**ÔN TẬP HỆ ĐIỀU HÀNH**

**Câu 1:** Xét tập các tiến trình sau (với thời gian yêu cầu CPU và độ ưu tiên kèm theo)

|  |  |  |  |
| --- | --- | --- | --- |
| Tiến trình | Thời điểm vào RL | Thời gian CPU | Độ ưu tiên |
| P1 | 3.5 | 10 | 2 |
| P2 | 1.5 | 3 | 3 |
| P3 | 2 | 7 | 1 |
| P4 | 1.5 | 9 | 2 |
| P5 | 2.5 | 5 | 4 |

Vẽ sơ đồ Gautt, tính thời gian đợi của các process và thời gian đợi trung bình trong từng giải thuật định thời?

1. FCFS:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P2 | P4 | P3 | P5 | P1 |

0 1,5 4,5 13,5 20,5 25,5 35,5

Thời gian chờ:

-P2 = 0ms

-P4 = 4,5 – 1,5 = 3ms

-P3 = 13,5 – 2 = 11,5ms

-P5 = 20,5 – 2,5 = 18ms

-P1 = 25,5 – 3,5 = 22 ms

Thời gian chờ trung bình= (0+3+11,5+18+22)/5=10,9ms

1. SJF độc quyền

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P2 | P5 | P3 | P4 | P1 |

0 1,5 4,5 9,5 16,5 25,5 35,5

Thời gian chờ:

-P1 = 25,5 – 3,5 = 22 ms

-P2 = 0ms

-P4 = 16,5 – 1,5 = 3ms

-P3 = 9,5 – 2 = 7,5ms

-P5 = 4,5 – 2,5 = 2ms

Thời gian chờ trung bình= (0+22+3+7,5+2)/5=6,9ms

1. SJF không độc quyền

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P2 | P5 | P3 | P4 | P1 |

0 1,5 4,5 9,5 16,5 25,5 35,5

Thời gian chờ:

-P1 = 25,5 – 3,5 = 22 ms

-P2 = 0ms

-P4 = 16,5 – 1,5 = 3ms

-P3 = 9,5 – 2 = 7,5ms

-P5 = 4,5 – 2,5 = 2ms

Thời gian chờ trung bình= (0+22+3+7,5+2)/5=6,9ms

1. RR (quantumn = 3)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P2 | P4 | P3 | P5 | P1 | P4 | P3 | P5 | P1 | P4 | P3 | P1 | P1 |

0 1,5 4,5 7,5 10,5 13,5 16,5 19,5 22,5 24,5 27,5 30,5 31,5 34,5 35,5

Thời gian chờ:

-P1 = (13,5 – 3,5)+(24,5-16,5) +(31,5-27,5) = 22 ms

-P2 = 0ms

-P4 = (4,5 – 1,5) + (16,5 - 7,5)+(27,5 - 19,5) = 20ms

-P3 = (7,5 – 2)+(19,5 - 10) + (27,5 – 19,5)= 20ms

-P5 = (10,5 – 2,5) + (22,5 – 13,5) = 17ms

Thời gian chờ trung bình= (0+22+20+20+17)/5=15,8ms

1. Priority độc quyền (càng nhỏ càng ưu tiên, cùng độ ưu tiên xét Pi và Pj, Pi ưu tiên hơn nếu i < j).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P4 | P3 | P1 | P2 | P5 |

0 1,5 10,5 17,5 27,5 30,5 35,5

Thời gian chờ:

-P1 = 17,5 – 3,5= 14 ms

-P2 = 27,5 – 1,5 = 26ms

-P4 = 0ms

-P3 = 10,5 – 2= 8,5ms

-P5 = 30,5 – 2,5= 28ms

Thời gian chờ trung bình= (0+14+26+8,5+28)/5=15,3ms

1. Priority không độc quyền

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | P4 | P3 | P1 | P4 | P2 | P5 |

0 1,5 2 9 19 26,5 29,5 35,5

Thời gian chờ:

-P1 = 19 – 3,5= 15,5 ms

-P2 = 27,5 – 1,5 = 25ms

-P4 = 17ms

-P3 = 0ms

-P5 = 29,5 – 2,5= 26,5ms

Thời gian chờ trung bình= (0+14+26+7+28)/5=16,8ms

**Câu 2:** Xét 1 tập các process sau có thời gian thực thi CPU tính bằng miligiây:

|  |  |  |  |
| --- | --- | --- | --- |
| Tiến trình | Thời điểm vào RL | Thời gian CPU | Độ ưu tiên |
| P1 | 2 | 4.5 | 2 |
| P2 | 4 | 3 | 3 |
| P3 | 5 | 11 | 1 |
| P4 | 8 | 5 | 2 |
| P5 | 3 | 9 | 4 |

Giả sử thứ tự đến để thực thi của các process là P1, P2, P3, P4, P5. (thứ tự vào RL của các tiến trình lần lượt là: 2, 4, 5, 8, 3).

Vẽ sơ đồ Gautt, tính thời gian đợi của các process và thời gian đợi trung bình trong từng giải thuật định thời?

