PHƯƠNG PHÁP GIẢI BÀI TẬP

HỆ ĐIỀU HÀNH

BY SITDE PK

MỤC LỤC

CHIẾN LƯỢC ĐIỀU PHỐI FIRST-IN-FIRST-OUT/FIRST-COME-FIRST-SERVE (FIFO/FCFS) 4
CHIẾN LƯỢC ĐIỀU PHỐI XOAY VÒNG (ROUND ROBIN – RR) 5
CHIẾN LƯỢC ĐIỀU PHỐI SHORTEST JOB FIRST (SJF) 6
CHIẾN LƯỢC ĐIỀU PHỐI SHORTEST REMAINING TIME FIRST (SRTF) 7
CHIẾN LƯỢC ĐIỀU PHỐI ĐỘ ƯU TIÊN PRIORITY (P) 7
A - ĐIỀU PHỐI ƯU TIÊN ĐỘC QUYỀN: 8
B – ĐIỀU PHỐI ƯU TIÊN KHÔNG ĐỘC QUYỀN: 8
CHIẾN LƯỢC ĐIỀU PHỐI XOAY VÒNG CÓ ĐỘ ƯU TIÊN (PRIORITY ROUND ROBIN) 9
THUẬT TOÁN CẤP PHÁT BỘ NHỚ FIRST FIT 10
THUẬT TOÁN CẤP PHÁT BỘ NHỚ BEST FIT 11
THUẬT TOÁN CẤP PHÁT BỘ NHỚ WORST FIT 11
ĐÁNH GIÁ ĐỘ HIỆU QUẢ CỦA CÁC THUẬT TOÁN CẤP PHÁT BỘ NHỚ 12
THUẬT TOÁN THAY TRANG FIRST IN FIRST OUT (FIFO) 14
THUẬT TOÁN THAY TRANG TỐI ƯU (OPT) 15
THUẬT TOÁN THAY TRANG LEAST RECENTLY USED (LRU) 16
KỸ THUẬT QUÉT THEO HÀNG ĐỢI FIRST COME FIRST SERVE (FCFS) 17
KỸ THUẬT QUÉT SHORTEST SEEK TIME FIRST (SSTF) 18
KỸ THUẬT QUÉT THANG MÁY (SCAN) 19
KỸ THUẬT QUÉT CIRCULAR SCAN (C-SCAN) 19
KỸ THUẬT QUÉT LOOK 20
KỸ THUẬT QUÉT CIRCULAR LOOK (C-LOOK) 21
TÍNH MA TRẬN NEED 24
TÍNH TỔNG SỐ THỂ HIỆN CỦA CÁC TÀI NGUYÊN 25
KIỂM TRA HỆ THỐNG CÓ Ở TRẠNG THÁI AN TOÀN HAY KHÔNG 26
CHIẾN LƯỢC PHÂN BỐ LIÊN TỤC (CONTIGUOUS) 29
CHIẾN LƯỢC PHÂN BỐ LIÊN KẾT (LINKED) 29
CHIẾN LƯỢC PHÂN BỐ LẬP CHỈ MỤC (INDEX) 30
GIẢI THÍCH CHẾ ĐỘ PHÂN QUYỀN 31
SỬ DỤNG LỆNH CHMOD ĐỂ PHÂN QUYỀN 31
GIẢI THÍCH CÁC QUYỀN KHÓ HIỂU (???) 33
A – QUYỀN ĐỌC* VÀ GHI*: 33
B – QUYỀN SỞ HỮU (CỦA FILE): 34
C – QUYỀN ĐIỀU KHIỂN: 34
D – QUYỀN CHUYỂN: 35
XÂY DỰNG, SỬ DỤNG MA TRẬN QUYỀN TRUY CẬP ĐỂ TRẢ LỜI CÁC CÂU HỎI VỀ PHÂN QUYỀN 35
PHÂN MẢNH NỘI ĐỐI VỚI BÀI TOÁN CHO KÍCH THƯỚC TRANG 37
PHÂN MẢNH NỘI ĐỐI VỚI BÀI TOÁN CHO SỐ LƯỢNG TIẾN TRÌNH 37

DẠNG BÀI SỐ 01
GIẢI THUẬT ĐIỀU PHỐI TIẾN TRÌNH

KIẾN THỨC CƠ SỞ

- ***Độc quyền***: Cho phép một tiến trình khi nhận được CPU sẽ có quyền độc chiếm CPU đến khi hoàn tất xử lý hoặc tự nguyện giải phóng CPU
- ***Không độc quyền:*** Cho phép tạm dừng hoạt động của một tiến trình đang sẵn sàng xử lý. Khi một tiến trình nhận được CPU, nó vẫn được sử dụng CPU đến khi hoàn tất hoặc tự nguyện giải phóng CPU, nhưng một tiến trình khác có độ ưu tiên có thể dành quyền sử dụng CPU của tiến trình ban đầu.

Đề bài

Cho một dãy các tiến trình P_i. Biết rằng tiến trình thứ i có thời điểm vào RL là t_i và thời gian xử lý là p_i. Lập bảng điều phối tiến trình/Vẽ biểu đồ Gantt bằng chiến lược điều phối ... ?

Dãy tiến trình tổng quát

Tiến trình	Thời điểm vào RL	Thời gian CPU
P1	t_1	p_1
P2	t_2	p_2
...	...	...
Pi	t_i	p_i

KIẾN THỨC CƠ SỞ

- ***Thời gian chờ (WT):*** $\text{WT} = \text{Thời điểm thực hiện} - \text{Thời điểm vào RL}$
- ***Thời gian lưu hệ thống (TAT):*** $\text{TAT} = \text{WT} + \text{Thời gian CPU}$

CHIẾN LƯỢC ĐIỀU PHỐI FIRST-IN-FIRST-OUT/FIRST-COME-FIRST-SERVE (FIFO/FCFS)

Nguyên tắc: Thực hiện xử lý theo thứ tự vào hệ thống

Điều phối theo **nguyên tắc độc quyền**

Ví dụ trực quan

Xét tập các tiến trình sau:

Tiến trình	Thời gian vào RL	Thời gian CPU
P1	0	9
P2	1	3
P3	2	4
P4	2	8
P5	4	1

Cho biết kết quả điều phối hoạt động của các tiến trình trên theo thuật toán **FIFO**?

Bài giải

Thời điểm	Tiến trình trong hệ thống	Tiến trình hoàn thành	Tiến trình tiếp	WT	TAT
0	{P1}	-	P1	-	-
9	{P2, P3, P4, P5}	P1	P2	0	9
12	{P3, P4, P5}	P2	P3	8	11
16	{P4, P5}	P3	P4	10	14
24	{P5}	P4	P5	14	22
25	Ø	P5	-	20	21

Giải thích bài làm

- Tại thời điểm 0, tiến trình P1 đi vào hệ thống và trở thành tiến trình được xử lý tiếp theo.

- Tại thời điểm 9, trong hệ thống có các tiến trình P2, P3, P4, P5 (đã đi vào hệ thống lần lượt ở các thời điểm 1, 2, 2, 4). Tiến trình P1 đã xử lý xong (hoàn thành), thời gian chờ của tiến trình P1 là 0 (do vừa vào hệ thống đã được xử lý ngay), thời gian lưu trong hệ thống là 9. Tiến trình xử lý tiếp theo là P2.

- Tại thời điểm $9 + 3 = 12$ (3 là thời gian CPU của P2), trong hệ thống còn các tiến trình P3, P4, P5. Tiến trình P2 đã hoàn thành, thời gian chờ của tiến trình P2 bằng $9 - 1 = 8$ (9 là thời điểm P2 được xử lý, 1 là thời gian vào RL của P2); thời gian lưu trong hệ thống là $8 + 3 = 11$ (8 là thời gian chờ của tiến trình P2, 3 là thời gian CPU của P2). Tiến trình xử lý tiếp theo là P3.

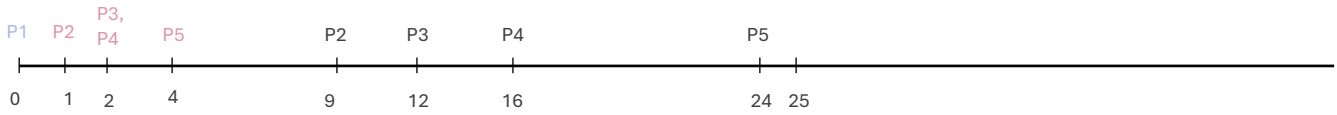
- Tại thời điểm $12 + 4 = 16$ (4 là thời gian CPU của P3), trong hệ thống còn các tiến trình P4, P5. Tiến trình P3 đã hoàn thành, thời gian chờ của tiến trình P3 bằng $12 - 2 = 10$ (12 là thời

điểm P3 được xử lý, 2 là thời gian vào RL của P3); thời gian lưu trong hệ thống là $10 + 4 = 14$ (10 là thời gian chờ của tiến trình P3, 4 là thời gian CPU của P3). Tiến trình xử lý tiếp theo là P4.

- Tương tự như vậy với các tiến trình còn lại.

Vẽ biểu đồ Gantt

Từ bảng điều phối tiến trình trên, ta dễ dàng vẽ được biểu đồ Gantt như sau:



(Màu hồng là thời điểm vào RL, màu đen là bắt đầu xử lý. Riêng P1 vừa vào đã được xử lý ngay)

CHIẾN LƯỢC ĐIỀU PHỐI XOAY VÒNG (ROUND ROBIN – RR)

Nguyên tắc: Bộ điều phối lần lượt cấp phát cho từng tiến trình trong danh sách một khoảng thời gian sử dụng CPU gọi là **quantum**. Điều phối này **không độc quyền**.

Nếu tiến trình hoàn thành trước khi hết thời gian **quantum**; hoặc hết thời gian **quantum** thì hệ điều hành cấp phát CPU cho tiến trình khác và đưa tiến trình về cuối *ready list* nếu vẫn chưa xử lý xong.

