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

Hướng dẫn Lab 8.4 – Biến điều kiện.

Các biến điều kiện [condition variable] cung cấp một cách khác cho các tiến trình để đồng bộ hóa. Trong khi các biết mutex thực hiện đồng bộ hóa bằng cách kiểm soát truy cập của tiểu trình vào dữ liệu, các biến điều kiện cho phép các tiểu trình xử lý đồng bộ hóa dựa trên giá trị thực của dữ liệu.

Nếu không có các biến điều kiện, lập trình viên sẽ cần phải có các tiểu trình liên tục thăm dò [polling] (có thể trong đoạn mã nguy cơ), để kiểm tra xem điều kiện có được đáp ứng hay không. Điều này có thể rất tốn tài nguyên vì tiểu trình sẽ liên tục bận rộn trong hoạt động này [busy waiting]. Một biến điều kiện là một cách để đạt được cùng một mục tiêu mà không cần thăm dò.

Một biến điều kiện luôn được sử dụng cùng với khóa mutex.

Mục tiêu	**Lý thuyết liên quan**	**Tài nguyên**
Biến điều kiện	Ch7.3.3 POSIX Condition Variables	https://github.com/Trantin84/LAB_IntroOS (mã nguồn ví dụ). Sử dụng image Ubuntu 16 / 18
Ứng dụng biến điều kiện		

Yêu cầu sinh viên: Hiểu lý thuyết về biến điều kiện và lợi ích của nó. Hiểu và thực thi bài tập ví dụ.

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 *pthread_cond_destroy(3p)*,

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

Yêu cầu 1: (SV tự học) Cho một chương trình có 2 tiểu trình cùng thao tác tăng giá trị biến số chia sẻ; tiểu trình thứ 3 quan sát quá trình đó và thông báo khi biến số chia sẻ đạt một giá trị nhất định.

Hướng dẫn: Tải về tập tin ***LAB_IntroOS/LAB_8/task4_1_WatchSum.c***, biên dịch và thực thi.

Bài tập lập trình.

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

Phụ lục

Tạo ra và hủy bỏ các biến điều kiện

```
pthread_cond_init (condition,attr)
pthread_cond_destroy (condition)
pthread_condattr_init (attr)
pthread_condattr_destroy (attr)
```

Các biến điều kiện phải được khai báo với loại pthread_cond_t và phải được khởi tạo trước khi chúng có thể được sử dụng. Có hai cách để khởi tạo một biến điều kiện:

Khai báo tĩnh:

```
pthread_cond_t myconvar = PTHREAD_COND_INITIALIZER;
```

Khai báo động với hàm pthread_cond_init(). ID của biến điều kiện đã tạo được trả về tiểu trình gọi thông qua tham số điều kiện. Phương pháp này cho phép thiết lập các thuộc tính đối tượng biến điều kiện thông qua attr.

Đối tượng attr tùy chọn được sử dụng để đặt thuộc tính biến điều kiện. Chỉ có một thuộc tính được xác định cho các biến điều kiện: process-shared, cho phép biến điều kiện được nhìn thấy

bởi các tiểu trình trong các tiến trình khác. Đối tượng thuộc tính, nếu được sử dụng, phải là loại pthread_condattr_t (có thể được chỉ định là NULL để chấp nhận mặc định).

Lưu ý rằng không phải tất cả các hiện thực có thể cung cấp thuộc tính process-shared.

Lời gọi pthread_condattr_init() và pthread_condattr_destroy() được dùng để tạo và hủy các đối tượng thuộc tính của các biến số điều kiện. pthread_cond_destroy() dược dùng để hủy bỏ một biến điều kiện không còn sử dụng nữa.

Ra hiệu và chờ tín hiệu với biến điều kiện

```
pthread_cond_wait (condition,mutex)
pthread_cond_signal (condition)
pthread_cond_broadcast (condition)
```

- Lời gọi pthread_cond_wait () chặn tiểu trình đã thực hiện lời gọi cho đến khi điều kiện đã được chỉ định trước có tín hiệu. Lời gọi này cần thực thi khi mutex bị khóa và nó sẽ tự động giải phóng mutex trong khi nó chờ. Sau khi tín hiệu được nhận và tiểu trình được đánh thức, mutex sẽ tự động bị khóa để sử dụng bởi tiểu trình. Lập trình viên cần mở khóa mutex khi tiểu trình kết thúc với nó.

- Ghi chú: Sử dụng vòng lặp WHILE thay vì câu lệnh IF (xem lời gọi watch_count trong ví dụ 5.1) để kiểm tra điều kiện chờ đợi để tránh một số vấn đề tiềm ẩn, chẳng hạn như:

+ Nếu một số tiểu trình đang chờ tín hiệu đánh thức giống nhau, chúng sẽ lần lượt lấy được mutex và bất kỳ một trong số chúng sau đó có thể sửa đổi điều kiện mà tất cả chúng chờ đợi.

+ Nếu tiểu trình nhận được tín hiệu do lỗi chương trình

+ Thư viện Pthreads được phép đưa ra các đánh thức giả cho một chuỗi chờ mà không vi phạm tiêu chuẩn.

- Lời gọi pthread_cond_signal() được sử dụng để báo hiệu (hoặc đánh thức) một tiểu trình khác đang chờ trên biến điều kiện. Nó nên được gọi sau khi mutex bị khóa và phải mở khóa mutex để hoàn thành lời gọi pthread_cond_wait().

- Lời gọi pthread_cond_broadcast () nên được sử dụng thay cho pthread_cond_signal () nếu có nhiều hơn một tiểu trình trong trạng thái chờ chặn.

- Sẽ xuất hiện lỗi luận lý khi gọi pthread_cond_signal() trước khi gọi pthread_cond_wait().

- Lưu ý: Khóa và mở khóa của biến mutex liên quan cần phải được thực thi đúng và hợp lý. Ví dụ:

+ Không khóa mutex trước khi gọi pthread_cond_wait () có thể khiến nó KHÔNG bị chặn.

+ Không mở khóa mutex sau khi gọi pthread_cond_signal() có thể không cho phép một lời gọi pthread_cond_wait() phù hợp hoàn thành (nó sẽ vẫn bị chặn).