảy ra, process hiện thời chuyển sang trang thái sắn sàng và chờ được cấn CPU để chay tiến.

- Chay -> chờ đơi: Khi process đó chi chay khi có 1 sư kiện nào đó xảy ra, chuyển sang tt chờ được phân phối CPU để chạy tiếp. (short-term scheduling)



. Process address space (địa chỉ của tiến trình)

[Stack]: được cấp phát 1 cách tự động khi khái báo biến đó (Tham số hàm, return và các biến local khi một process được load

vào bô nhớ).

[Heap]: những biến được cấp phát đông. [Data]: hiến toàn cục hoặc hiến tĩnh

[Text]: file nội dụng ở dạng nhị nhân Process control block (PCB)

Trong bộ nhớ chính (phân vùng đặc biệt được quản lý bởi hệ điều hành). Bao gồm:

Process identifier: pid Process state: trang thái hiện tại

Program counter: đ/chi lênh kế tiếp được thực thi CPU register: thanh ghi lưu thông tin gắn với lệnh cần được thục

CPU scheduling information: lưu những thông tin cần thiết cho OS điều phối CPU

Memory management information: lưu thông tin địa chi của tiến trình trong bô nhớ Accounting information: lưu thông tin thống kê hiệu suất của hệ

I/O status information: luu thông tin về các thiết bị nhận xuất.

file,. 4. IPC liên lạc giữa các tiến trình

Signal: không đồng bộ, giao tiếp giữa kernel với user hoặc user với nhau, không trao đổi dữ liệu. | Pipe: có kích thước giới hạn, hỗ trợ liên lạc một chiều giữa 2 tiến trình cha con (original) hoặc liên lạc 2 chiều, dữ liệu có cấu trúc (named). | Shared memory: không gian được os tạo ra để chia sẽ dữ liệu | Socket: IP xác định

máy tính, port xác định tiến trình. ĐIỀU PHÓI TIẾN TRÌNH

1. Scheduling Procedure

2. Tiêu chí: (1) Công bằng và hiệu quả, (2) Maximize CPU ultilization, (3) Maximize throughput (số process/ đvị tgian), (4) Maximize turnaround time, waiting time, and respone time

- Turnaround time (TT): từ lúc tiến trình đi vào ready queue cho tới lúc hoàn tất tác vu → ra ngoài
- Waiting time (WT): khoảng tgian nằm trong ready queue chờ
- Respone time (RT): từ lúc tác vụ đi vào cho tới khi nó có phản hồi đầu tiên cho người dùng. 3. Mục tiêu điều phối:
- Hệ thống theo lô: tối ưu throughput, giảm thiều TT,..
- Hệ thống t.tác: giảm thiều WT, cân đối mong muốn user.
- Hệ thống tgian thực: tuân thủ deadline, có khả năng dư đoán,...

4. Các thuật toán

- (FCFS): hđh theo lô, độc quyền, (+) đơn giản, dễ cài đặt, (-) hiện tương tích lũy tgian chò, không phù hợp hđh t.tác user.
- Round Robin (RR): h

 ßh timesharing, không độc quyền, m

 ß tiến trình sử dụng một lượng q cho mỗi lần sử dụng, sử dụng queue, (+) công bằng, giảm WT đối với những tiến trình có CPU burst it, (-) how to set quantum time hiệu quả? phù hợp hđh t.tác user. $a \in [100; 1000] (ms)$.
- Shortest job first (SJF): mặc định độc quyền, chọn tiến trình có CPU Burst ngắn nhất để thực thi, giải quyết tranh chấp: chung thủy/ không chung thủy/ tích hợp thêm một số độ ưu tiên. (+) đơn giản, giảm TTT & WT đ/với những tiến trình có CPU burst it, cải thiên throughput (-) phải xđ đc CPU Burst của tất cả tiến trình, starvation. Phiên bản không độc quyền: Shortest-Remaining-Time-Next (SRTN). - Priority: hđh timesharing, không độc quyền hoặc độc quyền
- (+) mỗi tiến trình có int(priority). Vấn đề Starvation → Giải pháp Aging: tăng priority tiến trình chờ lâu.
- Multilevel Priority Queue Scheduling: chia ready queue thành nhiều queue, mỗi hàng có đô ưu tiên và thuật toán điều phối riêng (FCFS, RR, v.v).
- (*) Zombie: tiến trình con kết thúc nhưng cha chưa nhân được tín hiệu. (*) Mổ côi: tiến trình cha kết thúc nhưng tiến trình con

