|  |
| --- |
| **TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI** |
| **VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG** |
| A picture containing logo  Description automatically generated |
|  |

|  |
| --- |
| BÁO CÁO BÀI TẬP LỚN |
| MÔN: NGUYÊN LÝ HỆ ĐIỀU HÀNH |
| *Chủ đề: Xây dựng chương trình minh họa giải pháp cho bài toán Reader – Writer* |

|  |  |
| --- | --- |
| Giảng viên: | TS. Đỗ Quốc Huy |
| Sinh viên: | Chu Đình Đức 20194021 |
|  | Chu Nhật Minh 2019 |
| Lớp: | Khoa học máy tính 05 – K64 |

Mục lục

Phần 0: Lời nói đầu và một số khái niệm cơ bản

1. Một số khái niệm cơ bản

2. Bài toán đồng bộ tiến trình

3. Yêu cầu của chương trình điều độ

4. Các giải pháp đồng bộ hóa

Phần 1: Giới thiệu đề tài

1. Bài toán readers – writers

2. Các vấn đề cần giải quyết

Phần 2: Nội dung chính của đề tài

1. Giải quyết bài toán

2. Giải thuật và lời giải đã lựa chọn: Kỹ thuật đèn báo

Phần 3: Miêu tả chương trình

1. Môi trường phát triển chương trình

2. Biểu đồ UML miêu tả chương trình

3. Các hàm trong chương trình

4. Kết quả chạy chương trình

Phần 4: Miêu tả giao diện chương trình

1. Môi trường phát triển

2. Sơ đồ hoạt động

3. Xây dựng chương trình

4. Kết quả demo

Phần 5: Tài liệu tham khảo

**Phần 0: Lời nói đầu và một số khái niệm cơ bản**

*1. Một số khái niện cơ bản*

* Process (tiến trình): Tiến trình là một chương trình đang xử lý, sở hữu một con trỏ lệnh, tập các thanh ghi và các biến. Để hoàn thành tác vụ của mình, một tiến trình có thể cần đến một số tài nguyên – như CPU, bộ nhớ chính, các tập tin và thiết bị nhập/xuất.
* Tài nguyên: tất cả những gì cần thiết cho thực hiện tiến trình
* Tài nguyên găng:
* Tài nguyên hạn chế về khả năng sử dụng chung
* Cần đông thời cho nhiều tiến trình
* Có thể là thiết bị vật lý hay dữ liệu dùng chung
* Đoạn găng: đoạn chương trình sử dụng tài nguyên găng.
* Thread (luồng): là một đơn vị xử lý độc lập của máy tính có thể thực hiện một công việc riêng biệt và có thể xem là một tiến trình con. Một process có thể chứa nhiều thread khác nhau (multithreading). Các thread trong cùng một process chia sẻ chung các tài nguyên như vùng nhớ.

*2. Bài toán đồng độ tiến trình*

Khảo sát các process/thread thực thi đồng thời và chia sẻ dữ liệu (qua shared memory, file) ta thấy:

* Nếu không có sự kiểm soát khi truy cập các dữ liệu dùng chung thì có thể dẫn tới trường hợp không nhất quán dữ liệu (data inconsistency), làm cho chương trình không xác định
* Để duy trì sự nhất quán dữ liệu, hệ thống cần có cơ chế bảo đảm sự thực thi có trật tự của các process đông thời, cơ chế đồng bộ hóa (synchronize) các tiến trình thực hiện đông thời.
* Xử lý đồng thời (concurrent): Là quá trình các thread chạy độc lập với nhau trong các khoản thời gian chồng chéo để xử lý các yêu cầu riêng biệt. Các yêu cầu đó có thể được dùng để hoàn thành một yêu lớn hơn.
* Đồng bộ hoá xử lý đồng thời: Khi chạy đồng thời, các thread có thể sử dụng chung các tài nguyên. Để tránh tình trạng xung đột giữa các luồng xử lý ta phải đồng bộ hoá các thread, tức là chỉ cho phép lần lượt các thread truy cập và sử dụng tài nguyên trong một thời điểm nhất định.

*3. Yêu cầu của chương trình điều độ*

* Loại trừ lẫn nhau (Mutual Exclusion): mỗi thời điểm, tài nguyên găng không phải phục vụ số lượng tiến trình vượt quá khả năng của nó.
* Tiến triển (Progress): tài nguyên găng còn khả năng phục vụ và tồn tại tiến trình muốn vào đoạn găng thì tiến trình đó phải được sử dụng tài nguyên găng.
* Chờ đợi hữu hạn (Bounded Waiting): nếu tài nguyên găng hết khả năng phục vụ và vẫn tồn tại tiến trình muốn vào đoạn găng, thì tiến trình đó phải được xếp hàng chờ đợi và sự chờ đợi là hữu hạn.

