KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG MÁY TÍNH VÀ TT DỮ LIỆU

Hướng dẫn Lab 7.2 – Giải pháp Peterson

Một giải pháp phần mềm cho vấn đề cạnh tranh vùng nguy cơ được đề xuất bởi Peterson, trong đó một biến số nguyên và hai biến luận lý được dùng. Thật không may, các biến số cũng bị tình trạng cạnh tranh và giải pháp này không hoạt động, mặc dù thường được nhắc đến do tính chất đơn giản trong bài giảng lý thuyết. Hơn nữa, Peterson cũng thất bại trong kiến trúc đa nhân.

Trong LAB này, giải pháp Peterson được áp dụng vào bài toán Sản xuất – Tiêu thụ, mặc dù cuối cùng, dữ liệu vẫn sai sót.

Mục tiêu	**Lý thuyết liên quan**	**Tài nguyên**
Giải pháp Peterson	Ch6.3 Peterson solution	https://github.com/Trantin84/LAB_IntroOS (mã nguồn ví dụ). Sử dụng image Ubuntu 16 / 18
Bài toán P-C		
Bài toán số PI		

Yêu cầu sinh viên: Hiểu lý thuyết về liên lạc giữa các tiến trình. Hiểu và thực thi các đoạn mã đã cung cấp. Áp dụng cho các bài toán liên lạc giữa các tiến trình.

Đánh giá sinh viên: Hỏi đáp các vấn đề lý thuyết. Kỹ năng thực hành. Bài tập.

Yêu cầu nộp bài: các tập tin mã nguồn .c và tập tin khả thực thi .out của các Yêu cầu trong buổi thực hành và Bài tập cuối hướng dẫn trong thời gian cho phép của giảng viên.

Preferences

[1] Abraham Silberschatz, Peter B. Galvin, Greg Gagne, [2018], Operating System Concepts, 10th edition, John Wiley & Sons, New Jersey.

Programming Problems of Chapter 3.

[2] Greg Gagne , [2019], GitHub OS-BOOK OSC10e, Westminster College, United States

Access https://github.com/greggagne/osc10e in September 2019.

Yêu cầu 1: Giải pháp Peterson áp dụng cho bài toán Consumer – Producer như thế nào?

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_7/task2_1_PC_Peterson.c***, biên dịch và thực thi.

- Dòng 68 và 83 lần lượt là vùng nguy cơ của tiến trình Producer và Consumer. Tại sao?

- Dòng 64 – 66 là Entry section của Producer, hãy so sánh với dòng 79 – 81 là Entry section của Consumer.

- Dòng 70 và 85 là các Exit section.

```
64      flag[0] = true;
65      turn = 1;
66      while (flag[1] && turn == 1);
68      count++;
70      flag[0] = false;

79      flag[1] = true;
80      turn = 0;
81      while (flag[0] && turn == 0);
83      count--;
85      flag[1] = false;
```

Giải pháp này có hoạt động không?

```
$ gcc -o pc.out task2_1_PC_Peterson.c -lpthread
$ ./pc.out 100000
```

Bài tập lập trình.

1. Xem xét Yêu cầu 2 trong LAB 7.1 (***LAB_IntroOS/LAB_7/task1_2_PI.c***)

   Giả sử rằng chỉ có 2 tiểu trình được tạo ra khi gọi chạy, hãy áp dụng Peterson và đánh giá hiệu quả của giải thuật (độ chính xác của số Pi có được cải thiện không?).

```
$gcc -o pi.out task1_2_PI_ex.c -lpthread
$./pi.out 100
$./pi.out 10000
$./pi.out 100000
```