1. FCFS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P1 | P5 | P2 | P3 | P4 |

0 2 6,5 15,5 18,5 29,5 34,5

Thời gian chờ:

-P1 = 0ms

-P2 = 15,5 – 4 = 11,5

-P3= 18,5 – 5 =13,5

-P4= 29,5 – 8 =21,5

-P5= 6,5 – 3 = 3,5

Thời gian chờ trung bình= (0+11,5+13,5+21,5+3,5)/5=9,8ms

1. SJF độc quyền

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P4 | P5 | P3 |

0 2 6,5 9,5 14,5 23,5 34,5

Thời gian chờ:

-P1 = 0ms

-P2 = 6,5 – 4 = 2,5

-P3= 24,5 – 5 =19,5

-P4= 9,5 – 8 =1,5

-P5= 15,5 – 3 = 12,5

Thời gian chờ trung bình= (0+2,5+18,5+1,5+11,5)/5=6,8ms

1. SJF không độc quyền

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P4 | P5 | P3 |

0 2 6,5 9,5 14,5 23,5 34,5

Thời gian chờ:

-P1 = 0ms

-P2 = 6,5 – 4 = 2,5

-P3= 23,5 – 5 =18,5

-P4= 9,5 – 8 =1,5

-P5= 14,5 – 3 = 11,5

Thời gian chờ trung bình= (0+2,5+18,5+1,5+11,5)/5=6,8ms

1. RR (quantumn = 2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | P5 | P2 | P1 | P3 | P5 | P4 | P2 | P1 |

0 2 4 6 8 10 12 14 16 17 17,5

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P3 | P4 | P5 | P3 | P4 | P5 | P3 | P5 | P3 | P3 |

17,5 19,5 21,5 22,5 24,5 25,5 28,5 30,5 31,5 33,5 34,5

Thời gian chờ:

-P1 =0 + (8-4)+(17 – 10) =11ms

-P2 = (6 – 4) + (16 – 8) = 10ms

-P3= (10 – 5) + (17,5 – 12) + (22,5 – 19,5) +(28,5 – 24,5)+(31,5 – 30,5) = 18,5ms

-P4= (12 – 8) + (19,5 - 14) + (24,5 – 21,5) =12,5ms

-P5= (4 – 3) + (14 - 6) + (21,5 - 16) + (25,5 – 22,5) + (30,5 – 28,5) = 19,5ms

Thời gian chờ trung bình= (11+10+18,5+12,5+19,5)/5=14,3ms

1. Priority độc quyền (càng nhỏ càng ưu tiên, cùng độ ưu tiên xét Pi và Pj, Pi ưu tiên hơn nếu i < j).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P1 | P3 | P4 | P2 | P5 |

0 2 6,5 17,5 22,5 25,5 34,5

Thời gian chờ:

-P1 = 0ms

-P2 = 22,5 – 4 = 18,5

-P3= 6,5 – 5 =1,5

-P4= 17,5 – 8 =9,5

-P5= 25,5 – 3 = 22,5

Thời gian chờ trung bình= (0+18,5+1,5+9,5+22,5)/5=10,4ms

1. Priority không độc quyền

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | P1 | P3 | P1 | P4 | P2 | P5 |

0 2 5 16 17,5 22,5 25,5 34,5

Thời gian chờ:

-P1 = 11ms

-P2 = 22,5 – 4 = 18,5ms

-P3= 0ms

-P4= 17,5 – 8 =9,5

-P5= 25,5 – 3 = 22,5

Thời gian chờ trung bình= (0+18,5+11+9,5+22,5)/5=12,3ms

**Câu 3:** Xét giải pháp đồng bộ Peterson sửa đổi như sau:

|  |  |
| --- | --- |
| while(TRUE)  {  flag[0]=TRUE;  **turn= 0;**  while (turn==1 && flag[1] );  Critical section  flag[0]= FALSE;  Non-critical section  } | while(TRUE)  {  flag[1]=TRUE;  **turn= 1;**  while (turn==0 && flag[0] );  Critical section  flag[1]= FALSE;  Non-critical section  } |

Đây có phải là giải pháp đảm bảo độc quyền truy xuất không?

* Không phải, Vì khi flag[0] và flag[1] đều bằng true , thì đoạn nào có turn = 1 thì ms được thực hiện, nhưng khi đã gán turn =0,1 nhưng đk hàm while lại xét ngược lại nên k bao h critical section đc thực hiện => k có tiến trình vào miền găng, đảm bảo độc quyền truy xuất.

**Câu 4:** Một biến X được chia sẻ bởi hai tiến trình cùng thực hiện đoạn code sau:

do{

X = X+1;

if (X==20) X = 0;

}while (TRUE);

Bắt đầu với giá trị X=0, chứng tỏ rằng X có thể vượt quá 20. Hãy sửa đoạn chương trình trên để X không vượt quá 20

* Do X được chia sẻ chung ở cả hai tiến trình và chỉ bị reset khi X == 20 nên X có thể vượt quá 20 khi:
* Có một lý do trong máy đột ngột khiến tiến trình 2 dừng lại. Sau đó tại tiến trình 1, khi X = 19 thì tiến trình 2 được release đúng ngay đoạn X = X + 1, và cộng dồn với X = 20 sau lệnh X = X + 1 của tiến trình 1 dẫn tới X vượt quá 20 và còn bị cộng tới vô cùng.

Semaphore a = 1;

Do{

Down(a);

X=X+1;

If(x==20) x= 0;

Up(a);

}while(true)

- Để giải quyết vấn đề này ( làm cho X luôn nhỏ hơn 20 ) thì ta có thẻ giải quyết bằng cách đưa 2 tiến trình về làm 1, tức là làm cho cùng một lúc chỉ có một trong hai tiến trình được chạy bằng một biến Semaphore khởi tạo bằng 1.