*4. Các giải pháp đồng bộ hóa*

* Phương pháp khóa trong
* Phương pháp kiểm tra và xác lập
* Kỹ thuật đèn báo
* Giải thuật Monitor

Về kỹ thuật đèn báo:

* Đèn báo (Semaphore): là một biến nguyên S, khởi tạo bằng khả năng phục vụ của tài nguyên nó điều độ.
* Giá trị của S chỉ có thể thay đổi bởi 2 thao tác cơ bản không xử lý tách rời:

|  |  |
| --- | --- |
| P(S) – wait(S)  wait(S){  while (S<=0) no-op;  S--;  } | V(S) – signal(S)  signal(S){  S++;  } |

**Phần 1: Về bài toán Reader – Writer**

*1. Bài toán readers - writers*:

Bài toán yêu cầu xử lý vấn đề khi nhiều thread sử dụng chung một tài nguyên (có thể là đọc hoặc ghi) với ràng buộc:

* Nhiều Readers có thể cùng lúc truy cập tài nguyên
* Khi 1 Writer đang truy cập tài nguyên thì không một tiến trình nào khác được truy cập

*2. Các vấn đề cần giải quyết:*

Yêu cầu đặt ra của bài toán là cấp phát quyền truy cập và sử dụng tài nguyên cho các thread một cách hợp lý, sao cho:

* Chỉ có một writer sử dụng tài nguyên tại một thời điểm nhất định.
* Các reader sử dụng chung tài nguyên tại một thời điểm nhất định và không chờ đợi lẫn nhau.
* Writer/Reader nào yêu cầu sử dụng tài nguyên trước sẽ được sử dụng trước.

**Phần 2: Nội dung chính của đề tài**

1. *Giải quyết bài toán*

Bài toán được giải quyết theo nguyên tắc ai đến trước sẽ được quyền sử dụng tài nguyên trước.

Đối với các Reader, nếu có một reader đang sử dụng tài nguyên mà tiếp theo sau là các Reader khác đang chờ thì cho phép các Reader đó truy cập tài nguyên (Do các Reader không làm thay đổi tài nguyên trong quá trình sử dụng). Như vậy sẽ giải quyết được vấn đề các reader đến sau nhưng được sử dụng tài nguyên trước các writer đến trước và vấn đề các Reader chờ đợi một Reader khác không cần thiết.

2. *Giải thuật và lời giải đã lựa chọn: Kỹ thuật đèn báo*

Dùng 2 semaphore (readLock và writeLock khởi tạo bằng 1) để lưu khả năng phục vụ của tài nguyên và 1 biến kiểu nguyên (readCount khởi tạo bằng 0) để kiểm tra quyền truy cập tài nguyên.

Khi readCount = 1 thì Writer không được truy cập vào tài nguyên, còn khi readCount = 0 thì Writer được phép truy cập vào tài nguyên.

Cấu trúc tổng quát được sử dụng để xây dựng chương trình ở cả 2 phần: **Phần 3: Chương trình minh họa** và **Phần 4: Giao diện chương trình.**

|  |  |
| --- | --- |
| TT Read | TT Write |
| P(RS)  RC++  if (RC==1) P(WS)  {Đoạn găng}  V(RS)  …  P(RS)  RC—  if (RC==0) V(WS)  {Đoạn găng}  V(RS) | P(WS)  {Đoạn găng}  V(WS) |

RS: readLock

WS: writeLock

RC: readCount

**Phần 3: Chương trình minh họa**

*1. Môi trường phát triển*

* Ngôn ngữ: Java
* IDE: IntelliJ IDEA
* Hệ điều hành: Window 10

*2. Biểu đồ UML miêu tả chương trình*

Diagram

Description automatically generated

*3. Các hàm trong chương trình*

* Chương trình sử dụng các lớp chồng nhau trong Java (Java inner class) cụ thể các lớp Read và Write nằm ở bên trong lớp ReaderAndWritersProblem.
* Lớp Read và lớp Write sử dụng giao diện Runnable.

a. Lớp ReaderAndWritersProblem

Text

Description automatically generated with medium confidence

* Mô tả cấu trúc: Ban đầu trong lớp này ta khai báo ra 2 biến kiểu Semaphore quản lý các tiến trình Read – Write (khởi tạo bằng 1) và 1 biến kiểu nguyên để kiểm tra quyền truy cập tài nguyên (khởi tạo bằng 0). Tiếp theo lớp chứa 2 lớp Read và Write. Cuối cùng là hàm main để chạy chương trình.
* Chức năng: chứa toàn bộ chương trình và chạy chương trình.
* *Khai báo biến readCount với từ khóa volatile nhằm mục đích giải quyết vấn đề: khi 2 tiến trình Reader thực hiện, chúng sử dụng biến readCount cùng lúc, vậy thì readCount tăng lên 1 chứ không phải 2. Khi khai báo biến với từ khóa volatile, ta đánh dấu biến này “đã được lưu trong bộ nhớ chính”, có nghĩa sự thay đổi của biến này ngay lập tức được ghi vào bộ nhớ chính, vào khi sử dụng biến này sẽ được lấy ra từ bộ nhớ chính chứ không phải từ bộ nhớ đệm của CPU.*
* *Hoặc nếu không khai báo volatile thì có thể giải quyết bằng cách đưa readCount++ và readCount—vào khối đồng độ hóa synchronized block như dưới đây*

Text

Description automatically generated with medium confidence

b. Lớp Read: implements Runnable interface, chứa phương thức run(), gồm 3 phần nhỏ. Hàm acquire() tương tự như thao tác P(S), hàm release() tương tự như thao tác V(S).