Ví dụ trực quan

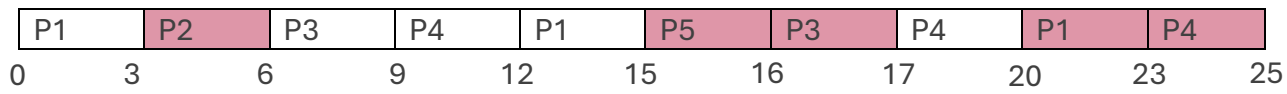
Xét tập các tiến trình sau:

Tiến trình	Thời gian vào RL	Thời gian CPU
P1	0	9
P2	1	3
P3	2	4
P4	2	8
P5	4	1

Cho biết kết quả điều phối hoạt động của các tiến trình trên theo thuật toán **ROUND ROBIN (RR)** với **quantum** bằng **3**?

Bài giải

Đối với giải thuật Round Robin, để tránh bị rối và nhầm lẫn trong các phép toán, ta nên vẽ biểu đồ Gantt:



(Ô màu hồng đánh dấu tiến trình đã hoàn thành tại đó)

Giải thích bài làm

Ở đây sẽ giải thích với P1, các tiến trình khác tương tự

- Tại thời điểm 0, P1 vào hệ thống và được xử lý. Do quantum bằng 3, đến thời điểm 3 sẽ bị tạm dừng để nhường chỗ cho tiến trình khác. Lúc này, P1 vẫn chưa được thực hiện xong và còn cần thêm 9 – 3 = 6 thời gian CPU nữa.

- P1 được tiếp tục tại thời điểm 12 sau khi hệ thống đã quay vòng hết các tiến trình khả dụng. Đến thời điểm 15, P1 tạm dừng nhường chỗ cho tiến trình khác. Như vậy, P1 còn cần thêm 6 – 3 = 3 thời gian CPU nữa.

- P1 được tiếp tục tại thời điểm 23. Đến thời điểm 26 thì hoàn thành.

CHIẾN LƯỢC ĐIỀU PHỐI SHORTEST JOB FIRST (SJF)

Nguyên lý: Tiến trình có thời gian CPU **ngắn nhất** được **thực hiện trước**. Chiến lược điều phối này **độc quyền**.

Ví dụ trực quan

Xét tập các tiến trình sau:

Tiến trình	Thời gian vào RL	Thời gian CPU
P1	0	9
P2	1	3
P3	2	4
P4	2	8
P5	4	1

Cho biết kết quả điều phối hoạt động của các tiến trình trên theo thuật toán **SJF**?

Bài giải

Dễ thấy thứ tự thực hiện sẽ là P1, P5, P2, P3, P4. Ta có bảng sau:

Thời điểm	Tiến trình trong hệ thống	Tiến trình hoàn thành	Tiến trình tiếp	WT	TAT
0	{P1}	-	P1	-	-
9	{P2, P3, P4, P5}	P1	P5	0	9
10	{P2, P3, P4}	P5	P2	5	6
13	{P3, P4}	P2	P3	9	12
17	{P4}	P3	P4	11	15
25	{}	P4	-	15	23

CHIẾN LƯỢC ĐIỀU PHỐI SHORTEST REMAINING TIME FIRST (SRTF)

Nguyên lý: Tiến trình có thời gian CPU **còn lại** ngắn thì thực hiện trước.

Ví dụ trực quan

Xét tập các tiến trình sau:

Tiến trình	Thời gian vào RL	Thời gian CPU
P1	0	9
P2	1	3
P3	2	4
P4	2	8
P5	4	1

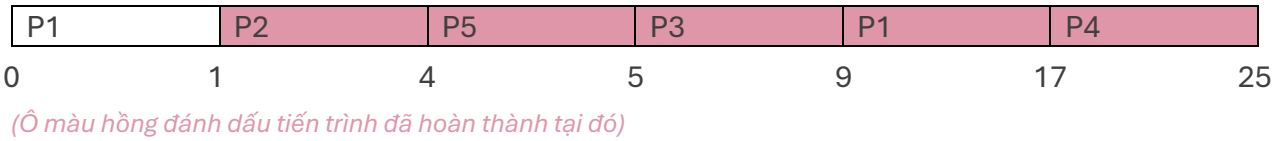
Cho biết kết quả điều phối hoạt động của các tiến trình trên theo thuật toán **SRTF**?

Bài giải

Ta có bảng điều phối tiến trình như sau:

Thời điểm	Tiến trình trong hệ thống	Tiến trình hoàn thành	Tiến trình tiếp	WT	TAT
0	{P1}	-	P1	-	-
1	{P1, P2}	-	P2	-	-
4	{P1, P3, P4, P5}	P2	P5	0	3
5	{P1, P3, P4}	P5	P3	0	1
9	{P1, P4}	P3	P1	3	7
17	{P4}	P1	P4	8	17
25	{}	P4	-	15	23

Tương ứng với biểu đồ Gantt:



(Ô màu hồng đánh dấu tiến trình đã hoàn thành tại đó)

CHIẾN LƯỢC ĐIỀU PHỐI ĐỘ ƯU TIÊN PRIORITY (P)

Nguyên lý: Tiến trình nào có **độ ưu tiên cao** *(Giá trị ưu tiên là số nhỏ hơn)* thì thực hiện trước.

Ví dụ trực quan

Xét tập các tiến trình sau:

Tiến trình	Thời gian vào RL	Thời gian CPU	Độ ưu tiên
P1	0	9	3
P2	1	3	2
P3	2	4	1
P4	2	8	3
P5	4	1	4

Cho biết kết quả điều phối hoạt động của các tiến trình trên theo thuật toán **Priority độc quyền/không độc quyền**?

A – ĐIỀU PHỐI ƯU TIÊN ĐỘC QUYỀN:

Bài giải

Thời điểm	Tiến trình trong hệ thống	Tiến trình hoàn thành	Tiến trình tiếp	WT	TAT
0	{P1}	-	P1	-	-
9	{P2, P3, P4, P5}	P1	P3	0	9
13	{P2, P4, P5}	P3	P2	7	11
16	{P4, P5}	P2	P4	12	15
24	{P5}	P4	P5	14	22
25	{}	P5	-	20	21

Tương ứng biểu đồ Gantt:

0 9 13 16 24 25

(Ô màu hồng đánh dấu tiến trình đã hoàn thành tại đó)

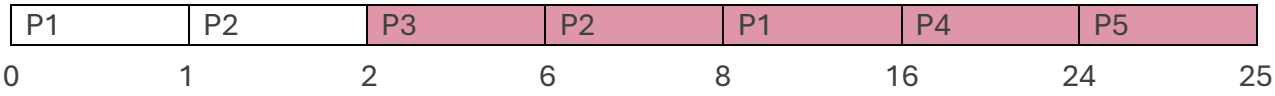
B – ĐIỀU PHỐI ƯU TIÊN KHÔNG ĐỘC QUYỀN:

Bài giải

Thời điểm	Tiến trình trong hệ thống	Tiến trình hoàn thành	Tiến trình tiếp	WT	TAT
0	{P1}	-	P1	-	-
1	{P1, P2}	-	P2	-	-

2	{P1, P2, P3, P4}	-	P3	-	-
6	{P1, P3, P4, P5}	P3	P2	0	4
8	{P1, P4, P5}	P2	P1	4	7
16	{P4, P5}	P1	P4	7	16
24	{P5}	P4	P5	14	22
25	{}	P5	-	20	21

Tương ứng với biểu đồ Gantt:



CHIẾN LƯỢC ĐIỀU PHỐI XOAY VÒNG CÓ ĐỘ ƯU TIÊN (PRIORITY ROUND ROBIN)

Nguyên tắc: Tiến trình nào có **độ ưu tiên cao** *(Giá trị ưu tiên là số nhỏ hơn)* thì thực hiện trước. Các tiến trình có cùng độ ưu tiên thì thực hiện xoay vòng (Round Robin). Giả sử tất cả các tiến trình cùng vào hệ thống tại thời điểm 0.

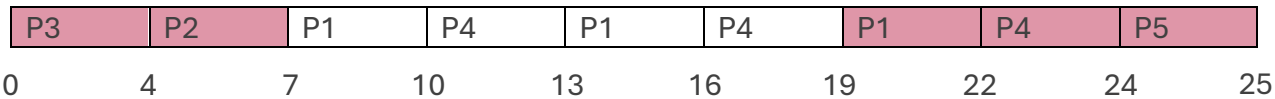
Ví dụ trực quan

Xét tập các tiến trình sau:

Tiến trình	Thời gian CPU	Độ ưu tiên
P1	9	3
P2	3	2
P3	4	1
P4	8	3
P5	1	4

Cho biết kết quả điều phối hoạt động của các tiến trình trên theo thuật toán **Round Robin (RR) có độ ưu tiên** với **quantum** bằng **3**?

Bài giải



DẠNG BÀI SỐ 02
PHÂN VÙNG/CẤP PHÁT BỘ NHỚ

Đề bài

Giả sử bộ nhớ chính được phân chia thành các vùng có kích thước $A_1, A_2, \dots, A_n$ (theo thứ tự). Cho biết các tiến trình có kích thước $P_1, P_2, \dots, P_m$ (theo thứ tự) sẽ được cấp phát bộ nhớ như thế nào?

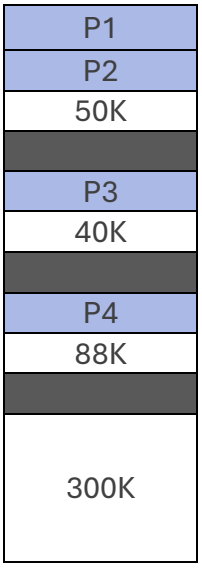
THUẬT TOÁN CẤP PHÁT BỘ NHỚ FIRST FIT

Nguyên tắc: Lần lượt cho các tiến trình vào các vùng bộ nhớ theo thứ tự. Nếu tiến trình không vừa vùng bộ nhớ đó thì đẩy sang vùng nhớ tiếp theo.