QUẨN LÝ TIỂU TRÌNH

Mỗi tiến trinh luôn có một tiều trình chính. Các tiểu trình dùng chung: code, data, các tài nguyên (files), dùng riêng: stack, local data, register.

1. Khối quản lý tiểu trình - Thread control block - TCB

- (*) Thường chứa: id tiểu trình (tid): Không gian lưu các thanh ghị Ptr tới vị trí xác định trong ngăn xếp. Trạng thái của tiểu trình. 2. So sánh tiến trình và tiểu trình
- 2. Address binding (Relocation): Kết buộc địa chỉ là quá trình - Các tác vụ điều hành tiểu trình (tạo, kết thúc, điều phối, chuyển mapping không gian địa chỉ logic với một không gian địa chỉ vật đồi,..) it tổn chi phí hơn so với tiền trình. lý. Các giai đoan có thể Binding.
- Liên lạc giữa tiểu trình qua chia sẻ bộ nhớ, không cần can thiệp kernel.

1. 3 tiêu chí của 1 giải pháp đồng bộ: (1) mutual exclusion (độc quyền truy xuất), (2) progess (không ngăn tiến trình khác vào vùng găng - CS), bounded waiting (giới han chờ)

2. Giải pháp: a. Busy Waiting

Software solutions (phần mềm)- giải pháp do lập trình viên làm - Lock variable: Sử dụng 1 biến làm khóa mỗi một tiến trình

- muốn vào CS thì phải chờ cho đến khi lock thỏa mãn → 2 tiến trình vẫn có thể vào cùng lúc do biến lock chưa được gán sau khi hết vòng, sử dụng biến cờ hiệu (lock cũng là CS→không đảm bác
- Strict Alternation: sử dụng 1 biến luân phiên, chỉ dành cho 2 process, 1 tiến trình không có nhu cầu vào vùng găng nhưng lai giữ quyền truy cập → ko đảm bảo [2]
- Peterson's solution: sử dụng phối hợp 2 phương pháp trên
- → hao phí CPU, đảo lôn đô ưu tiên của các tiến trình, đáp ứng 3 đk nhưng không thể mở rộng cho N tiến trình.

Hardware solutions (phần cứng) - cần có sự hỗ trợ từ hệ thống - Interrupt Disabling: tạm thời tắt ngắt để tránh bị gián. không hiệu quả trong multiprocessor (các CPU khác vẫn chay bình thường).

- TSL (Test-and-Set Lock): Lệnh phần cứng nguyên tử kiểm tra và thiết lập biến khóa (lock). Đảm bảo (1) bằng cách lặp "busy waiting" đến khi lấy được lock. Đễ dùng nhưng gây lãng phí CPU nếu lock bị giữ lâu.

b. Sleep & Wakeup: Semaphore: Dijkstra năm 1965

+ Semaphore: Biến đồng bô đặc biết dùng để kiểm soát truy cập tài nguyên dùng chung. Có hai thao tác: down - wait - P, up -

+ Monitor: CTDL đóng gói mức cao dùng để đồng bô, bao gồm: condition variable: dùng wait() và signal(), tự động nutual exclusion khi vào monitor. đảm bảo được độc quyền truy xuất trên các biến dùng chung

A: Mutual Exclution - tài nguyên chi được sử dụng độc quyền

B: Hold and wait – P1 giữ tài nguyên R1 và đợi tài nguyên R2

C: No preemption – Tài nguyên chỉ được giải phóng bởi người

Ignorance: sử dụng Ostrich Algorithm (pretend that deadlocks

do not cause any problem) vì chi phí xử lí deadlock cao hơn rất

A → không thực tế. B → yêu cầu tài nguyên cần thiết ngay từ

quyết). C → cho phép hệ thống lấy lại tài nguyên (để gây lỗi).