**Câu 5:** Xét hai tiến trình xử lý đoạn chương trình sau:

process P1 { A1(); A2(); }

process P2 { B1(); B2(); }

Đồng bộ hóa sự hoạt động của hai tiến trình này sao cho cả A1 và B1 đều hoàn tất trước khi A2 hay B2 bắt đầu

Semaphore\_1 = 0;

Semaphore\_2 = 0;

Process p1{

A1;

Wait(Semaphore\_1);

Signal(Semaphore\_2);

A2; }

Process p2{

B1;

Wait(Semaphore\_2);

Signal(Semaphore\_1);

B2; }

**Câu 6:** Cho các tiến trình xử lý đoạn chương trình sau

process P1 { for(i=1; i<=100; i++) Ai(); }

process P2 { for(j=1; j<=100; j++) Bj(); }

Đồng bộ hóa hoạt động của hai tiến trình này sao cho với mọi 2 ≤ k ≤ 100, Ak chỉ có thể bắt đầu khi Bk-1 đã kết thúc và Bk chỉ có thể bắt đầu khi Ak-1 kết thúc

Semaphore\_1 = 1;

Semaphore\_2 = 1;

Process A:

For (int i = 1; i< =100;i++){

Wait(Semaphore\_1);

Ai;

Signal(Semaphore\_2)

}

Process B:

For (int i = 1; i< =100;i++){

Wait(Semaphore\_2);

Bi;

Signal(Semaphore\_1)

}

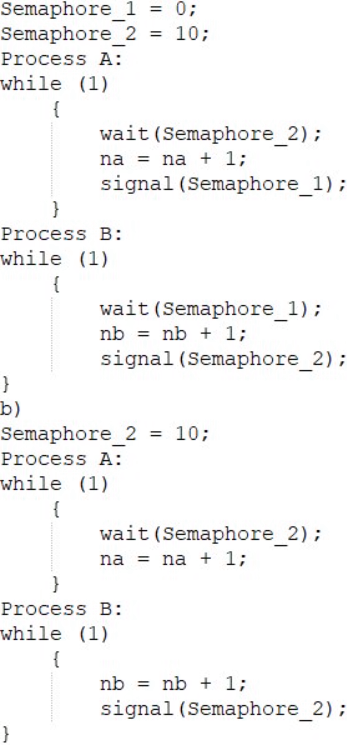
**Câu 7:** Xét hai tiến trình sau:

|  |  |
| --- | --- |
| process A {  while(TRUE)  na = na+1;  } | process B {  while(TRUE)  nb = nb+1;  } |

na và nb khởi tạo là 1

1. Đồng bộ hóa hai tiến trình trên dùng semaphore sao cho tại bất cứ thời điểm nào đều có na ≤ nb + 10
2. Thực hiện câu a với điều kiện nb ≤ na ≤ nb + 10

Chúng ta phải làm 2 việc với 2 semaphore đã cho: đảm bảo na >= nb và na <= nb + 10, như vậy nghĩa là khi na = nb thì tiến trình B bị block cho tới khi A tiến hành được ít nhất một lần, cũng như khi na = nb + 10 thì A bị block cho tới khi B tiến hành được ít nhất một lần.



**Câu 8:** Xét trạng thái hệ thống:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Max | | | Allocation | | | Available | | |
| R0 | R1 | R2 | R0 | R1 | R2 | R0 | R1 | R2 |
| P0 | 3 | 3 | 2 | 2 | 2 | 1 | 2 | 1 | 3 |
| P1 | 6 | 2 | 3 | 2 | 0 | 2 |  |  |  |
| P2 | 4 | 2 | 5 | 3 | 1 | 1 |  |  |  |
| P3 | 7 | 3 | 3 | 3 | 0 | 2 |  |  |  |

1. Cho biết nội dung bảng Need.

|  |  |  |
| --- | --- | --- |
| Need | | |
| R0 | R1 | R2 |
| 1 | 1 | 1 |
| 4 | 2 | 1 |
| 1 | 2 | 4 |
| 4 | 3 | 1 |

1. Hệ thống có ở trong trạng thái an toàn không?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | Allocation | | | Available | | |
| R0 | R1 | R2 | R0 | R1 | R2 | R0 | R1 | R2 |
| P0 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 1 | 3 |
| P1 | 4 | 2 | 1 | 2 | 0 | 2 |  |  |  |
| P2 | 1 | 2 | 4 | 3 | 1 | 1 |  |  |  |
| P3 | 7 | 3 | 1 | 3 | 0 | 2 |  |  |  |

* Chọn P0 để cấp phát và P0 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | Allocation | | | Available | | |
| R0 | R1 | R2 | R0 | R1 | R2 | R0 | R1 | R2 |
| P0 |  |  |  |  |  |  | 4 | 3 | 4 |
| P1 | 4 | 2 | 1 | 2 | 0 | 2 |  |  |  |
| P2 | 1 | 2 | 4 | 3 | 1 | 1 |  |  |  |
| P3 | 7 | 3 | 1 | 3 | 0 | 2 |  |  |  |

* Chọn P1 để cấp phát và P1 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | Allocation | | | Available | | |
| R0 | R1 | R2 | R0 | R1 | R2 | R0 | R1 | R2 |
| P0 |  |  |  |  |  |  | 6 | 3 | 6 |
| P1 |  |  |  |  |  |  |  |  |  |
| P2 | 1 | 2 | 4 | 3 | 1 | 1 |  |  |  |
| P3 | 7 | 3 | 1 | 3 | 0 | 2 |  |  |  |

* Chọn P2 để cấp phát và P2 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | Allocation | | | Available | | |
| R0 | R1 | R2 | R0 | R1 | R2 | R0 | R1 | R2 |
| P0 |  |  |  |  |  |  | 9 | 4 | 7 |
| P1 |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  |  |  |  |
| P3 | 7 | 3 | 1 | 3 | 0 | 2 |  |  |  |

* Chọn P3 để cấp phát và P3 hoàn tất.
* Chuỗi cấp phát : P0,P1,P2,P3.Hệ thống an toàn.

