**Bài 14/126: Bài toán cây cầu cũ**

* **Tối đa 3 xe được phép qua cầu cùng một thời điểm.**

#define N 3 // Điều kiện xe tối đa typedef int sephamore;

sephamore permission = N; // Cờ xác định xe được phép qua cầu

**ArriveBridge(direction)**

{

// Khi xe đến cầu, hạ số xe được phép qua cầu xuống

// Nếu xuống bằng 0 thì chờ cho đến khi permission >0

// Nếu vẫn còn > 0 thì thực hiện xử lí tiếp theo down(&permission);

}

**ExitBridge()**

{

up(&permission); // Tăng số xe được phép qua cầu

}

**Bài 15/126: Bài toán sản xuất xe hơi.**

int count\_tire = 0; // Đếm số bánh xe đã được sản xuất

**Make\_tire()** {

// Đặt vòng lặp vô hạn chỗ này để việc sản xuất bánh xe được liên tục.

**Produce\_tire();** // Sản xuất bánh xe

count\_tire = count\_tire + 1; // Tăng số bánh xe đã sản xuất được if ( count\_tire == 4 ){

count\_tire = 0;

up(&tire\_available); // Đủ bánh xe để làm khung xe

}

**Put\_tire\_to\_chassis()** {

// Nếu đủ 4 bánh và khung xe thì thực hiện lắp ghép

// Nếu không đủ thì chờ down(&chassis\_available);

}

}

**Bài 2/156:**

* Việc phân biệt giữa CPU-bounded và I/O-bounded là quan trọng đối với các bộ điều phối tiến trình vì càng ngày tốc độ xử lí của CPU được cải tiến rất nhanh, so với tốc độ đọc ghi trên các thiết bị lưu trữ ngoài.Vì vậy sẽ ngày càng nhiều tiến

**Bài 3/156:**

Average Turn-around Time: (4+11+47+59+92+98)/6 = 51.83 Average Waiting: (0+4+35+39+67+68)/6 = 35.5

**Bài 4/157:**

|  |  |  |
| --- | --- | --- |
| Process | Enter Time | CPU Burst |
| P1 | 0 | 10 |
| P2 | 1 | 5 |
| P3 | 2 | 9 |
| P4 | 3 | 6 |

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

0 4 8 12 16 20 21 25 27 29 30

Average Turn-around Time: (29+20+28+24)/4 = 25.25 Average Waiting: (19+15+19+18)/4= 17.75

**Bài 5/157:**

\* Thứ tự tiến trình thực thi quan sát được: 1a – 1b – 2a – 2b – 1b. Thuật toán điều phối có thể đã được hệ điều hành sử dụng:

* RR: round-robin :

Khi thời gian xử lí của tiến trình 1b > quantum time > 1a,2a,2b. Như vậy khi đến tiến trình 1b, máy sẽ chưa xử lí xong nên sẽ thấy tiến trình 1b được xử lí ở một lần nữa khi xong tiến trình 2b.

* PRI:

Khi điều phối theo độ ưu tiên sử dụng nguyên tắc không độc quyền. Trường hợp đề bài xảy ra khi độ ưu tiên của các tiến trình được xếp theo thứ tự **giảm** dần của các tiến trình từ 1a – 2a – 2b – 1b. Như vậy, khi xử lí xong tiến trình 1a và đến tiến trình 1b, đang xử lí tiến trình 1b thì tiến trình 2a được đưa vào hàng chờ. Vì 2a có độ ưu tiên cao hơn nên hệ thống sẽ dừng xử lí ở tiến trình 1b mà chuyển qua xử lí ở 2a, tương tự đối với 2b. Cuối cùng sẽ xử lí phần còn lại của tiến trình 1b.