Ví dụ trực quan

Giả sử bộ nhớ chính được phân chia thành các vùng có kích thước 600K, 100K, 200K, 300K (theo thứ tự). Cho biết các tiến trình có kích thước 250K, 300K, 60K, 112K, 426K (theo thứ tự) sẽ được cấp phát bộ nhớ như thế nào nếu dùng thuật toán **First fit**?

Bài giải

Bộ nhớ khi chưa cấp phát	**Cấp phát theo thuật toán First fit**
	 Không còn đủ bộ nhớ để cấp phát cho P5

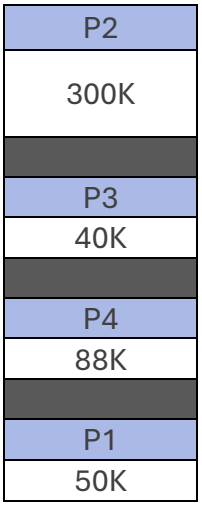
THUẬT TOÁN CẤP PHÁT BỘ NHỚ BEST FIT

Nguyên tắc: Lần lượt cho các tiến trình vào ô có **kích thước bé nhất vừa đủ với kích thước tiến trình.**

Ví dụ trực quan

Giả sử bộ nhớ chính được phân chia thành các vùng có kích thước 600K, 100K, 200K, 300K (theo thứ tự). Cho biết các tiến trình có kích thước 250K, 300K, 60K, 112K, 426K (theo thứ tự) sẽ được cấp phát bộ nhớ như thế nào nếu dùng thuật toán **Best fit**?

Bài giải

Bộ nhớ khi chưa cấp phát	**Cấp phát theo thuật toán Best fit**
	 Không còn đủ bộ nhớ để cấp phát cho P5

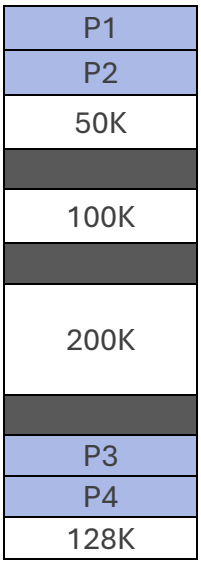
THUẬT TOÁN CẤP PHÁT BỘ NHỚ WORST FIT

Nguyên tắc: Lần lượt cho các tiến trình vào các ô có **kích thước lớn nhất**.

Ví dụ trực quan

Giả sử bộ nhớ chính được phân chia thành các vùng có kích thước 600K, 100K, 200K, 300K (theo thứ tự). Cho biết các tiến trình có kích thước 250K, 300K, 60K, 112K, 426K (theo thứ tự) sẽ được cấp phát bộ nhớ như thế nào nếu dùng thuật toán **Worst fit**?

Bài giải

Bộ nhớ khi chưa cấp phát	Cấp phát theo thuật toán Worst fit
	 Không còn đủ bộ nhớ để cấp phát cho P5

ĐÁNH GIÁ ĐỘ HIỆU QUẢ CỦA CÁC THUẬT TOÁN CẤP PHÁT BỘ NHỚ

Nếu đề bài yêu cầu sử dụng từ 02 thuật toán cấp phát bộ nhớ trở lên và yêu cầu đánh giá thuật toán nào hiệu quả hơn thì ta làm như sau:

Bước 1: Xét về khả năng đáp ứng bộ nhớ cho các tiến trình:

- Nếu có duy nhất 01 thuật toán đáp ứng đủ bộ nhớ cho tất cả các tiến trình thì đó là thuật toán hiệu quả.
- Nếu nhiều hơn 01 thuật toán đáp ứng đủ bộ nhớ cho tất cả các tiến trình hoặc các thuật toán đều không thể đáp ứng bộ nhớ cho các tiến trình thì xét đến bước 2.

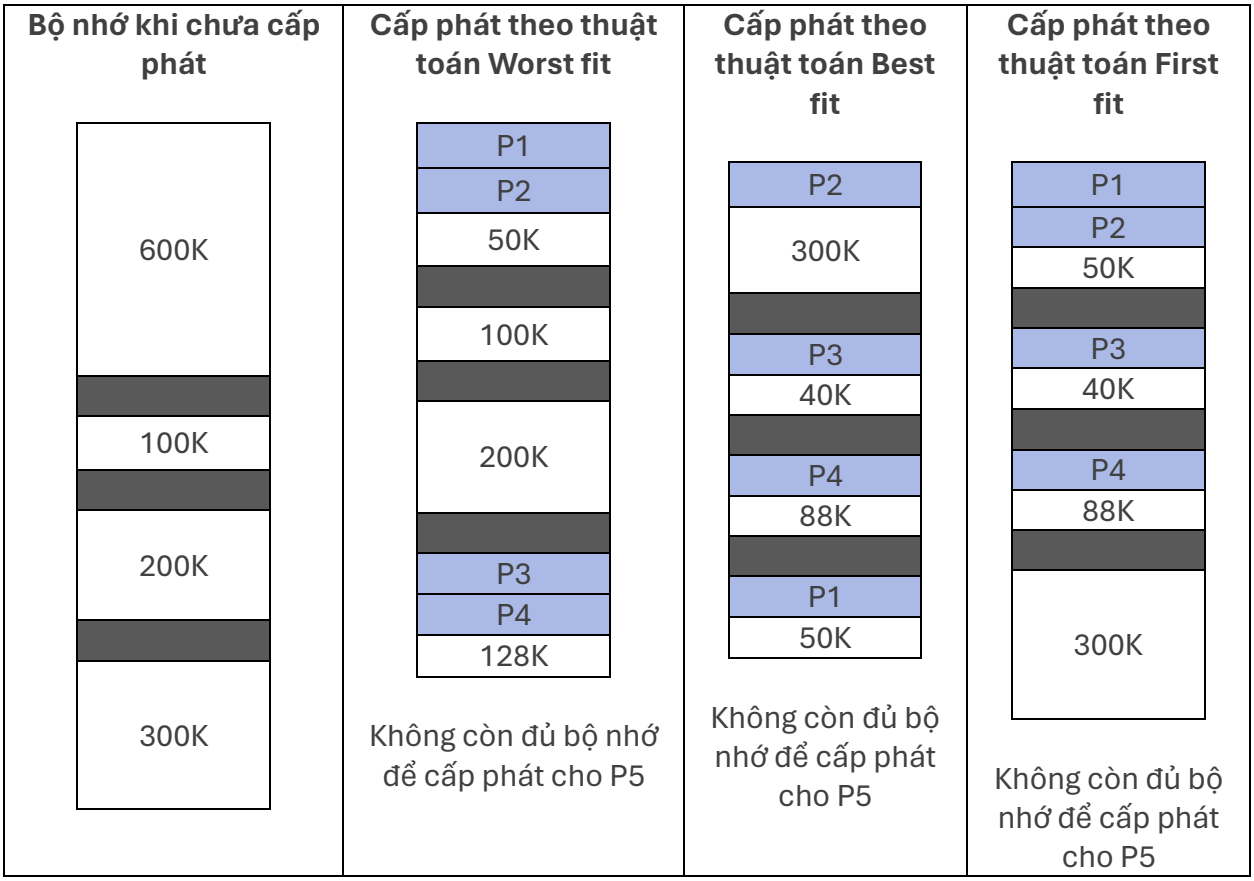
Bước 2: Xét đến ưu điểm của các giải thuật (và đây cũng là thứ tự ưu tiên lựa chọn):

- *First fit*: Tốc độ nhanh
- *Best fit*: Hạn chế phân mảnh ngoại vi
- *Worst fit*: Tăng cường khả năng mở rộng

Ví dụ trực quan

Giả sử bộ nhớ chính được phân chia thành các vùng có kích thước 600K, 100K, 200K, 300K (theo thứ tự). Cho biết các tiến trình có kích thước 250K, 300K, 60K, 112K, 426K (theo thứ tự) sẽ được cấp phát bộ nhớ như thế nào nếu dùng thuật toán **First fit, Best fit, Worst fit**? Thuật toán nào cho phép **sử dụng bộ nhớ hiệu quả** nhất?

Bài giải



Ta thấy cả 3 thuật toán đều không đủ bộ nhớ để cấp phát cho P5.

Do đề bài hỏi thuật toán nào sử dụng bộ nhớ hiệu quả nhất nên trong trường hợp này **phải ưu tiên tốc độ**. => Thuật toán **First Fit** là hiệu quả nhất.

DẠNG BÀI SỐ 03
KỸ THUẬT THAY TRANG

KIẾN THỨC CƠ SỞ

Lỗi thay trang

Lỗi thay trang xuất hiện **khi một trang bất kỳ bị thay thế** hoặc khi **bộ nhớ chính chưa chứa đầy trang**.

Trong tài liệu này, bộ nhớ chính được giả định sử dụng 4 frames (4 ô nhớ trang). Với số lượng frames khác cũng làm tương tự.

THUẬT TOÁN THAY TRANG FIRST IN FIRST OUT (FIFO)

Nguyên tắc: Lần lượt cho các trang **theo thứ tự** vào bộ nhớ chính. Nếu cần thay thế trang, trang ở **trong bộ nhớ lâu nhất** sẽ bị thay thế.

Ưu điểm: Đơn giản

Nhược điểm: Không tối ưu, cần thay trang nhiều lần

Ví dụ trực quan

Cho chuỗi truy xuất:

1, 3, 1, 4, 2, 1, 5, 6, 2, 1, 2, 3, 6, 7, 3, 3, 2, 1, 2, 3, 6

Thực hiện kỹ thuật thay trang cho chuỗi truy xuất bên trên bằng kỹ thuật thay trang **FIFO**?