Phần nhỏ 1: Acquire section

Graphical user interface, text, application

Description automatically generated

* Mô tả: nếu readCount = 1 (có 1 tiến trình Read đang thực hiện) thì giảm writeLock về 0 – tiến trình Write bị khóa (thực chất là giảm khả năng phục vụ mà tài nguyên có thể phục vụ cho tiến trình Write)
* Chức năng: chắc chắn rằng nếu có 1 tiến trình Read đang truy cập tài nguyên thì tiến trình Write phải đợi.

Phần nhỏ 2: Reading Section

Text

Description automatically generated

* Mô tả: gồm 3 lệnh cơ bản: 2 lệnh hiên thị ra màn hình và 1 lệnh thiết lập thời gian.
* Chức năng: hiển thị ra màn hình thông báo bắt đầu và kết thúc 1 luồng, Thread.sleep() dùng để tạm ngừng thực thi luồng hiện tại trong khoảng thời gian random từ 1 đến 2000 ms, để các luồng khác được thực thi.

Phần nhỏ 3: Releasing Section

Graphical user interface, text, application

Description automatically generated

* Mô tả: nếu readCount = 0 (không còn tiến trình Read nào đang thực hiện) thì tăng writeLock lên 1.
* Chức năng: chắc chắn rằng nếu không còn tiến trình Read nào đang truy cập tài nguyên thì tiến trình Write được thực hiện.

c. Lớp Write

Text

Description automatically generated

* Mô tả: gồm các lệnh P(writeS) -> thread.sleep() -> V(writeS)
* Chức năng: chỉ cho phép 1 tiến trình Write được truy cập tài nguyên tại 1 thời điểm.

*4. Kết quả chạy chương trình*

Graphical user interface, text, application, email

Description automatically generated

**Phần 4: Giao diện chương trình**

*1. Môi trường phát triển*

* Ngôn ngữ: Java
* IDE: Eclipse
* Hệ điều hành: Window 10

*2. Sơ đồ hoạt động*

Diagram

Description automatically generated

*3. Xây dựng chương trình*

* Lớp Operator: Là lớp cha của lớp Reader/Writer. Thừa kế từ lớp JPanel, thể hiện một Writer/Reader trên màn hình.
* Lớp Writer: Là lớp con của lớp Operator, implements Runnable interface. Lớp này chứa một biến tham chiếu đến tài nguyên được sử dụng chung, hai biến semaphore để điều khiển việc truy cập tài nguyên. Mỗi object của lớp writer có thể remove hoặc add thêm một item trong stack (Là tài nguyên sử dụng chung, chứa các item). Mỗi writer được thể hiện bằng một phần tử màu xanh trên màn hình.
* Lớp Reader: Là lớp con của lớp Operator, implements Runnable interface. Lớp này chứa một biến tham chiếu đến tài nguyên được sử dụng chung, ba biến semaphore để điều khiển việc truy cập dữ liệu. Mỗi object của lớp reader có thể đọc số lượng item có trong stack (Là tài nguyên sử dụng chung, chứa các item) và in ra màn hình. Mỗi reader được thể hiện bằng một phần tử màu tím trên màn hình.
* Lớp Item: Kế thừa lớp JPanel, tạo ra 1 item.
* Lớp ProcessUI: Lớp này xử lý phần UI cho toàn bộ chương trình.
* Lớp Main: Tạo ra một đối tượng của lớp ProcessUI và hiển thị đối tượng ra màn hình. Trong lớp ProcessUI chứa hàm main để chạy chương trình.

Mô tả chương trình: Chương trình sẽ hiển thị hai button để người dùng tạo ra các reader, writer tương ứng. Với mỗi reader/writer được tạo ra, reader/writer đó được thể hiện bằng các phần tử xanh lá/đỏ tương ứng và được thêm vào vùng chờ trước khi được cấp phát quyền truy cập dữ liệu. Khi reader/writer được cấp phát quyền truy cập, chúng sẽ được chuyển sang vùng service (thể hiện reader/writer đang sử dụng tài nguyên). Sau khi sử dụng xong thì các reader/writer đó sẽ bị remove khỏi vùng service và quyền truy cập sẽ được cấp phát cho các reader/writer đang chờ trong khu vực chờ.

*4. Kết quả demo*

Graphical user interface

Description automatically generated

**Phần 5: Tài liệu tham khảo**

1. Giáo trình Hệ Điều Hành (Nguyên lý các hệ điều hành) – Viện CNTT&TT, ĐHBKHN

2. http://codereview.stackexchange.com

https://bitly.com.vn/9k5h21

3. https://sharecode.vn

https://bitly.com.vn/kjg1oa

4. https://en.wikipedia.org/wiki

<https://bitly.com.vn/ty3pfo>