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

Hướng dẫn Lab 8.3 – Semaphore có tên.

LAB 8.2 đã nói về semaphore không tên, được sử dụng nội bộ giữa các tiểu trình HOẶC tiến trình cha – con. Trong trường hợp là các tiến trình độc lập, semaphore có tên cần được dùng kèm theo các Vùng nhớ chia sẻ hay Hàng đợi thông điệp (Xem LAB 5 về các IPC)

Mục tiêu	Lý thuyết liên quan	Tài nguyên
Semaphore có tên	Ch6.6.1 Semaphore Usage	https://github.com/Trantin84/LAB_IntroOS (mã nguồn ví dụ). Sử dụng image Ubuntu 16 / 18
Thao tác trên semaphore		
Áp dụng cho bài toán P-C		

Yêu cầu sinh viên: Hiểu lý thuyết về semaphore. Phân tích khi nào dùng semaphore không tên và có tên. Áp dụng vào các bài toán đồng bộ.

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] Linux manual page, [2021], on *sem_open(3),*

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

[3] Filipe Gonçalves, [2021], Stackoverflow "Share POSIX semaphore among multiple processes",

Access https://stackoverflow.com/questions/32205396/share-posix-semaphore-among-multiple-processes

Yêu cầu 1: (SV tự học) Mô phỏng bài toán Producer – Consumer liên lạc qua Vùng nhớ chia sẻ.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_8/task3_1_Producer.c và task3_1_Consumer.c***, biên dịch và thực thi.

** Chọn lựa "-lpthread" vẫn cần thiết cho mục đích khác.*

Trên Terminal thứ nhất

```
$ ./gcc -o producer task3_1_Producer.c -lrt -lpthread
$ ./producer 1000
```

Trên Terminal thứ hai

```
$ ./gcc -o consumer task3_1_Consumer.c -lrt -lpthread
$ Receive = ...
```

Yêu cầu 2: Đồng bộ bài toán Producer – Consumer ở Yêu cầu 1 bằng semaphore có tên.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_8/task3_1_ProducerSEM.c và task3_1_ConsumerSEM.c***, biên dịch và thực thi.

Tiến trình Producer

- Vùng nhớ chia sẻ được tạo ra để chứa n vị trí Buffer + 1 vị trí biến count + 1 vị trí đánh dấu việc gửi hoàn tất.

- Dòng 22 khai báo Tên của semaphore là **`/sem-mutex`**

- Dòng 26 khai báo con trỏ kiểu semaphore để quản lý semaphore được tạo ra ở dòng 30 có ý nghĩa:

  - Tên đã khai báo ở dòng 22, giống nhau ở mọi tiến trình.
  - Được tạo mới (tiến trình Producer tạo ra)
  - Toàn quyền đọc ghi 0666.
  - Giá trị khởi tạo là 1: vì được dùng như Mutex Semaphore.

- Dòng 55 và 57 là lời gọi đồng bộ bảo vệ vùng nguy cơ (dòng 56 – biến count được chia sẻ thông qua một ô nhớ trong vùng nhớ chia sẻ).

```
22      #define SEM_MUTEX_NAME "/sem-mutex"
26        sem_t * mutex_sem;

30      mutex_sem = sem_open(SEM_MUTEX_NAME, O_CREAT, 0660, 1);

55          sem_wait(mutex_sem);
56          ptr[BUFFER_SIZE]++;    //varible count
57          sem_post(mutex_sem);
```

Tiến trình Consumer

- Dòng 20 khai báo Tên của semaphore là **`/sem-mutex`**

- Dòng 23 khai báo con trỏ kiểu semaphore để quản lý semaphore ở dòng 25 "có tên được khai báo và xin quyền đọc ghi". So sánh với lời gọi "sem_open" của Producer.

- Dòng 47 và 49 là lời gọi đồng bộ bảo vệ vùng nguy cơ là dòng 48 (count--)

- Dòng 52 huỷ bỏ semaphore.

```
20      #define SEM_MUTEX_NAME "/sem-mutex"
23        sem_t * mutex_sem;

25        mutex_sem = sem_open(SEM_MUTEX_NAME, O_RDWR);

47          sem_wait(mutex_sem);
48          ptr[BUFFER_SIZE]--;    //varible count
49          sem_post(mutex_sem);

52        sem_unlink(SEM_MUTEX_NAME);
```

Bài tập lập trình.

1. Nộp các đoạn mã đã biên dịch và thực thi.

2. Trong Yêu cầu 2 đã thực hiện chúng ta đã đồng bộ bảo vệ biến count (`ptr[BUFFER_SIZE]`). Tuy vậy, vẫn còn hai nơi gây ra tình trạng busy waiting làm tiêu hao tài nguyên hệ thống. Đó là:

   ```
   (Producer) while (ptr[BUFFER_SIZE] == BUFFER_SIZE;
   (Consumer) while (ptr[BUFFER_SIZE]  == 0);
   ```

   Hãy tạo ra 2 semaphore full và empty, để thay thế 2 dòng trên. Bài tập này tương tự Bài tập 1 trong LAB 8.2 nhưng cần hiện thực là semaphore có tên.