Bài giải

1	3	1	4	2	1	5	6	2	1	2	3	6	7	3	3	2	1	2	3	6
1	1		1	1		5	5		5		5		7			7				7
	3		3	3		3	6		6		6		6			2				2
			4	4		4	4		1		1		1			1				6
				2		2	2		2		3		3			3				3
*	*	*	*			*	*		*		*		*			*				*

⇨ Có tổng cộng 11 lỗi *(các dấu *. Các cột bị để trống là tại đó không có gì thay đổi trong bộ nhớ chính)*

THUẬT TOÁN THAY TRANG TỐI ƯU (OPT)

Nguyên tắc: Lần lượt cho các trang **theo thứ tự** vào bộ nhớ chính. Nếu cần thay thế trang, trang **xa nhất được dùng trong tương lai** sẽ bị thay thế.

Ưu điểm: Tối ưu

Nhược điểm: Phải dự báo trước

Ví dụ trực quan

Cho chuỗi truy xuất:

1, 3, 1, 4, 2, 1, 5, 6, 2, 1, 2, 3, 6, 7, 3, 3, 2, 1, 2, 3, 6

Thực hiện kỹ thuật thay trang cho chuỗi truy xuất bên trên bằng kỹ thuật thay trang **OPT**?

Bài giải

1	3	1	4	2	1	5	6	2	1	2	3	6	7	3	3	2	1	2	3	6
1	1		1	1		1	1						1							6
	3		3	3		3	3						3							3
			4	4		5	6						7							7
				2		2	2						2							2
*	*	*	*			*	*						*							*

⇨ Có tổng cộng 8 lỗi *(các dấu *. Các cột bị để trống là tại đó không có gì thay đổi trong bộ nhớ chính)*

Giải thích bài làm

Ở trước khi thực hiện bước thứ 7, ta thấy trong bộ nhớ đang có các trang 1, 3, 4, 2 và cần thay trang 5 vào. Nhìn về tương lai, ta thấy:

- Trang 1 xuất hiện ở bước 10
- Trang 3 xuất hiện ở bước 12
- Trang 4 không xuất hiện nữa
- Trang 2 xuất hiện ở bước 11

⇨ Cần thay trang 4.

Tương tự với các bước còn lại.

Đối với bước cuối cùng, ta thấy cần thay trang 6 vào bộ nhớ; nhưng không còn tương lai để nhìn về nữa, nên ta thay trang đầu tiên tìm được trong bộ nhớ (trang 1) thành trang 6.

THUẬT TOÁN THAY TRANG LEAST RECENTLY USED (LRU)

Nguyên tắc: Lần lượt cho các trang **theo thứ tự** vào bộ nhớ chính. Nếu cần thay thế trang, trang **xa nhất được dùng trong quá khứ** sẽ bị thay thế.

Ưu điểm: Tối ưu

Nhược điểm: Tốn bộ nhớ

Ví dụ trực quan

Cho chuỗi truy xuất:

1, 3, 1, 4, 2, 1, 5, 6, 2, 1, 2, 3, 6, 7, 3, 3, 2, 1, 2, 3, 6

Thực hiện kỹ thuật thay trang cho chuỗi truy xuất bên trên bằng kỹ thuật thay trang **LRU**?

Bài giải

1	3	1	4	2	1	5	6	2	1	2	3	6	7	3	3	2	1	2	3	6
1	1		1	1		1	1				1		7				7			6
	3		3	3		5	5				3		3				3			3
			4	4		4	6				6		6				1			1
				2		2	2				2		2				2			2
*	*	*	*			*	*				*		*				*			*

⇨ Có tổng cộng 10 lỗi *(các dấu *. Các cột bị để trống là tại đó không có gì thay đổi trong bộ nhớ chính)*

Giải thích bài làm

Ở trước khi thực hiện bước thứ 7, ta thấy bộ nhớ đang có các trang 1, 3, 4, 2. Lúc này cần thay trang 5 vào. Nhận thấy:

- Trang 1 xuất hiện ở bước 6
- Trang 3 xuất hiện ở bước 2
- Trang 4 xuất hiện ở bước 4
- Trang 2 xuất hiện ở bước 5

⇨ Thay trang 3.

Các bước khác làm tương tự.

DẠNG BÀI SỐ 04
BÀI TOÁN LIÊN QUAN ĐẾN ĐẦU ĐỌC CYLINDER (LẬP LỊCH ĐĨA)

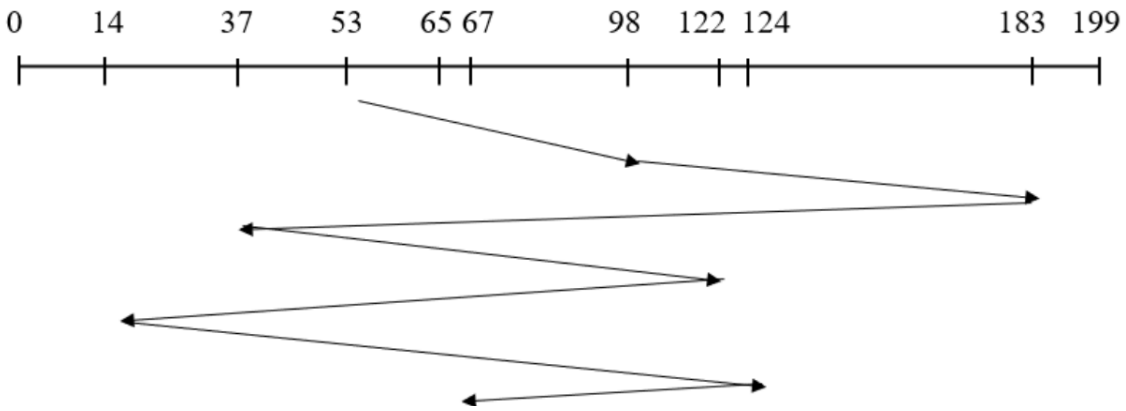
KỸ THUẬT QUÉT THEO HÀNG ĐỢI FIRST COME FIRST SERVE (FCFS)

Nguyên lý: Di chuyển đầu đọc để quét theo thứ tự vào hệ thống (thứ tự của đề bài).

Ví dụ trực quan

Giả sử đĩa cứng có 200 cylinders (0 – 199) được đánh số từ trong ra ngoài. Cần đọc các khối cylinder: 98, 183, 37, 122, 14, 124, 65, 67. Giả sử hiện tại đầu đọc đang ở vị trí 53. Xác định số cylinder cần đọc nếu dùng thuật toán **FCFS**?

Bài giải



⇨ Tổng số cylinder cần đọc là $(183-53)+(183-37)+(122-37)+(122-14)+(124-14)+(124-67)=636$ cylinders

Giải thích bài làm

Ta lần lượt quét đến các vị trí theo thứ tự đề bài: 98, 183, 37, 122, 14, 124, 65, 67.

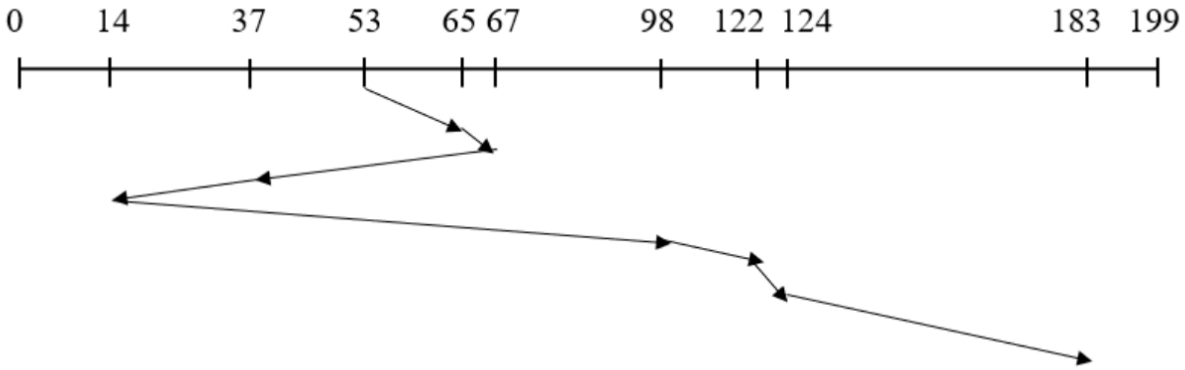
KỸ THUẬT QUÉT SHORTEST SEEK TIME FIRST (SSTF)

Nguyên lý: Lần lượt quét đến các vị trí gần đầu đọc nhất.

Ví dụ trực quan

Giả sử đĩa cứng có 200 cylinders (0 – 199) được đánh số từ trong ra ngoài. Cần đọc các khối cylinder: 98, 183, 37, 122, 14, 124, 65, 67. Giả sử hiện tại đầu đọc đang ở vị trí 53. Xác định số cylinder cần đọc nếu dùng thuật toán **SSTF**?

Bài giải



⇨ Tổng số cylinder cần đọc là $(67 - 53) + (67 - 14) + (183 - 14) = 236$

Giải thích bài làm

Đầu đọc đang ở vị trí 53; do đó, 65 gần đầu đọc nhất => Quét đến 65.
Đầu đọc đang ở vị trí 65; do đó, 67 gần đầu đọc nhất => Quét đến 67
Đầu đọc đang ở vị trí 67; do đó, 37 gần đầu đọc nhất (67 – 37 = 30; trong khi 98 – 67 = 31) => Quét đến 37
Tương tự với các vị trí còn lại.