đầu hoặc giải phóng tài nguyên rồi đơi tài nguyên khác (khó giải

D → Khả thi. Cấp phát tài nguyên theo thứ tự ưu tiên.

Avoidance: ngăn ngừa bằng cách kiểm tra trước khi cấp tài

Với tài nguyên chi 1 instance: dùng Resource Allocation Graph

Với nhiều instance: Banker's algorithm: Safe sate → tồn tại một

chuỗi cấp phát an toàn đảm bảo mọi yêu cầu tài nguyên đều được

Detection & recovery: cho phép hệ thống xảy ra deadlock, sau

+Với tài nguyên chỉ 1 instance: dùng Wait-for Graph → kiểm tra

FVới nhiều instance: dùng matrix-based algorithm (giống

Recovery: Terminate tiên trình (một vài hoặc tất cả) để phá vỡ

deadlock, Rollback vè trang thái trước (nếu có checkpoint), Thu

CPU chi tương tác được Registers, Cache, Memory (RAM). Vậy

Hệ điều hành sẽ quản lý Memory như thế nào để cùng một lúc xử

Ô nhớ là một thành phần đơn vị của RAM chứa thông tin dữ liệu.

Địa chi là thông tin định vị ô nhớ. Tập hợp các địa chi sẽ tạo thành

Virtual (Logical) Address: là địa chỉ ảo được generated, quản lý,

thao tác bởi CPU và là địa chỉ mà không gian tiến trình, không

-Physical Address: là địa chi thực sự của Virtual Address được

1. Giai đoan Compile time: từ source program tạo thành object

2. Giai doan Load time: Linker (linkage editor) và Loader (Load

3. Memory Management Unit (MMU): MMU là một thiết bi

phần cứng tích hợp với CPU làm nhiệm vụ mapping địa chỉ logic

Allocation/Deallocation: cấp phát và hủy cấp phát bô nhó

. CÁP PHÁT LIÊN TỤC: Không gian địa chỉ logic của các tiến

trình sẽ nằm liên tục trong bô nhớ chin. Tiến trình không được

chia nhỏ, toàn bộ tiến trình phải nằm liên tục trong bộ nhớ chính.

Main memory sẽ được chia sẵn thành các partitions khác nhau:

+ Fixed partitions: kgian có kích thước cổ định được tạo trước

khi nap tiến trình. Kích thước của partition có thể sẽ không hoàn

oàn phù hợp, vừa vặn với nhu cầu của tiến trình. Mỗi tiến trình

có kích thước cố định đi vào sẽ đợi ở hàng đợi, tiến trình nào phù

+ Variable-partitions (Hole: khối ô nhớ đang trống) được hình

thành trong quá trình nap, xử lý và hoàn tất tiến trình.

Kích thước partition cấp cho tiến trình vừa đúng với kích thước

First-fit: tiến trình sẽ được nạp vào không gian trống đầu tiên có

thể chứa được nó (+) Nhanh. (-) Làm cho bộ nhớ có khả năng bị

hợp với partition nào thì sẽ sử dụng partitions đó.

. CHIẾN LƯỢC CÁP PHÁT BỘ NHỚ:

module kèm theo system library) (hiện tại ít sử dụng).

vào thực thị rồi đầy ra. (hệ điều hành hiện đại là chủ yếu)

nhiều so với việc để cho deadlock xảy ra, deadlock cũng rất

Prevention: không để A, B, C, D đồng thời xảy ra.

nguyên xem hệ thống có còn ở safe state hav không.

vnchronization problems...)

Bốn điều kiện để deadlock xảy ra

DEADLOCK

trong P2 đợi X.

Xir lí deadlock

hiếm khi xảy ra.

(RAG).

chu trình.

lý các tiến trình.

cấp phát mà không deadlock.

đó phát hiện và khôi phục.

Banker's nhưng xem vêu cầu hiện tại).

hồi tài nguyên để cấp lại theo cách khác...