1. Nếu tiến trình P1 yêu cầu 3 cho R0, 2 cho R1, 2 cho R2. Hãy xác định xem sau khi thực hiện yêu cầu này thì hệ thống có an toàn không?

Request <= need

(3,2,2) <= (4,2,1) sai.

* lỗi process vượt quá yêu cầu tối đa, hệ thống không an toàn.

**Câu 9:** Xét trạng thái hệ thống:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Max | | | | Allocation | | | | Available | | | |
| A | B | C | D | A | B | C | D | A | B | C | D |
| P0 | 2 | 1 | 1 | 2 | 0 | 0 | 1 | 2 | 2 | 4 | 1 | 1 |
| P1 | 1 | 5 | 7 | 0 | 1 | 0 | 0 | 0 |  |  |  |  |
| P2 | 2 | 3 | 5 | 6 | 1 | 3 | 4 | 4 |  |  |  |  |
| P3 | 2 | 6 | 4 | 2 | 0 | 5 | 3 | 2 |  |  |  |  |
| P4 | 2 | 6 | 5 | 6 | 0 | 1 | 1 | 4 |  |  |  |  |

1. Cho biết nội dung bảng Need.

|  |  |  |  |
| --- | --- | --- | --- |
| Need | | | |
| A | B | C | D |
| 2 | 1 | 0 | 0 |
| 0 | 5 | 7 | 0 |
| 1 | 0 | 1 | 2 |
| 2 | 1 | 1 | 0 |
| 2 | 5 | 4 | 2 |

1. Hệ thống có ở trong trạng thái an toàn không?

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | | Allocation | | | | Available | | | |
| A | B | C | D | A | B | C | D | A | B | C | D |
| P0 | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 4 | 1 | 1 |
| P1 | 0 | 5 | 7 | 0 | 1 | 0 | 0 | 0 |  |  |  |  |
| P2 | 1 | 0 | 1 | 2 | 1 | 3 | 4 | 4 |  |  |  |  |
| P3 | 2 | 1 | 1 | 0 | 0 | 5 | 3 | 2 |  |  |  |  |
| P4 | 2 | 5 | 4 | 2 | 0 | 1 | 1 | 4 |  |  |  |  |

=> chọn P0 để cấp phát và P0 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | | Allocation | | | | Available | | | |
| A | B | C | D | A | B | C | D | A | B | C | D |
| P0 |  |  |  |  |  |  |  |  | 2 | 4 | 2 | 3 |
| P1 | 0 | 5 | 7 | 0 | 1 | 0 | 0 | 0 |  |  |  |  |
| P2 | 1 | 0 | 1 | 2 | 1 | 3 | 4 | 4 |  |  |  |  |
| P3 | 2 | 1 | 1 | 0 | 0 | 5 | 3 | 2 |  |  |  |  |
| P4 | 2 | 5 | 4 | 2 | 0 | 1 | 1 | 4 |  |  |  |  |

=> chọn P2 để cấp phát và P2 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | | Allocation | | | | Available | | | |
| A | B | C | D | A | B | C | D | A | B | C | D |
| P0 |  |  |  |  |  |  |  |  | 3 | 7 | 6 | 7 |
| P1 | 0 | 5 | 7 | 0 | 1 | 0 | 0 | 0 |  |  |  |  |
| P2 |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 | 2 | 1 | 1 | 0 | 0 | 5 | 3 | 2 |  |  |  |  |
| P4 | 2 | 5 | 4 | 2 | 0 | 1 | 1 | 4 |  |  |  |  |

=> chọn P3 để cấp phát và P3 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | | Allocation | | | | Available | | | |
| A | B | C | D | A | B | C | D | A | B | C | D |
| P0 |  |  |  |  |  |  |  |  | 3 | 12 | 9 | 9 |
| P1 | 0 | 5 | 7 | 0 | 1 | 0 | 0 | 0 |  |  |  |  |
| P2 |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |  |  |  |  |  |  |
| P4 | 2 | 5 | 4 | 2 | 0 | 1 | 1 | 4 |  |  |  |  |

=> chọn P1 để cấp phát và P1 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | | Allocation | | | | Available | | | |
| A | B | C | D | A | B | C | D | A | B | C | D |
| P0 |  |  |  |  |  |  |  |  | 4 | 12 | 9 | 9 |
| P1 |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |  |  |  |  |  |  |
| P4 | 2 | 5 | 4 | 2 | 0 | 1 | 1 | 4 |  |  |  |  |

=> chọn P4 để cấp phát và P4 hoàn tất.

=> Chuỗi cấp phát : P0, P2,P3, P1,P4.Hệ thống an toàn

c) Nếu P1 có yêu cầu tài nguyên (3, 3, 2, 1) thì yêu cầu này có được đáp ứng không?

Request <= need

(3, 3, 2, 1) <= (0,5,7,0) sai

* lỗi tiến trình vượt quá yêu cầu tối đa => không được đáp ứng.