KIẾN THỨC CƠ SỞ

Từ trong ra ngoài
Từ phải qua trái

Từ ngoài vào trong
Từ trái qua phải

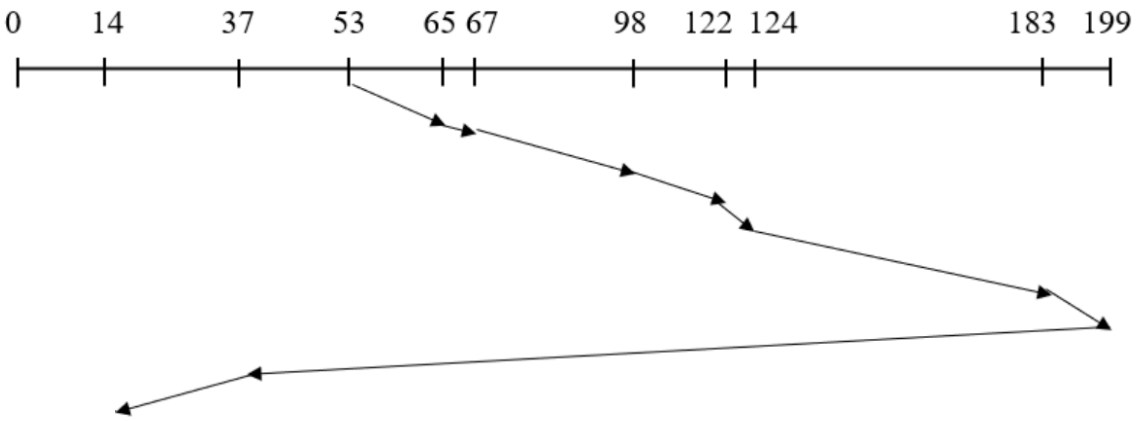
KỸ THUẬT QUÉT THANG MÁY (SCAN)

Nguyên lý: Quét đến các vị trí **theo chiều** (*trong ra ngoài* hoặc *ngoài vào trong*) cho trước; **chạm vào biên**; sau đó quay lại quét các vị trí ở chiều ngược lại **(không chạm biên nữa)**.

Ví dụ trực quan

Giả sử đĩa cứng có 200 cylinders (0 – 199) được đánh số từ trong ra ngoài. Cần đọc các khối cylinder: 98, 183, 37, 122, 14, 124, 65, 67. Giả sử hiện tại đầu đọc đang ở vị trí 53. Xác định số cylinder cần đọc nếu dùng thuật toán **SCAN** *(Giả sử lúc bắt đầu, đầu đọc đang đi từ ngoài vào trong)*?

Bài giải



⇨ Tổng số cylinder cần đọc là $(199 - 53) + (199 - 14) = 331$ cylinders.

Giải thích bài làm

- Đầu đọc đang ở vị trí 53 và chiều quét là từ ngoài vào trong (là từ trái qua phải) nên ta quét hết các điểm nằm ở bên phải 53.
- Sau khi quét đến 183, cần phải chạm biên 199
- Đầu đọc quay lại, quét về điểm 37. Cuối cùng là quét điểm 14.

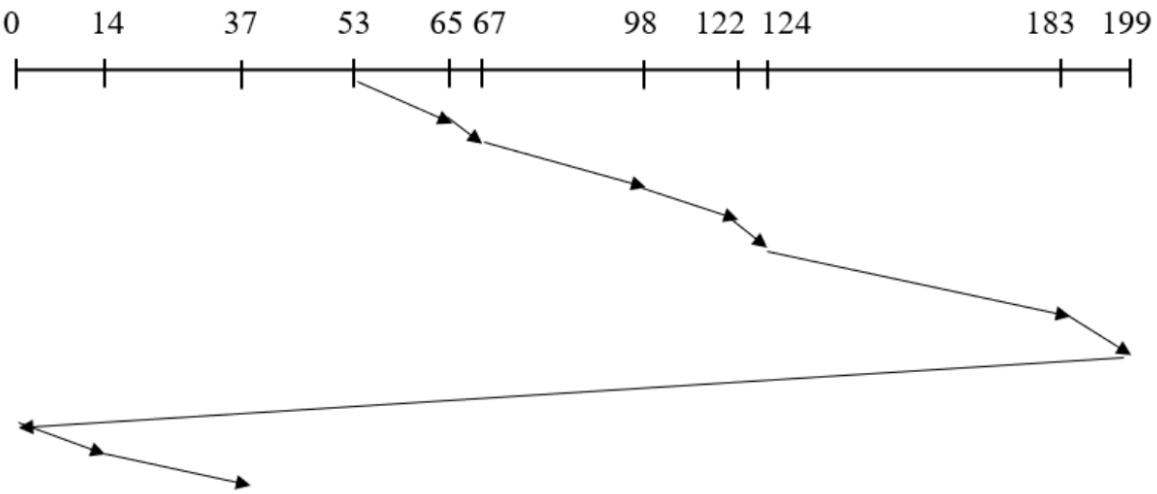
KỸ THUẬT QUÉT CIRCULAR SCAN (C–SCAN)

Nguyên lý: Quét đến các vị trí **theo chiều** (*trong ra ngoài* hoặc *ngoài vào trong*) cho trước; **chạm vào biên**; sau đó quay lại **chạm vào biên còn lại** mà không quét bất kỳ điểm nào trên đường đến biên đó. Cuối cùng, quét các điểm còn lại.

Ví dụ trực quan

Giả sử đĩa cứng có 200 cylinders (0 – 199) được đánh số từ trong ra ngoài. Cần đọc các khối cylinder: 98, 183, 37, 122, 14, 124, 65, 67. Giả sử hiện tại đầu đọc đang ở vị trí 53. Xác định số cylinder cần đọc nếu dùng thuật toán **C-SCAN** *(Giả sử lúc bắt đầu, đầu đọc đang đi từ ngoài vào trong)*?

Bài giải



⇨ Tổng số cylinder cần đọc là: $(199 - 53) + (199 - 0) + (37 - 0) = 382$

Giải thích bài làm

- Đầu tiên, đầu đọc đang ở vị trí 53 và chiều đọc đang đi từ ngoài vào trong (trái qua phải); do đó, ta quét hết các điểm nằm ở bên phải 53.
- Khi quét xong 183, cần phải chạm biên 199.
- Lúc này, lập tức di chuyển đầu đọc về biên còn lại (biên 0) mà trên đường đi **KHÔNG QUÉT** điểm nào.
- Từ biên 0, tiếp tục quét các điểm còn lại (14 và 37)

KỸ THUẬT QUÉT LOOK

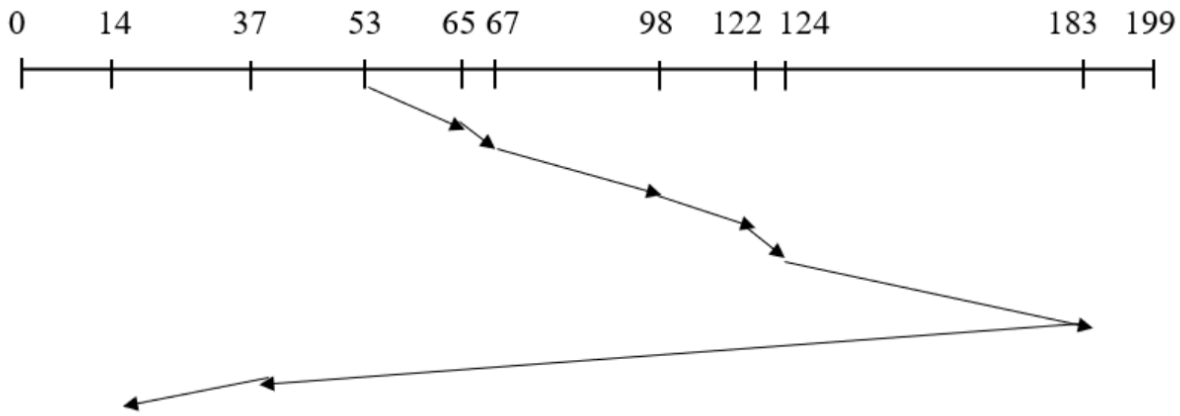
Nguyên lý: Quét đến các vị trí **theo chiều** (*trong ra ngoài* hoặc *ngoài vào trong*) cho trước; **KHÔNG chạm vào biên**; sau đó quay lại quét các vị trí ở chiều ngược lại.

Ví dụ trực quan

Giả sử đĩa cứng có 200 cylinders (0 – 199) được đánh số từ trong ra ngoài. Cần đọc các khối cylinder: 98, 183, 37, 122, 14, 124, 65, 67. Giả sử hiện tại đầu đọc đang ở vị trí 53.

Xác định số cylinder cần đọc nếu dùng thuật toán **LOOK** *(Giả sử lúc bắt đầu, đầu đọc đang đi từ ngoài vào trong)*?

Bài giải



⇨ Tổng số cylinder cần đọc là $(183 - 53) + (183 - 14) = 299$ cylinders.

Giải thích bài làm

- Đầu đọc đang ở vị trí 53 và chiều quét là từ ngoài vào trong (là từ trái qua phải) nên ta quét hết các điểm nằm ở bên phải 53.
- Sau khi quét đến 183, lập tức quay đầu đọc lại.
- Quét về điểm 37. Cuối cùng là quét điểm 14.

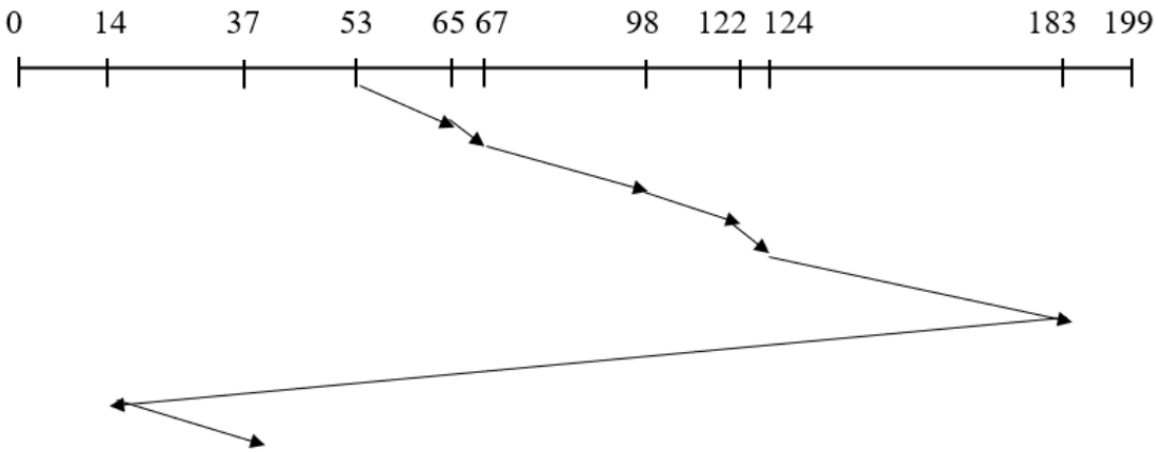
KỸ THUẬT QUÉT CIRCULAR LOOK (C-LOOK)

Nguyên lý: Quét đến các vị trí **theo chiều** (*trong ra ngoài* hoặc *ngoài vào trong*) cho trước; **KHÔNG chạm vào biên**; sau đó quay lại **chạm vào điểm xa nhất so với chiều hiện tại** mà không quét bất kỳ điểm nào trên đường đến điểm đó. Cuối cùng, quét các điểm còn lại.