MAIN MEMORY (BỘ NHỚ CHÍNH)

. Address space - Không gian địa chỉ:

gian người dùng làm việc trên đó.

nap vào và nằm trong RAM.

module (Hiện tại ít sử dụng).

Management Unit

mà tiến trình cần.

2. Address Binding (Reallocation)

3. Protection: bảo vệ vùng nhớ

4. Sharing: Chia sẻ vùng nhớ

→ Code rõ ràng hơn semaphore, dễ kiểm soát nhưng phụ thuộc vào nnlt. chí phí xử lý. (* Độc quyền truy xuất và Phối hợp xử lí, Classic

thể chứa được nó (+): Tiết kiệm không gian (-): Tốn nhiều chi phí thời gian xử lý

Worst-fit: tiến trình sẽ được nap vào không gian trống lớn nhất có thể chứa được nó. (+) Giảm thiểu khả năng phân mành. (-) Tốn

6. BẢO VỀ ĐỊA CHỈ VÀ BỊNDỊNG:

+ Relocation register (base register): là thanh ghi tái đinh vi lưu địa chi đầu tiên của tiến trình đó trong bô nhớ chính.

+ Limit register: lưu kích thước của tiến trình đó trong MM Cả 2 thanh ghi đều được quản lý bởi CPU. MMU sẽ căn cứ vào giá trị trong 2 thanh ghi để mapping (chuyển đổi địa chỉ). So sánh ogical address với limit register:

 Nếu lớn hơn báo lỗi. - Nếu nhỏ hơn thì sẽ địa chỉ trong bộ nhớ chính cần lấy (physical addr) = logical addr + base reg

Swapping

Nguyên tắc: Backing store/Fast disk là một phân vùng đặc biệt trong ổ đĩa để tam thời chứa những tiến trình (toàn bộ) đang chờ hệ thống xử lý thay cho bộ nhớ chính. Tiến trình có thể được luận phiên <-> giữa bộ nhớ chính và backing store.

Do đó Swapping có thể gia tăng số lượng tiến trình được xử lý tại một thời điểm. Tuy nhiên, Swapping không hỗ trợ nạp một phần của tiến trình nên không thể thiết kế một tiến trình có kích thước lớn hơn bộ nhớ vật lý.

8. Kiểm tra ô nhớ trống hay không

- + Cơ chế quản lý với bitmap. Bô nhớ sẽ được chia thành nhiều đơn vi, mỗi đơn vi sẽ được quản lý bằng 1 bit trong bitmap. (0 - trống). (1 - đã được sở hữu). (+) nhanh. (-) tốn bô nhớ, không hiệu quả
- + Cơ chế quản lý với danh sách liên kết
- Bộ nhớ sẽ được quản lý bởi 1 linkedlist với các thông tin Kev (P [Process]: occupied or H [Hole]: free) cho biết không gian đó có trống hay không.
- Start cell cho biết không gian đó bắt đầu từ cell nào.
- Size cho biết không gian đó có kích thước bao nhiều cell

Swap Time:

Total swap time = 2 x (SizeOfProcess / TransferRate)

sue: Fragmentation - Vấn đề phân mảnh

+ Phân mảnh ngoại vi - External Fragmentation: Tổng kích thước free của bộ nhớ đủ để chứa 1 tiến trình nhưng lại phân bố không liên tục nên không thể chứa được tiến trình đó. > Giải pháp: Compaction: Combine các không gian free thành môt không gian free lớn hơn.

+ Phân mảnh nội vi - Internal Fragmentation: Bô nhớ đã được chia sẵn thành các partition cổ định. Là phân mạnh do các partition cấp phát cho một tiến trình nhưng tiến trình đó không sử dung hết không gian và không thể cấp không gian đó cho tiến trình khác, Giải pháp: Best-fit Allocation

VIRTUAL MEMORY (BỘ NHỚ ĂO)

Không gian bộ nhớ của tiến trình (Virtual Memory) có thể có kích thước lớn hơn bô nhớ thất (Physical Memory) nên không thể nạp toàn bô vào bô nhớ thất.

