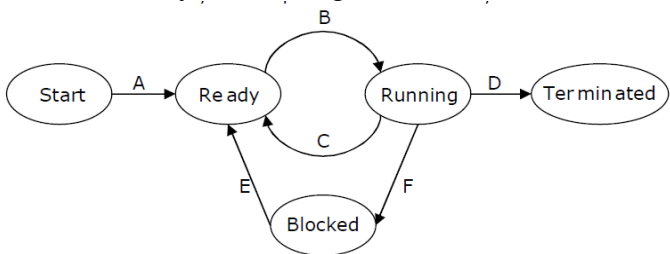


BÀI TẬP HỆ ĐIỀU HÀNH – CHƯƠNG 2

| Câu | Câu hỏi |
|-----|---|
| 1. | Ưu điểm của đa chương trình (multiprogramming) là gì? |
| 2. | <p>Với sơ đồ chuyển đổi trạng thái sau:</p>  <pre> graph LR Start([Start]) -- A --> Ready([Ready]) Ready -- B --> Running([Running]) Running -- C --> Ready Running -- D --> Terminated([Terminated]) Running -- F --> Blocked([Blocked]) Blocked -- E --> Ready </pre> <p>Với A, B, C, D, E, F là các bước chuyển đổi trạng thái. Giả sử rằng luôn có một số tiến trình ở trạng thái ready, với các phát biểu sau:</p> <ol style="list-style-type: none"> Nếu một tiến trình thực hiện chuyển đổi D, sẽ có một tiến trình khác thực hiện chuyển đổi A ngay lập tức. Một tiến trình P ở trạng thái bị Blocked có thể thực hiện chuyển đổi E trong khi một tiến trình khác ở trạng thái Running Hệ điều hành có vai trò điều phối tiến trình Nếu một tiến trình thực hiện chuyển đổi F, sẽ có một tiến trình khác thực hiện chuyển đổi E ngay lập tức <p>Đáp án nào sau đây là chính xác? Giải thích</p> <p>(A) 1 và 2 đúng (B) 1 và 3 đúng (C) 2 và 3 đúng (D) 1 và 4 đúng</p> |
| 1. | <p>Khi thực thi đoạn chương trình sau (trong file tinhtong.cpp):</p> <pre> int main () { int a, b; cout << "Nhap a va b:"; cin >> a >> b; int c = a + b; cout << "Tong cua a va b la " << c << endl; return 0; } </pre> <p>Hãy mô tả quá trình chuyển đổi trạng thái của tiến trình khi chương trình trên được thực thi</p> |
| 2. | Cho biết một số lợi ích của lập trình đa luồng |
| 3. | <p>Giả sử một tiến trình thực thi đoạn code sau</p> <pre> fork(); fork(); fork(); </pre> <p>Cho biết có bao nhiêu tiến trình con được tạo?</p> |
| 4. | <p>Cho đoạn chương trình sau:</p> <pre> int X = 0; do { X = X + 1; if (X==50) X = 0; }while (TRUE); </pre> |

| | |
|----|--|
| | Giả sử có hai tiến trình cùng thực hiện đoạn chương trình trên, cho biết có xảy ra Race Condition không? Nếu có hãy giải thích, xác định miền vắng và đề xuất giải pháp xử lý |
| 5. | <p>Giả sử hai tiến trình P1 và P2 thực thi hai đoạn code độc lập như sau:</p> <p>P1: while (S1 == S2) ; Critical Section S1 = S2;</p> <p>P2: while (S1 != S2) ; Critical Section S2 = not (S1);</p> <p>Phát biểu nào sau đây là chính xác về vấn đề đồng bộ hoạt động giữa hai tiến trình? Giải thích?</p> <p>A. Đạt mục tiêu Mutual exclusion nhưng không đạt Progress B. Đạt mục tiêu Progress nhưng không đạt Mutual exclusion C. Không đạt cả hai mục tiêu Mutual exclusion và Progress D. Đạt cả hai mục tiêu Mutual exclusion và Progress</p> |
| 6. | <p>Xét giải pháp đồng bộ hoá sau :</p> <pre>while (TRUE) { int j = 1 - i; flag [i]= TRUE; turn = i; while (turn == j && flag[j]==TRUE); critical-section (); flag[i] = FALSE; Noncritical-section (); }</pre> <p>Đây có phải là một giải pháp bảo đảm được độc quyền truy xuất không?</p> |
| 7. | <p>Với thuật toán Peterson's, giả sử có hai tiến trình P_i và P_j cùng thực thi với đoạn chương trình sau đây:</p> <pre>repeat flag [i] = true; turn = j; while (B) do no-op; Enter CS Exit CS flag [i] = false; NonCS until false;</pre> <p>Để chương trình đảm bảo loại trừ lẫn nhau, biểu thức điều kiện B trong vòng lặp while phải là</p> <p>a) flag [j] = true và turn = i b) flag [j] = true và turn = j c) flag [i] = true và turn = j d) flag [i] = true và turn = i</p> <p>Hãy chọn đáp án đúng và giải thích</p> |
| 8. | <p>Với giải pháp đồng bộ hóa tiến trình sử dụng lệnh test-and-set, các hàm enter_CS () và left_CS () để tiến trình đi vào và ra khỏi vùng CS như sau:</p> <pre>void enter_CS (X)</pre> |

```

{
    while test-and-set (X) ;
}
void leave_CS(X)
{
    X = 0;
}

```

Trong đó, X được khởi tạo là 0. Với các phát biểu sau:

I. Giải pháp trên không có deadlock (có một hoặc nhiều tiến trình vĩnh viễn không được thực thi xong)

II. Giải pháp trên không gây ra tình trạng starvation

III. Các tiến trình lần lượt vào CS theo thứ tự FIFO.

IV Nhiều tiến trình có thể vào CS cùng một lúc.

Đáp án nào sau đây là ĐÚNG?

- (A) Chỉ I đúng
- (B) I và II đúng
- (C) II và III đúng
- (D) Chỉ IV đúng

9. Điều phối tiến trình là gì? Tiêu chí nào ảnh hưởng đến hiệu suất của việc điều phối tiến trình?

10. Giả sử các tiến trình trong hệ thống cùng đến tại một thời điểm T_0 với thứ tự và thời gian sử dụng CPU được mô tả trong bảng sau.

| Process | Burst Time |
|---------|------------|
| P1 | 10 |
| P2 | 29 |
| P3 | 3 |
| P4 | 7 |
| P5 | 12 |

Thực hiện điều phối với các chiến lược FCFS, SJF, Round Robin (quantum=10)

Với mỗi chiến lược:

- Vẽ biểu đồ Gantt
- Tính thời gian chờ trung bình trong hệ thống

11. Giả sử các tiến trình trong hệ thống với thời điểm đến và thời gian sử dụng CPU được mô tả trong bảng sau:

| Process | Arrival Time | Burst Time | Priority |
|---------|--------------|------------|----------|
| P1 | 0 | 10 | 4 |
| P2 | 1 | 5 | 3 |
| P3 | 5 | 8 | 1 |
| P4 | 7 | 4 | 5 |
| P5 | 8 | 5 | 2 |

Thực hiện điều phối theo chiến lược:

- FCFS
- Priority độc quyền
- Priority không độc quyền

- | | |
|--|--|
| | <ul style="list-style-type: none">- SJF- SRTF- Round Robin (quantum =2)- HRRN |
|--|--|

Với mỗi chiến lược:

- Vẽ biểu đồ Gantt
- Tính thời gian đáp ứng, thời gian hoàn thành, thời gian chờ trung bình trong hệ thống