**Câu 10:** Cho 1 hệ thống có 4 tiến trình, P1 đến P4, và 3 loại tài nguyên, R1 (3 thực thể), R2 (4 thực thể), R3 (2 thực thể). Tiến trình P1 giữ 2 R1, và yêu cầu 2 R2. Tiến trình P2 giữ 3 R2 và yêu cầu 1 R1 và 2 R3. P3 giữ 1 R1, 1 R2 và yêu cầu 1 R3. P4 giữ 2 R3 và yêu cầu 1 R1.

1. Vẽ đồ thị tài nguyên cho hệ thống này.
2. Có nguy cơ deadlock không?
3. Có chuỗi an toàn không? Vì sao?

R2

a.

R3

R1

P2

P3

P4

P1(0,5,7,0)

b. Vì P1 chờ P2, P2 chờ P3, P3 chờ P4 và P4 chờ P1 nên có nguy cơ Deadlock.

c. không có chuỗi an toàn, vì cả 4 tiến trình đều phải chờ, từ đó tạo nên chu trình không thể giải phóng bất kỳ tiến trình nào dẫn đến deadlock nên không có chuỗi an toàn.

**Câu 11:** Cho hệ thống 5 tiến trình và 3 loại tài nguyên (A, B, C). Giả sử hệ thống đang ở trạng thái như sau:

Sử dụng giải thuật Banker.

1. Tính nhu cầu còn lại của mỗi tiến trình và số tài nguyên của mỗi loại hệ thống.

|  |  |  |
| --- | --- | --- |
| Need | | |
| A | B | C |
| 3 | 6 | 3 |
| 4 | 7 | 4 |
| 2 | 2 | 2 |
| 3 | 5 | 3 |
| 2 | 3 | 1 |

1. Hãy tìm 1 trạng thái an toàn.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | Allowcation | | | available | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 | 3 | 6 | 3 | 1 | 1 | 2 | 2 | 6 | 3 |
| P2 | 4 | 7 | 4 | 2 | 2 | 1 |
| P3 | 2 | 2 | 2 | 1 | 0 | 0 |
| P4 | 2 | 5 | 3 | 0 | 1 | 1 |
| P5 | 2 | 3 | 2 | 0 | 0 | 1 |

* Chọn P3 để cấp phát, P3 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | Allowcation | | | available | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 | 3 | 6 | 3 | 1 | 1 | 2 | 3 | 6 | 3 |
| P2 | 4 | 7 | 4 | 2 | 2 | 1 |
| P3 |  |  |  |  |  |  |
| P4 | 2 | 5 | 3 | 0 | 1 | 1 |
| P5 | 2 | 3 | 2 | 0 | 0 | 1 |

* Chọn P1 để cấp phát, P1 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | Allowcation | | | available | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 |  |  |  |  |  |  | 4 | 7 | 5 |
| P2 | 4 | 7 | 4 | 2 | 2 | 1 |
| P3 |  |  |  |  |  |  |
| P4 | 2 | 5 | 3 | 0 | 1 | 1 |
| P5 | 2 | 3 | 2 | 0 | 0 | 1 |

* Chọn P2 để cấp phát, P2 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | Allowcation | | | available | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 |  |  |  |  |  |  | 6 | 9 | 5 |
| P2 |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |
| P4 | 2 | 5 | 3 | 0 | 1 | 1 |
| P5 | 2 | 3 | 2 | 0 | 0 | 1 |

* Chọn P4 để cấp phát, P4 hoàn tất.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Need | | | Allowcation | | | available | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 |  |  |  |  |  |  | 6 | 10 | 6 |
| P2 |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |
| P4 |  |  |  |  |  |  |
| P5 | 2 | 3 | 2 | 0 | 0 | 1 |

* Chọn P5 để cấp phát, P5 hoàn tất.
* Vậy chuỗi cấp phát P3,P1,P2,P4,P5. Hệ thống an toàn.

1. Nếu tiến trình P2 có yêu cầu thêm tài nguyên (A: 2; B:3; C: 2), hãy cho biết yêu cầu này có thể đáp ứng mà bảo đảm không xảy ra tình trạng deadlock hay không?

Request <= need

(2,3,2) <= (4,7,4) sai

* lỗi tiến trình vượt quá yêu cầu tối đa => không được đáp ứng, có nguy cơ xảy ra deadlock.

**Câu 12:** Trong mô hình cấp phát bộ nhớ liên tục, có bốn phân mảnh bộ nhớ theo thứ tự với kích thước là 2500KB, 500KB, 300KB, 750KB. Giả sử có 4 tiến trình đang chờ cấp phát bộ nhớ theo thứ tự P1, P2, P3, P4. Kích thước tương ứng của các tiến trình trên là: 112KB, 515KB, 225KB, 426KB. Hãy cấp phát bộ nhớ cho các tiến trình trên theo các thuật toán:

1. First-fit
2. Best-fit
3. Worst-fit
4. Next-fit.

a.

0

2500

3000

3300

4050

|  |
| --- |
| P1  P2  P3  P4 – Còn 1222 |
|  |
|  |
|  |

b.

|  |
| --- |
|  |
| P4 – Còn 74KB |
| P1 – Còn 188KB |
| P2  P3 – Còn 10KB |

c.

0

2500

3000

3300

4050

0

2500

3000

3300

4050

|  |
| --- |
| P1  P2  P3  P4 – Còn 1222 |
|  |
|  |
|  |

d.