Cơ chế: Sử dụng phân vùng đặc biệt ở ổ đĩa Backing Store để chứa một phần của tiến trình chưa được sử dụng đến tại thời điểm này. Khi nào CPU xử lý đến phần đó, nếu chưa nằm

sẵn trong bô nhớ thì lúc này Hệ điều hành sẽ nhân được thông điệp có lỗi xảy ra và nap phần CPU cần xử lý đó vào bô nhớ chính. (-) : Tốc đô truy xuất trên ổ đĩa thấp hơn so với bô nhớ chính. Có khả năng xảy ra hệ thống nap qua lại giữa bộ nhớ chính và backing store liên tuc.

[Q]: Tại sao chỉ cần nạp 1 phần thì process vẫn chạy được?

1. Dynamic linking: Không chép đoạn nội dung hàm thực thi vào 3: Giai đoạn Execution time: chương trình cần kích thước của mã nhị phân mà chi link tới đại diện (tên hàm) tới giai đoạn thực RAM lớn nên khi cần thì sẽ đầy một phần bộ nhớ chương trình thi (execution time) hàm nào được gọi tới thì sẽ link đông tới nôi dung của hàm đó.

- 2. Dynamic loading: Khi CPU thực thị tới đoạn mã nào thì mới load đoan đó vào bô nhớ. Không nhất thiết phải load toàn bô những hàm sẽ sử dụng vào bộ nhớ tại một thời điểm.
- (được quản lý bởi chương trình và CPU) với địa chỉ thực sự của nó trong bô nhớ chính. Các quá trình cần đến Memory 3. Overlays: Chi một vài phần của tiến trình đạng được thực th tại thời điểm -> là kỹ thuật của lập trình viên viết sao cho đc v
 - CÁP PHÁT KHÔNG LIÊN TỤC Kỹ thuật cấp phát áp dụng cho Virtual Memory được xếp vào

nhóm kỹ thuật cấp phát không liên tục. Cho phép tiến trình được chia thành nhiều phần và mỗi phần được loạd vào một phân vùng không liên tục nào đó của bộ nhớ chính. Gồm: Segmentation và

1. Segmentation - chiến lược phân đoạn

+ Nguyên lý: Đoạn là đơn vị thực thi nhỏ nhất. Mỗi tiến trình fuoc chia thành các đoạn (có thể có kích thước khác nhau).

Việc chia đoạn phụ thuộc vào ý nghĩa luận lý của từng đoạn và tiêu chí phân đoạn của từng lập trình viên khác nhau. Các đoan này có thể nap vào những vùng không liên tục ở bô nhớ

+ Hai thanh ghi hỗ trợ: Base: Là địa chỉ đầu tiên của đoạn đó trong bô nhớ chính.Limit: Là

kích thước của đoan.

+ Address Binding: Trong Segmentation, moi địa chi logic do CPU thao tác có 2 phần <s:segment number, d: offset> - Segment Table (lưu ở bô nhớ chính): bảng phân đoan thông tin

thanh ghi base và limit của tất cả phân đoạn của tiến trình đó. -<u>Chuyển đổi địa chỉ logic <s, d> thành địa chỉ vật lý</u>: Dùng s để

Best-fit: tiến trình sẽ được nạp vào không gian trống nhỏ nhất có xác định limit, base> trong bảng phân đoạn. Kiểm tra (d≥0 và $d \le limit -1$).

Nếu sai: Addressing Error *Nếu đúng: Địa chi vật lý = base + d

2. Paging - Chiến lược phân trang

Nguyên lý: cấp phát bộ nhớ với paging: Tiến trình được chia thành các trang (page) Bô nhớ cũng được chia thành các đơn vị khung trang (frame)

Mỗi một trang sẽ được nạp vào 1 khung trang còn trống. page size = frame size và bằng bao nhiều tùy thuộc vào phần hỗ trợ của phần cứng (hệ thống).