Ví dụ trực quan

Giả sử đĩa cứng có 200 cylinders (0 – 199) được đánh số từ trong ra ngoài. Cần đọc các khối cylinder: 98, 183, 37, 122, 14, 124, 65, 67. Giả sử hiện tại đầu đọc đang ở vị trí 53. Xác định số cylinder cần đọc nếu dùng thuật toán **C-LOOK** *(Giả sử lúc bắt đầu, đầu đọc đang đi từ ngoài vào trong)*?

Bài giải



⇨ Tổng số cylinder cần đọc là: $(183 - 53) + (183 - 14) + (37 - 14) = 322$

Giải thích bài làm

- Đầu tiên, đầu đọc đang ở vị trí 53 và chiều đọc đang đi từ ngoài vào trong (trái qua phải); do đó, ta quét hết các điểm nằm ở bên phải 53.
- Khi quét xong 183, lập tức quay đầu đọc lại.
- Ta thấy, điểm xa đầu đọc nhất lúc này là 14. Quét về điểm 14.
- Từ điểm 14, tiếp tục quét điểm còn lại (37)

DẠNG BÀI SỐ 05
TÍNH TOÁN THỜI GIAN ĐƯA DỮ LIỆU TRÊN Ổ ĐĨA VÀO BỘ NHỚ

KIẾN THỨC CƠ SỞ

Công thức tính thời gian trễ trung bình

$$\text{Thời gian trễ trung bình} = \frac{1}{2}.\text{Thời gian trễ} = \frac{1}{2}.\frac{60}{\text{Số vòng / phút}} \text{ (đơn vị: giây)}$$

Thời gian truy cập trễ

$$\text{Thời gian truy cập trễ} = \text{Thời gian truy cập trung bình}$$
$$= \text{Thời gian tìm kiếm trung bình} + \text{Thời gian trễ trung bình}$$

Thời gian I/O trung bình

$$\text{Thời gian I/O trung bình} = \text{Thời gian truy cập trung bình} + \frac{\text{Lượng truyền vào}}{\text{Tốc độ truyền}}$$
$$+ \text{Thời gian xử lý trên đầu đọc}$$

Ví dụ trực quan

Tính toán thời gian để đưa 4MB dữ liệu trên ổ đĩa vào bộ nhớ. Biết rằng tốc độ đọc dữ liệu trên ổ đĩa là 1MB/s, thời gian xử lý trên đầu đọc là 0.1 ms, đĩa quay với tốc độ 6500 vòng/phút và thời gian trung bình tìm dữ liệu (seek time) là 10 ms.

Bài giải

Thời gian trễ trung bình: $\frac{1}{2}.\frac{60}{6500} = 0{,}509$ (giây)

Thời gian truy cập trung bình: $10 + 0{,}509.1000 = 519$ (ms)

Thời gian I/O trung bình: $519 + \frac{4}{1.1000} + 0{,}1 = 519{,}104$ (ms)

DẠNG BÀI SỐ 06
GIẢI THUẬT NHÀ BĂNG (BANKER)

Đề bài

Cho ma trận yêu cầu cấp phát như sau:

	Allocation				Max				Available			
	A	**B**	**C**	**...**	**A**	**B**	**C**	**...**	**A**	**B**	**C**	**...**
P0	?	?	?	...	?	?	?	...	α	β	γ	...
P1	?	?	?	...	?	?	?	...				
...	...	...	...	...	...	...	...	...				
Pn	?	?	?	...	?	?	?	...				

TÍNH MA TRẬN NEED

Phương pháp giải

Ta có bảng ma trận Need như sau:

	Need			
	A	**B**	**C**	**...**
P0				
P1				
...				
Pn				

Điền các giá trị của ma trận Need bằng cách sử dụng công thức:

$$Need = Max - Allocation$$

Ví dụ: Tiến trình P0 có tài nguyên A có: Allocation=3; Max=5 ➔ Giá trị Need của tài nguyên A của tiến trình P0 bằng 5-3=2

Ví dụ trực quan

Cho ma trận yêu cầu cấp phát như sau:

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	3	3	1	4	4	2	1	1	1
P1	2	1	2	4	3	2			
P2	3	2	1	7	4	2			
P3	3	2	3	4	3	3			
P4	1	2	1	3	2	2			

Tính **ma trận Need**?

Bài giải

Ta có ma trận Need như sau:

	Need		
	A	B	C
P0	1	1	1
P1	2	2	0
P2	4	2	1
P3	1	1	0
P4	2	0	1

TÍNH TỔNG SỐ THỂ HIỆN CỦA CÁC TÀI NGUYÊN

Phương pháp giải

Với tài nguyên r bất kỳ, ta tính được tổng số thể hiện của chúng bằng công thức:

$$\text{Tổng số thể hiện của r} = \text{Tổng Allocation của r} + \text{Available của r}$$

Ví dụ trực quan

Cho ma trận yêu cầu cấp phát như sau:

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	3	3	1	4	4	2	1	1	1
P1	2	1	2	4	3	2			
P2	3	2	1	7	4	2			
P3	3	2	3	4	3	3			
P4	1	2	1	3	2	2			

Tính tổng số thể hiện của các tài nguyên?

Bài giải

- Tổng số thể hiện của A: $(3 + 2 + 3 + 3 + 1) + 1 = 13$ thể hiện
- Tổng số thể hiện của B: $(3 + 1 + 2 + 2 + 2) + 1 = 11$ thể hiện

- Tổng số thể hiện của C: $(1 + 2 + 1 + 3 + 1) + 1 = 9$ thể hiện

KIỂM TRA HỆ THỐNG CÓ Ở TRẠNG THÁI AN TOÀN HAY KHÔNG

Phương pháp giải

Trước hết ta cần tính ma trận Work

Tiến trình	Work			
	A	B	C	...
	α	β	γ	...

Để tính các giá trị của ma trận Work ta làm như sau:

- **Bước 1:** Xét tiến trình P_i. Nếu các giá trị của P_i trong **ma trận Need** đều $\leq$ giá trị trong ma trận Work hiện tại thì chuyển đến bước 2. Nếu không thì xét tiếp đến tiến trình khác.
- **Bước 2:** Thực hiện cấp phát cho P_i. Giá trị trong ma trận Work mới sẽ bằng:

$$Work\ mới = Work\ cũ + Allocation\ của\ P_i$$

- **Bước 3:** Kiểm tra xem còn tiến trình nào chưa được cấp phát không, nếu còn thì quay lại bước 1. Nếu không còn thì kết thúc giải thuật.

Nếu:

- Có thể cấp phát hết cho tất cả các tiến trình => Hệ thống ở trạng thái an toàn
- Không thể cấp phát hết cho tất cả các tiến trình => Hệ thống không ở trạng thái an toàn

Ví dụ trực quan

Cho ma trận yêu cầu cấp phát như sau:

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	3	3	1	4	4	2	1	1	1
P1	2	1	2	4	3	2			
P2	3	2	1	7	4	2			
P3	3	2	3	4	3	3			
P4	1	2	1	3	2	2			

Kiểm tra xem hệ thống có ở trạng thái an toàn hay không? Tại sao?

Bài giải

Trước hết ta có ma trận Need:

	Need		
	A	**B**	**C**
P0	1	1	1
P1	2	2	0
P2	4	2	1
P3	1	1	0
P4	2	0	1

Tiếp tục, ta khởi tạo ma trận Work:

Tiến trình	**Work**		
	A	**B**	**C**
	1	**1**	**1**

- Xét tiến trình P0, ta thấy giá trị trong ma trận Need của P0 là 1, 1, 1. Cả 3 giá trị này đều ≤ giá trị trong ma trận Work hiện tại (1, 1, 1) nên ta thực hiện cấp phát cho P0. Và giá trị trong ma trận Work mới sẽ bằng Work hiện tại + Allocation của P0.

Tiến trình	**Work**		
	A	**B**	**C**
	1	**1**	**1**
P0	4	4	2

- Xét tiến trình P1, ta thấy giá trị ma trận Need của P1 là 2, 2, 0. Cả 3 giá trị này đều ≤ giá trị trong ma trận Work hiện tại (4, 4, 2) nên ta thực hiện cấp phát cho P1 tương tự như bên trên.

Tiến trình	**Work**		
	A	**B**	**C**
	1	**1**	**1**
P0	4	4	2
P1	6	5	4

Thực hiện tương tự với các tiến trình còn lại, ta được ma trận Work như sau:

Tiến trình	Work		
	A	**B**	**C**
	1	**1**	**1**
P0	4	4	2
P1	6	5	4
P2	9	7	5
P3	12	9	8
P4	13	11	9

*(**Mẹo**: Ta có thể kiểm tra xem ma trận Work đã đúng hay chưa bằng cách nhìn vào kết quả cuối cùng. Nếu kết quả cuối cùng ra đúng bằng số thể hiện của từng tài nguyên thì đáp án của ta là đáp án chính xác. Ở đây, ta thấy kết quả cuối cùng A=13, B=11, C=9 – khớp với số thể hiện của từng tài nguyên => Ta đã tính đúng)*

Ta thấy có thể cấp phát hết cho toàn bộ các tiến trình. Vậy hệ thống này có ở trạng thái an toàn; vì tồn tại thứ tự cấp phát an toàn là P0, P1, P2, P3, P4.