0

2500

3000

3300

4050

|  |
| --- |
| P1  P3 – Còn 2153 KB |
| P4 – Còn 74KB |
|  |
| P2 – Còn 235KB |

Trong các thuật toán trên, thuật toán nào cho phép sử dụng bộ nhớ hiệu quả nhất? Vì sao?

**Câu 13:** Một biến *x* được chia sẻ bởi hai tiến trình cùng thực hiện đoạn code sau:

*do*

*x = x +1;*

*if (x == 25) x = 0;*

*while (TRUE);*

Bắt đầu với giá trị x = 0, chứng tỏ rằng giá trị x có thể vượt quá 30. Cần sửa chữa đoạn chương trình trên như thế nào để bảo đảm x không vượt quá 30?

**Câu 14:** Cho bảng trang của tiến trình P1. Biết rằng kích thước mỗi frame là 1KB.

P f

1. Địa chỉ ảo 1045 sẽ được chuyển thành địa chỉ vật lý bao nhiêu?

a=1045

p = a div 1024 = 1

d = a mod 1024 = 21

với p = 1, ta có f = 4 => A = f\*PS + d = 4\*1024 +21= 4117

* Địa chỉ vật lý là 4117.

1. Địa chỉ vật lý 8076 sẽ được chuyển thành địa chỉ ảo bao nhiêu?

A = 8076

f = A div 1024 = 7

d = A mod 1024 = 908

với f = 7, ta có p = 3 =>a = p\*PS+d = 3\*1024+908=3980.

=> Địa chỉ ảo là 3980.

**Câu 15:** Xét bảng phân đoạn sau:

|  |  |  |
| --- | --- | --- |
| Segment | Base | Length |
| 0 | 121 | 300 |
| 1 | 1320 | 24 |
| 2 | 1323 | 300 |
| 3 | 2234 | 680 |
| 4 | 967 | 196 |

Cho biết địa chỉ vật lý tương ứng với các địa chỉ logic sau đây:

1. 0,530 [s,d]

Ta có s = 0, d =530 => b=121, l=300

D > l = 530 > 300 đúng.

=>lỗi địa chỉ vượt quá kích thước segment

1. 1,200

Ta có s = 1, d =200 => b=1320, l=24

D > l = 200 > 24 đúng.

=>lỗi địa chỉ vượt quá kích thước segment

1. 3,800

Ta có s = 3, d =800 => b=2234, l=680

D > l = 800 > 680 đúng.

=>lỗi địa chỉ vượt quá kích thước segment

1. 2,400

Ta có s = 2, d =400 => b=1323, l=300

D > l = 400 > 300 đúng.

=>lỗi địa chỉ vượt quá kích thước segment

1. 4,120

Ta có s = 4, d =120 => b=967, l=196

D > l = 120 > 196 sai.

Ta có: A = b + d = 967 + 120 = 1087

**Câu 16:** Giả sử một chuỗi truy xuất bộ nhớ có chiều dài là p với n số hiệu trang khác nhau xuất hiện trong chuỗi. Giả sử hệ thống sử dụng m khung trang (khởi động trống), n>m. Hãy cho biết:

* 1. Số lượng tối thiểu các lỗi trang xảy ra.
  2. Số lượng tối đa các lỗi trang xảy ra.

**Câu 17:** Giả sử có một hệ thống sử dụng kỹ thuật phân trang theo yêu cầu. Bảng trang được lưu trữ trong các thanh ghi. Để xử lý một lỗi trang tốn 8 miligiây nếu có sẵn một khung trang trống, và tốn 20 miligiây nếu cần thay thế trang. Giả sử xác suất cần thay thế trang khi có lỗi trang là 70%. Mỗi truy xuất bộ nhớ tốn 100 nanogiây. Giả sử tỷ lệ phát sinh lỗi trang là 1% thì thời gian truy xuất bộ nhớ trung bình là bao nhiêu?

Do bảng trang được lưu trữ trong các thanh ghi => thời gian truy xuất vào bàng trang không đáng kể. => Thời gian truy xuất bộ nhớ: 100 nanoseconds – Thời gian trung bình để xử lý lỗi trang: 0,3 \* 8 + 0,7 \* 20 = 16.4 miliseconds – Theo đề: EAT <= 200 nanoseconds => EAT = 100 + r\*16.4 <= 200 => r <= (200-100)/16,4 = 0,609%

**Câu 18:** Xét chuỗi truy xuất bộ nhớ sau

9, 6, 3 4, 2, 7, 5, 6, 2, 7, 2, 9, 5, 7, 5, 2

Minh họa quá trình thay thế trang với các thuật toán thay thế sau với số khung trang bộ nhớ lần lượt là 4 và 5:

* 1. FIFO

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 6 | 3 | 4 | 2 | 7 | 5 | 6 | 2 | 7 | 2 | 9 | 5 | 7 | 5 | 2 |
| \* | \* | \* | \* | \* | \* | \* | \* |  |  |  | \* |  |  |  | \* |
| 9 | 9 | 9 | 9 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 9 | 9 | 9 | 9 | 9 |
|  | 6 | 6 | 6 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 2 |
|  |  | 3 | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
|  |  |  | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |

* Có 10 lỗi trang

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 6 | 3 | 4 | 2 | 7 | 5 | 6 | 2 | 7 | 2 | 9 | 5 | 7 | 5 | 2 |
| \* | \* | \* | \* | \* | \* | \* | \* |  |  |  | \* |  |  |  |  |
| 9 | 9 | 9 | 9 | 9 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
|  | 6 | 6 | 6 | 6 | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
|  |  | 3 | 3 | 3 | 3 | 3 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 9 | 9 | 9 | 9 | 9 |
|  |  |  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

* Có 9 lỗi trang

b) OPT : lâu sử dụng nhất trong tương lai

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 6 | 3 | 4 | 2 | 7 | 5 | 6 | 2 | 7 | 2 | 9 | 5 | 7 | 5 | 2 |
| \* | \* | \* | \* | \* | \* | \* |  |  |  |  | \* |  |  |  |  |
| 9 | 9 | 9 | 9 | 9 | 9 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
|  | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 9 | 9 | 9 | 9 | 9 |
|  |  | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|  |  |  | 4 | 4 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |

* Có 8 lỗi trang.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 6 | 3 | 4 | 2 | 7 | 5 | 6 | 2 | 7 | 2 | 9 | 5 | 7 | 5 | 2 |
| \* | \* | \* | \* | \* | \* | \* |  |  |  |  |  |  |  |  |  |
| 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
|  | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  | 3 | 3 | 3 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
|  |  |  | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
|  |  |  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

* Có 7 lỗi trang
  1. LRU: lâu nhất chưa sử dụng trong quá khứ.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 6 | 3 | 4 | 2 | 7 | 5 | 6 | 2 | 7 | 2 | 9 | 5 | 7 | 5 | 2 |
| \* | \* | \* | \* | \* | \* | \* | \* |  |  |  | \* | \* |  |  |  |
| 9 | 9 | 9 | 9 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|  | 6 | 6 | 6 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
|  |  | 3 | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 9 | 9 | 9 | 9 | 9 |
|  |  |  | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 6 | 5 | 5 | 5 | 5 |

* Có 10 lỗi trang

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 6 | 3 | 4 | 2 | 7 | 5 | 6 | 2 | 7 | 2 | 9 | 5 | 7 | 5 | 2 |
| \* | \* | \* | \* | \* | \* | \* | \* |  |  |  | \* |  |  |  |  |
| 9 | 9 | 9 | 9 | 9 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
|  | 6 | 6 | 6 | 6 | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
|  |  | 3 | 3 | 3 | 3 | 3 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 9 | 9 | 9 | 9 | 9 |
|  |  |  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

* Có 9 lỗi trang

**Câu 19:** Cho chương trình sau:

int A[100][100];

for (i=0; i<100; i++)

for (j=0; j<100; j++) A[i][j]= 0;

Trong đó mảng A thuộc đoạn data và vòng lặp for thuộc đoạn code của chương trình. Đoạn code có kích thước là 200 byte, còn đoạn data có kích thước là 100x100x2 = 20.000 byte. Hai biến i, j là inline trong thanh ghi CPU.

Bộ nhớ máy tính được tổ chức theo mô hình phân trang với kích thước trang là 200 và có 3 khung trang. Đoạn code đã được nạp sẵn vào khung trang thứ nhất, 2 khung trang còn lại trống. Đoạn data đang ở trong bộ nhớ ảo.

Với thuật toán thay thế trang là LRU, hãy tính số lỗi trang phát sinh khi chạy chương trình trên.

**Câu 20:** Một ổ đĩa C: được định dạng dưới dạng FAT16 gồm có 15 cluster. Kích thước của mỗi cluster là 512 byte, giả sử có bảng FAT sau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | -1 | 0 | 5 | 6 | 8 | 7 | -1 | -1 | -1 | 14 | -1 | 0 | 10 | -1 |

Thư mục gốc bắt đầu tại cluster 0, tại cluster 0 và cluster 9 xem được các entry như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filename | Ext | attrib | Start cluster | size |
| Hdh | Doc |  | 11 | 400 |
| HinhAnh |  | D | 9 |  |
| pascal | Doc |  | 4 | 1200 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filename | Ext | attrib | Start cluster | size |
| Hoguom | Jpg |  | 3 | 1200 |
| Halong | Jpg |  | 13 | 1100 |

Hãy vẽ cây thư mục và cho biết các số liệu cluster của từng file và thư mục.

Fat:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | -1 | 0 | 5 | 6 | 8 | 7 | -1 | -1 | -1 | 14 | -1 | 0 | 10 | -1 |

Cluster:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| R |  |  | G1 | P1 | G2 | P2 | P3 | G3 | I | H2 | H |  | H1 | H3 |

R: root, I: HinhAnh, H:hdh,P:pascal,G:Hoguom,L:Halong

/C

HinhAnh

Hdh.doc

Pascal.doc

Halong.jpg

Hoguom.jpg

**Câu 21**: Một ổ đĩa có 17 cluster, kích thước của mỗi cluster là 1024 byte. Giả sử 17 phần tử đầu của bảng FAT có giá trị cho ở bảng sau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 1 | 2 | -1 | 0 | -1 | 0 | 13 | 8 | 9 | -1 | 0 | 12 | -1 | 14 | 16 | 0 | 4 |

Và 3 entry đầu của Root Dir có giá trị sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filename | Ext | attrib | Start cluster | size |
| Music |  | D | 11 |  |
| Autoexec | bat |  | 6 | 5032 |
| Vidu | txt | R | 7 | 3018 |

1. Cho biết các cluster dữ liệu của thư mục music, tập tin autoxec.bat và vidu.txt

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|  |  |  |  | A5 |  | A1 | R1 | R2 | R3 |  | M1 | M2 | A2 | A3 |  | A4 |

Music: M, Autoxec: A, Vidu: V.