Địa chỉ Logical: p: page number (số hiệu trang). d: offset. Ngoài ra địa chỉ tương đối còn được biểu diễn bằng gtrị tuyệt $d\hat{o}i = p*(S\hat{o} \text{ offset/ 1 trang}) + d.$

Dia chỉ vật lý: f: frame number (số hiệu khung trang). d: offset (vị trí tương đối của địa chi trong khung trang f, đánh số từ 0). Page Table Entry Structure

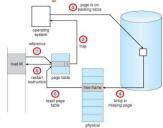
Frame No. Valid/ Invalid Protection Caching Referenced Modified

- Frame num: STT (= page p) Thông tin quan trong nhất, xác định vị trí vật lý của trang đó trong bộ nhớ chính.
- + Valid/Invalid: {0:1} 0 là chưa chứa thông tin trong bộ nhớ, c thể ở backing store; 1 là đã chứa thông tin trong bô nhớ
- + Protection: {0:1} 1: Read-only: 0: Read/Write
- + Caching {0;1} 0: Cho phép/1: Không cho phép caching (Luru dữ liệu vào cache)
- + Referenced {0;1} Cho biết trang đó trong chu kỳ đồng hồ trước đó có được truy cập hay không (Second Chance)
- + Modified (Dirty bit) {0;1} Check xem thông tin trang đã được chính sửa hay chưa

Demand Paging - Phân trang theo yêu cầu: Không nhất thiết toàn bộ các trang của tiến trình sẽ được nạp vào vùng nhớ. Chỉ nap những trang được yêu cầu. Vùng Backup Zone/Swapping Zone vùng đặc biệt nằm ngoài ổ đĩa tam thời chứa những phần của tiến trình

Quy trình xử lý lỗi trang

Dùng p (page number) vào bảng trang xác định bộ nhớ đã chứa dữ liệu trang hay chưa



Trong trường Valid/Invalid

Nếu 1, lấy giá trị f trong bảng trang vào khung trang f trong bộ nhớ chính để lấy dữ liêu

Nếu 0, báo về hệ điều hành "trap" và hệ điều hành ra backing store lấy trang đó nap vào khung trang trống trong bô nhớ vật lý. Reset lai bang trang và restart lai chi thi binding.

Translation Lookaside Buffers (TLBs) - Bô đêm tìm trang

Với số lượng trang nhiều, để tăng tốc độ tìm kiếm trên bảng trang, sử dụng High-speed cache (TLBs) là một cache được xử lý nhanh chỉ chứa một vài trang thường được sử dụng. Mục đích là giảm thời gian truy cập hiệu dung EAT. Khi trang CPU cần truy cận đã nằm sẵn trên TLBs (TLB hit) thì không cần truy câp bảng trang. Ngược lại, phải ra binding ở bảng trang

Thời gian truy cập <mark>hiệu dụng (EAT):</mark> EAT = (1-p)*tm + p*tp

> EAT with TLBs and Page Fault?

(p: xác xuất xảy ra lỗi trang; tm: thời gian truy cập bô nhớ; tp: thời gian xử lý lỗi trang)

> EAT with TLRs?

EAT = h*(tc + tm) + (1-h)*[tc + (level of Paging + 1)*tm](h: xác suất tìm ra trong TLB TLB hit ratio: tm: thời gian truy cân bộ nhớc tọ: thời gian tra trong TLB)

EAT = h * (tc + tm) + (1 - h) * [(1 - p) * (tc + (level of Paging+1)* tm) + p*tp]

(p: xác xuất xảy ra lỗi trang; h: xác suất tìm ra trong TLB, TLB hit ratio; tm: thời gian truy cập bô nhớ; tc: thời gian tra trong TLB: tp: thời gian xử lý lỗi trang)

Page Replacement Algorithms (Chiến lược thay thế trang). FIFO: chọn trang vào đầu tiên trong các trang nằm trong khung trang ở bô nhớ

(+): Đơn giản, dễ cài đặt

(-) : Không tối ưu, không hiệu quả do xảy ra quá nhiều Page Fault, Nghịch lý Belady

Optimal - Tối ưu hóa: Nhìn về tương lại xem trang nào ở xa nhất và chọn trang đó

(+): Hiệu quả tốt nhất, Ít Page Fault

cơ hội thứ 2 (bằng Ref Bit)

Hierarchical (Multilevel) Paging

thước rất lớn?