DẠNG BÀI SỐ 07
HOẠT ĐỘNG ĐĨA I/O

KIẾN THỨC CƠ SỞ

Việc phân bố hoạt động đĩa I/O có n khối bao gồm các thao tác sau:
- Thêm/Xoá khối ở đầu
- Thêm/Xoá khối ở cuối
- Thêm khối ở sau khối i bất kỳ
- Xoá khối ở vị trí i

Gọi n là số khối của tập tin.

CHIẾN LƯỢC PHÂN BỐ LIÊN TỤC (CONTIGUOUS)

Hành động	Số hoạt động cần thực hiện	Giải thích
Thêm khối ở đầu	$2n + 1$	- 2n: n lần đọc và n lần ghi để dịch n khối về sau - 1: ghi khối mới
Thêm khối ở cuối	1	- 1: ghi khối mới
Thêm khối sau khối i	$2(n - i) + 1$	- 2(n-i): (n-i) lần đọc và (n-i) lần ghi để dịch (n-i) khối đằng sau khối i về sau. - 1: ghi khối mới
Xoá khối ở đầu	0	Cập nhật là đã trống
Xoá khối ở cuối	0	Cập nhật là đã trống
Xoá khối ở vị trí i	$2(n - i)$	- 2(n-i): (n-i) lần đọc và (n-i) lần ghi để dịch (n-i) khối đằng sau khối i về sau.

CHIẾN LƯỢC PHÂN BỐ LIÊN KẾT (LINKED)

Hành động	Số hoạt động cần thực hiện	Giải thích
Thêm khối ở đầu	1	- 1: Thay đổi con trỏ trong khối điều khiển tệp để trỏ đến khối mới trong đĩa

Thêm khối ở cuối	3	- 1: Đọc khối cuối - 1: Ghi khối mới - 1: Cập nhật con trỏ
Thêm khối sau khối i	$i + 2$	- i: Đọc i khối đầu - 1: Ghi khối mới - 1: Cập nhật con trỏ
Xoá khối ở đầu	1	Cập nhật con trỏ
Xoá khối ở cuối	n	- n-1: Đọc n-1 khối đầu - 1: Cập nhật con trỏ
Xoá khối ở vị trí i	$i + 1$	- i: Đọc i khối đầu tiên - 1: Cập nhật con trỏ

CHIẾN LƯỢC PHÂN BỐ LẬP CHỈ MỤC (INDEX)

Hành động	Số hoạt động cần thực hiện	Giải thích
Thêm khối ở đầu	1	Ghi khối mới và cập nhật bộ nhớ
Thêm khối ở cuối	1	Ghi khối mới và cập nhật bộ nhớ
Thêm khối sau khối i	1	Ghi khối mới và cập nhật bộ nhớ
Xoá khối ở đầu	0	Chỉ cập nhật bộ nhớ
Xoá khối ở cuối	0	Chỉ cập nhật bộ nhớ
Xoá khối ở vị trí i	0	Chỉ cập nhật bộ nhớ

VÍ DỤ TRỰC QUAN

Xem xét một tập tin gồm 240 khối. Tính toán và giải thích xem cần bao nhiêu hoạt động đĩa I/O cần thiết để:

a) Xoá khối 45 với chiến lược phân bố liên tục, liên kết và lập chỉ mục?
b) Thêm khối sau khối 140 với chiến lược phân bố liên tục, liên kết và lập chỉ mục?

Bài giải

	Liên tục	Liên kết	Chỉ mục
Xoá khối 45	390	46	0
Thêm khối sau khối 140	201	142	1

DẠNG BÀI SỐ 08
BÀI TOÁN CHMOD

KIẾN THỨC CƠ SỞ

Cấu trúc thông tin phân quyền của một tập tin

Gồm 9 bit, có dạng **??? ??? ???**.

Mỗi bộ ba ??? lần lượt tương ứng với ba quyền:

- **r (read):** Quyền đọc
- **w (write):** Quyền ghi
- **x (execute):** Quyền thực thi

Đồng thời, các bộ 3 tương ứng với phân quyền:

chủ sở hữu – người dùng nhóm – người dùng khác

GIẢI THÍCH CHẾ ĐỘ PHÂN QUYỀN

Ví dụ trực quan

Cho một file có thông tin phân quyền như sau:

rw-r--r--

Giải thích chế độ phân quyền này?

Bài giải

- Chủ sở hữu: rw- => Có quyền đọc và ghi file
- Người dùng nhóm: r-- => Có quyền đọc file
- Người dùng khác: r-- => Có quyền đọc file

SỬ DỤNG LỆNH CHMOD ĐỂ PHÂN QUYỀN

Đề bài

Cho một file có thông tin phân quyền như sau:

??? ??? ???

Sử dụng lệnh chmod với tham số thế nào để được phân quyền như file trên?

Phương pháp giải

Ta sẽ xét từng bộ 03 ??? một *(nghĩa là xét từng nhóm người dùng)*:

- **Bước 1:** Chuyển ??? về dạng **nhị phân 3 bit** theo quy tắc:

$$?\ \text{mang giá trị} \begin{cases} - \Rightarrow 0 \\ \text{r, w, x} \Rightarrow 1 \end{cases}$$

- **Bước 2:** Chuyển số nhị phân 3 bit vừa thu được về dạng **số thập phân**

Sau khi xét xong 03 bộ ??? bằng các bước trên, ta thu được **3 số thập phân** $\boldsymbol{\alpha_1; \alpha_2; \alpha_3}$.

⇨ Sẽ dùng lệnh chmod với tham số $\boldsymbol{\alpha_1\alpha_2\alpha_3}$ để được phân quyền như file đề bài cho.

Ví dụ trực quan

Cho một file có thông tin phân quyền như sau:

rw-r--r--

Sử dụng lệnh chmod với tham số thế nào để được phân quyền như file trên?

Bài giải

Ta có:

	rw-	r--	r--
Dạng nhị phân	110	100	100
Dạng thập phân	6	4	4

⇨ Cần sử dụng lệnh chmod với tham số 644 để được phân quyền như đề bài

DẠNG BÀI SỐ 09
MA TRẬN QUYỀN TRUY CẬP

KIẾN THỨC CƠ SỞ

Ma trận quyền truy cập

Là ma trận mà có các dòng là các miền D; các cột là các đối tượng (bao gồm file, máy in, hoặc chính các miền)

Một miền D bất kỳ có thể thực hiện các lệnh:

- **Đọc**: Quyền được đọc file
- **Ghi**: Quyền được ghi file
- **Đọc***: Trao cho một miền khác quyền đọc file
- **Ghi***: Trao cho một miền khác quyền ghi file
- **Đọc+**: Chuyển quyền đọc file cho miền khác, sau đó miền D sẽ mất quyền đọc file
- **Ghi+**: Chuyển quyền ghi file cho miền khác, sau đó miền D sẽ mất quyền ghi file
- **Sở hữu (thường là của các đối tượng, không bao gồm đối tượng miền):** Miền D có quyền cấp phát hoặc huỷ quyền của các miền khác trên file nó sở hữu
- **Điều khiển (thường là của đối tượng miền):** Miền D có quyền cấp phát hoặc huỷ quyền nó có cho miền nó điều khiển
- **Chuyển**: Có thể chuyển qua miền khác để thực hiện các quyền.

GIẢI THÍCH CÁC QUYỀN KHÓ HIỂU (???)

A – QUYỀN ĐỌC* VÀ GHI*:

- Quyền **Đọc*** và **Ghi*** của một miền trên một file sẽ **có quyền** Đọc và Ghi file này; đồng thời **có thể** cấp quyền Đọc và Ghi cho các miền khác.

Ví dụ:

Có ma trận quyền truy cập như sau:

	F1	F2	F3	Máy in	D1	D2	D3	D4
D1	Đọc		Ghi					Chuyển
D2	Đọc*			In			Sở hữu	Chuyển
D3		Sở hữu	Ghi*					Chuyển
D4		Ghi+						

- Trước hết ta thấy miền D2 có quyền **Đọc*** tệp F1. Điều này nghĩa là:
  + Miền D2 **có quyền** đọc tệp F1
  + Miền D2 có thể trao quyền đọc tệp F1 cho các miền còn lại (D1, D3, D4). Nói cách khác, miền D1, D3, D4 **có cơ hội/có thể** đọc tệp F1.
- Tương tự, ta thấy miền D3 có quyền **Ghi*** tệp F3. Điều này nghĩa là:
  + Miền D3 **có quyền** ghi tệp F3
  + Miền D3 có thể trao quyền ghi tệp F3 cho các miền còn lại (D1, D2, D4). Nói cách khác, miền D1, D2, D4 **có cơ hội/có thể** ghi tệp F3.

B – QUYỀN SỞ HỮU (CỦA FILE):

- Quyền **Sở hữu** của một miền trên một đối tượng (file, máy in) nghĩa là miền này có quyền **đọc, ghi và thực thi** (đối với file) và **in** (đối với máy in); đồng thời nó **có thể** cấp phát quyền vừa nêu cho các miền khác.

Ví dụ:

Có ma trận quyền truy cập như sau:

	F1	F2	F3	Máy in	D1	D2	D3	D4
D1	Đọc		Ghi					Chuyển
D2	Đọc*			In			Điều khiển	Chuyển
D3		Sở hữu	Ghi*					Chuyển
D4		Ghi+						

Ta thấy miền D3 có quyền **sở hữu** file F2. Như vậy, D3 có thể **đọc, ghi và thực thi** file F2. Đồng thời, các miền D1, D2, D4 **có cơ hội/có thể** có mọi quyền với F2 nếu được D3 cấp phát.

