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

Hướng dẫn Lab 7.3 – Hàng rào bộ nhớ

Dữ liệu không nhất quán xảy ra khi nhiều câu lệnh thao tác lên dữ liệu cùng được thực thi và ghi lên thanh ghi / bộ nhớ. Nếu như các thao tác này được ra hiệu để vùng nhớ được bảo vệ đơn nhất, sai sót sẽ được tránh khỏi. Hàng rào bộ nhớ được hiện thực ở nhiều cấp độ, từ lệnh mã máy, có sẵn trong bộ biên dịch hay là thư viện POSIX.

Trong LAB này hai mức hiện thực được giới thiệu là "Built-in gcc" và "POSIX **pthread.h** library" được giới thiệu lần lượt ở 2 yêu cầu. Một số thư viện khác như **stdatomic.h** cũng hiện thực nhiều lời gọi đơn nguyên.

Mục tiêu	**Lý thuyết liên quan**	**Tài nguyên**
Hàng rào bộ nhớ Đồng bộ thao tác	Ch6.4.1 6.4.1 Memory Barriers	https://github.com/Trantin84/LAB_IntroOS (mã nguồn ví dụ). Sử dụng image Ubuntu 16 / 18

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 6.

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

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

[3] gcc manual, [2021], Legacy __sync Built-in Functions for Atomic Memory Access,

Access https://gcc.gnu.org/onlinedocs/gcc/_005f_005fsync-Builtins.html

[4] [3] Linux manual page, [2021], on *pthread_barrier_wait(3p)*,

Access https://man7.org/linux/man-pages/man3/pthread_barrier_wait.3p.html

Yêu cầu 1: Mô tả sự cần thiết của đồng bộ giữa bộ cộng và bộ in. Slide 6.18 và Sách [1] p. 266

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_7/task3_1_sumandprint.c***, biên dịch và thực thi.

- Dòng 18 là cờ hiệu đồng bộ giữa 2 tiểu trình.

- Dòng 19 cho một giá trị ngẫu nhiên của x.

- Dòng 38 (nếu chưa thực thi) sẽ ngăn chặn dòng 47. Việc dỡ ngăn chặn sẽ xảy ra khi Dòng 37 hoàn thành. Các lời gọi ở dòng 38 và 46 là built-in, tức là không cần thêm vào thư viện nào cả.

- Dòng 39 và 45 mô tả lại giải pháp Peterson.

** Lưu ý rằng, sự sai sót dữ liệu xảy ra mang tính ngẫu nhiên.*

```
18    bool flag = false;
19    int x = 83432; // random

37      x = 100;
38      __sync_synchronize();
39      flag = true;

45      if (!flag)
46        __sync_synchronize();
47      printf("\nValue x = %d", x);
```

Yêu cầu 2: Có 3 sinh viên cùng hẹn đi xem phim tại CGV, mỗi sinh viên cần lần lượt a, b, c giây để đến rạp. Cả nhóm chỉ mua vé khi đã có mặt đầy đủ. Người bán vé ở rạp phim cần chờ bao lâu?

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_7/task3_2_CGV.c***, biên dịch và thực thi.

- Dòng 6 là thư viện pthread.h bao gồm cả Hàng rào bộ nhớ mà chúng ta muốn sử dụng, tại dòng 18.

- Dòng 18 khai báo một hàng rào để sử dụng.

- Dòng 19 và 20 là thao tác của sinh viên và người bán vé.

- Dòng 23 là khởi tạo giá trị điểm hẹn cho biến barrier là 4, tức là có 4 mốc cần được gọi tín hiệu chờ thì hàng rào mới hoàn thành (và mở ra).

- Dòng 41 và 43 là mô tả thời gian đi đường với số giây đã truyền vào từ lời gọi cho mỗi sinh viên. Và khi vòng lặp này hoàn tất thì tiểu trình đó mới đạt đến điểm hẹn tại Dòng 46.

- Với người bán vé, hành động bán vé dòng 56 chỉ xảy ra khi điểm hẹn ở Dòng 55 đạt đủ số lần gọi (ở đây là 4: 3 sinh viên và chính người bán vé).

```
6       #include<pthread.h>
18      pthread_barrier_t barrier;
19      void * runner(void * param);
20      void * CGV(void * param);

23        pthread_barrier_init( & barrier, NULL, 4);

41        for (i = 1; i <= run; i++) {
43          sleep(1);
46        pthread_barrier_wait( & barrier);

55        pthread_barrier_wait( & barrier);
56        printf("\nCGV: Ban ve xem phim, sau ");
```

Bài tập lập trình.

1. Xem bài toán Ước lượng số PI (***LAB_IntroOS/LAB_7/task1_2_PI.c***)
   Xoá các dòng 43, 44

   ```
   for (int i = 0; i < n_thread; i++)
     pthread_join(tid[i], NULL);
   ```

   Lúc này lệnh in ở dòng 49 và 51 sẽ lập tức thực thi, mặc dù các tiểu trình chưa hoàn thành.

   ```
   printf("\nGettimeofday ...
printf("\nUoc tinh PI =  ...
   ```

   Hãy áp dụng phương pháp Hàng rào bộ nhớ để đồng bộ các tiểu trình.

2. Với bài toán Producer – Consumer, có thể dùng Hàng rào bộ nhớ để đồng bộ không? Tại sao?