1. Cho biết nội dung 17 phần tử đầu bảng FAT và 3 entry đầu của Root dir nếu thay tập tin autoexec.bat bằng tập tin boot.ini có kích thước 7318 byte.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 1 | 2 | -1 | 0 | 5 | 15 | 13 | 8 | 9 | -1 | 0 | 12 | -1 | 14 | 16 | -1 | 4 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filename | Ext | attrib | Start cluster | size |
| Music |  | D | 11 |  |
| boot | ini |  | 6 | 7318 |
| Vidu | txt | R | 7 | 3018 |

**Câu 22:** Một ổ đĩa C: được được định dạng dưới dạng FAT 16 gồm có 15 cluster. Kích thước của mỗi cluster là 512 byte. Giả sử có cây thư mục sau (trong ngoặc là kích thước mớc file):

\

Amnhac

Một entry trong bảng thư mục chiếm 32 byte. Hãy lập 1 phương án lưu trữ cây thư mục trên bằng cách:

a./ Cho biết nội dung 15 phần tử của bảng FAT.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | -1 | -1 | 0 | 5 | 7 | 13 | 1 | 9 | -1 | 0 | 12 | -1 | -1 | 0 | -1 |

b./ Cho biết nội dung 5 thuộc tính: filename, fileext, attribute, start cluster, size của entry trong thư mục gốc và thư mục Amnhac.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filename | Ext | attrib | Start cluster | size |
| Hello | cpp |  | 2 | 824 |
| Amnhac |  | D | 9 |  |
| Hnoi | cpp |  | 15 | 1021 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filename | Ext | attrib | Start cluster | size |
| Tcson | Mp3 |  | 6 | 1489 |
| Ntmien | Mp3 |  | 4 | 2320 |

**Câu 23:** Hàng đợi đĩa gồm các yêu cầu đọc dữ liệu tại các cylinder theo thứ tự sau:

137, 57, 52, 84, 135, 72, 21, 85, 38, 123, 27

Đầu đĩa đang ở vị trí 50, đĩa được đánh số từ 0-199. Hãy tính quãng đường di chuyển của đầu đọc nếu điều phối theo

* 1. FCFS

0 21 27 38 50 52 57 72 84 85 123 135 137 199

(137-50)+(137-57)+(57-52)+(84-52)+(135-84)+(135-72)+(72-21)+(85-21)+(85-38)+(123-38)+(123-27)=661 cylinder

1. SSTF

0 21 27 38 50 52 57 72 84 85 123 135 137 199

(52-50)+(57-52)+(72-57)+(84-72)+(85-84)+(123-85)+(135-123)+(137-135)+(137-38)+(38-27)+(27-21)=204 cylinder

* 1. SCAN

0 21 27 38 50 52 57 72 84 85 123 135 137 199

(50-38)+(38-27)+(27-21)+21+52+5+(72-57)+(84-72)+1+(123-85)+(135-123)+2=433

* 1. C-SCAN

0 21 27 38 50 52 57 72 84 85 123 135 137 199

2+5+(72-57)+(84-72)+1+(123-85)+ (135-123)+2+(199-137)+21+5+(38-27)=186

* 1. LOOK

0 21 27 38 50 52 57 72 84 85 123 135 137 199

(50-38)+(38-27)+(27-21)+52+5+(72-57)+(84-72)+1+(123-85)+(135-123)+2=412

* 1. C-LOOK

0 21 27 38 50 52 57 72 84 85 123 135 137 199

2+5+(72-57)+(84-72)+1+(123-85)+ (135-123)+2+21+5+(38-27)=124 cylinder

**Câu 24:** Nêu nguyên tắc hoạt động của cơ chế intterupt. Cơ chế intterupt có ưu điểm gì so với cơ chế polling?

►Nguyên tắc hoạt động: Khi thiết bị nhập có dữ liệu: -Thiết bị gởi tín hiệu ngắt về CPU -CPU tạm dừng chương trình đang thi hành -CPU thi hành trình xử lý ngắt để đọc dữ liệu tại port -Sau khi hoàn tất, CPU tiếp tục chương trình bị dừng

►Cơ chế intterupt có ưu điểm một ngắt là một cơ chế phần cứng, còn cơ chế polling là thăm dò ý kiến là một Giao thức.

**Câu 25:** Nêu nguyên tắc hoạt động của cơ chế DMA. Mục đích của DMA là gì?

►Nguyên tắc hoạt động: DMA thường thì hoạt động chuyển thông tin giữa thiết bị I/O với bộ nhớ máy tính được thực hiện thông qua CPU theo cơ chế tuần tự: CPU ra lệnh "import" để nhập 1 đơn vị thông tin (byte/word) từ thiết bị I/O vào thanh ghi CPU rồi ra lệnh ghi thông tin từ thanh ghi CPU vào ô nhớ RAM nào đó. Quy trình này rất chậm vì CPU phải tốn nhiều chu kỳ máy để thực hiện các lệnh di chuyển thông tin, hơn nữa thông tin còn phải đi vòng qua CPU trước khi đến được vị trí cuối cùng.

►Mục đích: người ta đã thiết kế mạch cứng DMA cho phép chuyển dữ liệu trực tiếp giữa I/O và RAM mà không cần đi ngang qua CPU để cho tăng tốc độ chuyển dữ liệu giữa I/O và RAM