entry có 2/12 offset

không được dùng trong thời gian dài

(+); Giảm Page Fault đáng kể, khả thị,

(+): khả thi. (-): thêm thông số bổ sung RefBit

(-): Chi có trên lý thuyết, không biết được tương lai. Least Recently Used (LRU) Nhìn về quá khứ xem trang nào

(-): Cần sư hỗ trợ của phần cứng (cần thêm stack, counter)

Second Chance: phiên bản cải tiến của FIFO, cho mỗi trang 1

Large Address Space: Nap bảng trang có kích thước rất lớn vào

bộ nhớ? Tìm kiếm 1 trang tồn tại trong 1 bảng trang có kích

Two-level paging in 32-bit system: [10 bits Outer Page Table

tri P1, mỗi P1 xác định 1 bảng trang P2 có 1024 entries. Mỗi

P1][10 bits Page Table P2][12 bits offset d]: Xác định 1024 giá

Hashed Page Table: Tăng tốc độ, giảm thời gian truy xuất trên

bảng trang. PageNum p của đia chỉ logic sẽ được đưa vào hàm

linkedlist có giá tri hash đó và và node p thì sẽ lấy được frame

trang. Mỗi entry sẽ chứa thông tin định danh tiến trình kèm số

→ Tìm kiếm không hiệu quả nhưng giảm không gian lưu trữ

quá lớn dành cho bảng trang trong trường hợp nhiều trang không

Block device (thiết bị khối): Là thiết bị vận chuyển, lưu và tru

- Interrupt (gián đoạn): Là những tín hiệu được phát ra từ phầi

cứng hoặc phần mềm để thông báo một sư kiến đến hệ thống. Khi

hệ thống nhân được một tín hiệu interrupt, CPU sẽ ngưng các tiến

+ Hardware interrupt: được truyền tín hiệu bởi các thiết bị bên

+ Software interrupt: được phát ra từ chương trình người dùng

Ví du khi người dùng gọi hàm printf trong C thì khi hệ thống giao

quyền từ user mode sang kernel mode thì sẽ phát xuất dữ liệu theo

khối có kích thước cố định và những khối này thường được đánh

- Character device: dữ liêu nào nhân trước thì xử lí trước v

- Clocks (Timers): Bộ định thời. Xử lý những tín hiệu lquan

đến thời gian. Vai trò: cực kì quan trong Trong Timesharing OS

1. Crystal (tinh thể): dao động theo chu kỳ (count--) đến khi = 0

1) User level libraries: Là nơi phát sinh yêu cầu nhập xuất o Giao

diện đơn giản, nói chung, được hỗ trợ bởi ngôn ngữ lập trình

+ Xử lí các tầng trừu tương bên trên device driver, không phu

Fao giao diện chung cho các Device driver (system call)

+ Thiết lập thanh ghi thiết bị, kiểm tra trạng thái thiết bi

+ Chương trình đặc biệt của thiết bị để giao tiếp với device

controller → lớp trung gian giữa các mô-đun nhân hê điều hành

+ Driver phổ biển được tích hợp vào như một thành phần của hệ

điều hành (ví du: chuôt, bàn phím, ... không cần cài thêm driver

+ Interrupt Service Routine (ISR) được gọi khi interrupt xuất hiện

1) Thành phần thiết bị cơ học (mechanical controller): là các

2) Thành phần mạch điện tử (còn gọi là bộ điều khiển thiết bi

Được tích hợp vào các thiết bị I/O (đối với các thiết bị I/O

chuyên biệt) hoặc gắn trực tiếp vào bus của máy tính (đối với các

+ CPU chỉ giao tiếp trực tiếp với device controller mà không giao

F Tạo ra tín hiệu cho biết quá trình nhập xuất hoàn thành

thiết bị nhận xuất. Ví dụ: chuột bàn nhím, màn hình

Xử lí lỗi o Phân bổ và giải phóng tài nguyên

3) Device drivers (phu thuộc vào thiết bi)

(Device-Independent Software) và phần cứng

Được cung cấp bởi nhà sản xuất thiết bị

(API), để triển khai chức nặng nhận xuất o Các thư viên C

ngoài để thông báo chúng đã hoàn thành nhiệm vụ.