C – QUYỀN ĐIỀU KHIỂN:

- Quyền **Điều khiển** của một miền trên một miền khác nghĩa là miền này **có thể** có các quyền mà miền nó điều khiển sở hữu.

Ví dụ:

Có ma trận quyền truy cập như sau:

	F1	F2	F3	Máy in	D1	D2	D3	D4
D1	Đọc		Ghi					Chuyển
D2	Đọc*			In			Điều khiển	Chuyển
D3		Sở hữu	Ghi*					Chuyển
D4		Ghi+						

Ta thấy D2 **sở hữu** quyền điều khiển **D3.** Như vậy, D2 **có cơ hội/có thể** có mọi quyền với F2 và ghi F3.

D – QUYỀN CHUYỂN:

- Quyền **Chuyển** của một miền D sang một miền khác nghĩa là miền D đó sẽ **có cơ hội** thực hiện các quyền không phải quyền đặc biệt (bao gồm: đọc, ghi, in, thực thi) nếu có.

Ví dụ:

Có ma trận quyền truy cập như sau:

	F1	F2	F3	Máy in	D1	D2	D3	D4
D1	Đọc		Ghi					Chuyển
D2	Đọc*			In			Điều khiển	Chuyển
D3		Sở hữu	Ghi*					Chuyển
D4		Ghi+						

Ta thấy D1 **sở hữu quyền chuyển** sang D4. Như vậy, D1 sẽ **có cơ hội** thực hiện quyền Ghi F2.

XÂY DỰNG, SỬ DỤNG MA TRẬN QUYỀN TRUY CẬP ĐỂ TRẢ LỜI CÁC CÂU HỎI VỀ PHÂN QUYỀN

Ví dụ trực quan

Hãy dựng bảng ma trận quyền truy cập cho một hệ thống có 3 miền D1, D2, D3, D4, biết rằng:

D1: <Máy in, {In}> <D4, {Chuyển}> <F1,{Ghi+}>

D2: <F1, {Ghi*}> <F3, {Ghi*}> <D4, {Chuyển}> <D3, {Chuyển}>

D3: <F2, {Đọc}> <F3, {Đọc}> <F2, {Sở hữu}> <D4, {Chuyển}>

a) Hãy cho biết tiến trình và người dùng thuộc miền D4 có những quyền gì, và có cơ hội thực hiện những quyền gì?
b) Hãy cho biết tiến trình và người dùng thuộc miền D2 có những quyền gì, và có cơ hội thực hiện những quyền gì?

Bài giải

Ta có ma trận quyền truy cập như sau:

	F1	F2	F3	Máy in	D1	D2	D3	D4
D1	Ghi+			In				Chuyển
D2	Ghi*		Ghi*				Chuyển	Chuyển
D3		Đọc	Đọc					Chuyển

		Sở hữu						
D4								

a) Tiến trình và người dùng thuộc miền D4:
- Có quyền: Không có quyền nào
- Có cơ hội thực hiện quyền: Ghi F1; Đọc, ghi và thực thi F2; Ghi F3

b) Tiến trình và người dùng thuộc miền D2:
- Có quyền: Ghi F1, Ghi F3, Chuyển D3 và Chuyển D4
- Có cơ hội thực hiện quyền: Ghi F1, Ghi F3, Chuyển D3, Chuyển D4; Đọc F3; Đọc, ghi và thực thi F2; Đọc F3

DẠNG BÀI SỐ 10
TÍNH TOÁN PHÂN MẢNH NỘI KHI SỬ DỤNG PHÂN TRANG

PHÂN MẢNH NỘI ĐỐI VỚI BÀI TOÁN CHO KÍCH THƯỚC TRANG

Đề bài

Tính toán hiện tượng phân mảnh nội xảy ra khi sử dụng hệ thống phân trang với kích thước trang là x bytes, cho 1 tiến trình cần X bytes. ?

Phương pháp giải

Tính số lượng trang bằng công thức: $pages = \frac{X}{x}$. Nếu $pages$:

+ Là số nguyên (X chia hết cho x): Cấp cho $pages$ trang đầy x bytes.
+ Bị lẻ:
    - Trước hết cần làm tròn lên (hàm trần – ceiling)
    - Sau đó: Cấp cho $(pages - 1)$ trang đầy x bytes và 1 trang còn lại $pages.x - X$ bytes

Ví dụ trực quan

Tính toán hiện tượng phân mảnh nội xảy ra khi sử dụng hệ thống phân trang với kích thước trang là 4096 bytes, cho 1 tiến trình cần 240445 bytes?

Bài làm

Có: $\frac{240445}{4096} \approx 58{,}7$ (trang)

Như vậy ta cần cấp phát 59 trang, trong đó:

- 58 trang đầy 4096 bytes
- 1 trang còn lại $59.4096 - 240445 = 1219$ (bytes)

PHÂN MẢNH NỘI ĐỐI VỚI BÀI TOÁN CHO SỐ LƯỢNG TIẾN TRÌNH

Đề bài

Tính toán hiện tượng phân mảnh nội xảy ra khi sử dụng hệ thống đang có p tiến trình cùng kích thước P bytes đang hoạt động biết rằng hệ thống sử dụng phân trang với kích thước trang là x?

Phương pháp giải

Tính số lượng trang bằng công thức: $pages = \frac{P}{x}$. Nếu $pages$:

+ Là số nguyên (P chia hết cho x): Cấp cho $pages$ trang đầy x bytes.
+ Bị lẻ:
  - Trước hết cần làm tròn lên (hàm trần – ceiling)
  - Sau đó: Cấp cho $(pages - 1)$ trang đầy x bytes và 1 trang còn lại $pages.x - P$ bytes

⇨ Nếu rơi vào trường hợp bị lẻ ➔ Tính lượng bộ nhớ bị lãng phí: $p.(pages.x - P)$ bytes

Ví dụ trực quan

Tính toán hiện tượng phân mảnh nội xảy ra khi sử dụng hệ thống đang có 120 tiến trình cùng kích thước 120312 bytes đang hoạt động; biết rằng hệ thống sử dụng phân trang với kích thước trang là 4096 bytes?

Bài làm

Có: $\frac{120312}{4096} \approx 29{,}37$ (trang)

Như vậy ta cần cấp phát 30 trang, trong đó:

- 29 trang đầy 4096 bytes
- 1 trang còn lại $29.4096 - 120312 = 2568$ (bytes)

⇨ Số bộ nhớ bị lãng phí là: $120.2568 = 308160$ (bytes)

DẠNG BÀI SỐ 11
HỆ THỐNG QUẢN LÝ BỘ NHỚ BẰNG PHƯƠNG PHÁP PHÂN ĐOẠN

Bài toán

Giả sử hệ thống quản lý bộ nhớ bằng phương pháp phân đoạn. Cho bảng phân đoạn sau:

Segment	Base	Length
0	b_0	l_0
1	b_1	l_1
...	...	...
n	b_n	l_n

Cho biết địa chỉ logic α, β sẽ được chuyển thành địa chỉ vật lý tương ứng nào và lỗi bảo vệ bộ nhớ có xảy ra không?

Phương pháp giải

Xét segment α. Giả sử segment này có Base là b_α và Length là l_α.

⇨ Địa chỉ vật lý tương ứng là: $Address = b_\alpha + \beta$

Để kiểm tra xem địa chỉ này có lỗi hay không, ta kiểm tra xem giá trị β có $\leq l_\alpha$ không.

Ví dụ

Giả sử hệ thống quản lý bộ nhớ bằng phương pháp phân đoạn. Cho bảng phân đoạn:

Segment	Base	Length
0	290	500
1	2100	40
2	90	100
3	1327	580
4	1952	98

Cho biết các địa chỉ logic sau sẽ được chuyển thành địa chỉ vật lý tương ứng nào và lỗi bảo vệ bộ nhớ có xảy ra không?

a) 2,78 b) 4,100 c) 3,420

Bài làm

a) 2,78: Địa chỉ vật lý tương ứng là: 90 + 78 = 168

⇨ Ta thấy 78 <= 100.

⇨ Địa chỉ hợp lệ

b) 4,100: Địa chỉ vật lý tương ứng là: 1952 + 100 = 2052

⇨ Ta thấy 100 > 98

⇨ Địa chỉ có lỗi

c) Làm tương tự.

DẠNG BÀI SỐ 12
HỆ THỐNG SỬ DỤNG KỸ THUẬT PHÂN TRANG

Bài toán

Xét một hệ thống sử dụng kỹ thuật phân trang, với bảng trang được lưu trữ trong bộ nhớ chính.

a) Nếu thời gian cho một lần truy xuất bình thường là t ns, thì mất bao nhiêu thời gian cho một thao tác bộ nhớ trong hệ thống này?
b) Nếu sử dụng TLBs với hit-ratio (tỷ lệ tìm thấy) là $x\%$, thời gian để tìm trong TLBs bằng y (ns). Tính thời gian truy xuất bộ nhớ trong hệ thống?

Phương pháp giải

- Thời gian cho một thao tác bộ nhớ bằng $2t$ (ns) *(Một thao tác = Truy cập + Xuất dữ liệu)*
- Thời gian truy xuất bộ nhớ trong hệ thống bằng:

$$(2 - x\%) * t + y$$

Ví dụ trực quan

Xét một hệ thống sử dụng kỹ thuật phân trang, với bảng trang được lưu trữ trong bộ nhớ chính.

a) Nếu thời gian cho một lần truy xuất bình thường là 150 ns, thì mất bao nhiêu thời gian cho một thao tác bộ nhớ trong hệ thống này?
b) Nếu sử dụng TLBs với hit-ratio (tỷ lệ tìm thấy) là 85%, thời gian để tìm trong TLBs gần như bằng 0. Tính thời gian truy xuất bộ nhớ trong hệ thống?

Bài giải

a) Thời gian cho một thao tác bộ nhớ là: $2 * 150 = 300$ ns
b) Thời gian truy xuất bộ nhớ trong hệ thống bằng: $(2 - 0.85) * 150 + 0 = 172{,}5$ ns