địa chị. Ví dụ: ổ đĩa lưu dữ liệu theo dạng khối

vì nó sẽ giúp đồng bô và điều phối hệ thống.

- Program interval clock: gồm 3 phần

3. Holding register to load counter

2) Device-Independent Software

thuộc vào thiết bị nhập xuất cụ thể

Các tầng phần mềm hệ thống I/O

Ví dụ: thiết bị nhập xuất báo hiệu trạng thái I/O kết thúc)

Inverted Page Table: Bang trang sẽ phân hoạch theo khung

Hash ra giá trị hash và đem qua tra vào Hash table và tra

hiệu trang p. Entry thứ i sẽ nằm ở khung trang thứ i.

nap vào bô nhớ chính tai thời điểm đó.

I/O HỆ THỐNG NHẬP XUẤT

trình hiện tại lại để xử lí interrunt

IO Management

Phân loai interrupt:

không lưu lại để xử li

thì -> clock interrupt reset

2 Counter

printf(), scanf(). ...

⊢ Ouán lý hô đêm

đặc biệt)

4) Interrupt Handlers

Cấu trúc thiết bị nhập xuấ

- device controller hav adapter)

thiết bị I/O theo quy chuẩn chung)

tiếp với mechanical controller

(lân trình viên).

- (command-ready, busy, error)
 - thi lênh của CPU

Ouv trình hoạt động

CPU kiểm tra status từ status register xem thiết bị sẵn sàng chưa o Sau khi kiểm tra thiết bị sẵn sàng thì CPU truyền lệnh cho command register để device controller thực thi o Khi nhân được lênh thiết bị sẽ thực hiện trao đổi thông tin và dùng đến local

Kĩ thuật I/O

1) Programmed I/O (Polling)

Cần hỗ trợ bởi Interrupt controller được tích hợp trên bo mạch để xử lí các tín hiệu interrupt truyền đi từ thiết bị nhập xuất

3) Direct memory access (DMA)

- DMA là kĩ thuật cho phép tương tác trực tiếp với bộ nhớ chính của máy tính không cần thông qua CPU
- Yêu cầu thiết bị DMA controller (thường được tích hợp trong bo mạch chính). Thiết bị này có thể tương tác trực tiếp với bộ nhớ và thiết bị nhập xuất không cần CPU

- > NTFS (New Technology File System): hệ thống tập tin cục
- VNIX File System (UFS): cũng là hê thống tập tin cục bô,
- ZFS: hê thống tâp tin nâng cao, hỗ trợ quản lý khối lượng lớn dữ liệu, nhưng không phải phân tán.
- > NFS (Network File System) là hệ thống tập tin phân tán (distributed file system)

Monolothic -> + Các dịch vụ hdh đều được tích hợp vào kernel

Microkernel: truvên message

Journaling file system là một loại hệ thống tập tin sử dụng log (nhật ký) để ghi lại các thay đổi trước khi thực sự áp dụng lên hệ

Bất thường Belady là tang frame nhưng page fault kh giảm ví dụ

monolithic: + Các dịch vụ hdh đều dược tích hợp vào kernel + shared memory => context switch nhanh, nhưng bị sập 1 cục. microkernel: ngược lại vs monolithic => chậm nhưng an toàn,

- bộ dùng trong Windows.
- dùng trên các hệ điều hành UNIX.

+ shared memory

thống tập tin. Mục tiêu là đề: Tăng tính toàn vẹn dữ liệu., Cho phép khôi phục hệ thống nhanh chóng sau khi xảy ra sư cố như mất điện, crash,... Tránh để hệ thống rơi vào trang thái không nhất quán.

trong thuật toán FIFO.

Cấu trúc device controller bao gồm

- + Có data buffer để lưu trữ các dữ liệu tạm thời (ví dụ: ổ đĩa đọc từng sector cần lưu trữ tạm thời)
- + Các thanh status register để lưu giữ trang thái của thiết bi
- + Các thanh command (hoặc control) register để lưu giữ các chi

data buffer khi cần lưu trữ các dữ liệu tạm thời

2) Interrupt driven I/O