TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

**BÁO CÁO**

MÔN: Nguyên lý hệ điều hành

**Tìm hiều về các bài toán kinh điển về đồng bộ tiến trình**

Sinh viên thực hiện : Nguyễn Quang Hưng

Mã sinh viên : 20183760

Lớp : Kỹ thuật máy tính **– Khoá 63**

Giáo viên hướng dẫn : TS. **Phạm Đăng Hải**

***Hà Nội, tháng 11 năm 2020***

# MỤC LỤC

[MỤC LỤC 2](#_Toc57057703)

[Danh mục hình ảnh 3](#_Toc57057704)

[Lời mở đầu 4](#_Toc57057705)

[CHƯƠNG 1. Bài toán đường ray xe lửa ở Andes – Rayways in the Andes 5](#_Toc57057706)

[1.1. Bài toán thực tiễn 5](#_Toc57057707)

[1.2. Thuật toán giải quyết bài toán 6](#_Toc57057708)

[CHƯƠNG 2. Vấn đề của người hút thuốc lá – Cigarette Smoker’s problem 7](#_Toc57057709)

[2.1. Bài toán 7](#_Toc57057710)

[2.2. Thuật toán giải quyết bài toán 7](#_Toc57057711)

[CHƯƠNG 3. Bài toán ông già Noel – Santa Claus Problem 9](#_Toc57057712)

[3.1. Bài toán 9](#_Toc57057713)

[3.2. Phân tích bài toán 10](#_Toc57057714)

[3.3. Thuật toán giải quyết 10](#_Toc57057715)

[CHƯƠNG 4. Bài toán người trợ giảng ngủ gật – The Sleeping Teaching Assistant Problem 11](#_Toc57057716)

[4.1. Bài toán 11](#_Toc57057717)

[4.2. Thuật toán giải quyết bài toán 11](#_Toc57057718)

[CHƯƠNG 5. Bài toán vượt sông – River Crossing Problem 13](#_Toc57057719)

[5.1. Bài toán 13](#_Toc57057720)

[5.2. Solution 13](#_Toc57057721)

[CHƯƠNG 6. Bài toán tàu lượn siêu tốc – The Roller Coaster Problem 14](#_Toc57057722)

[6.1. Bài toán 14](#_Toc57057723)

[6.2. Solution 15](#_Toc57057724)

[CHƯƠNG 7. Bài toán bữa tối của bộ tộc hoang dã – The Dining Savages Problem 16](#_Toc57057725)

[7.1. Bài toán thực tế 16](#_Toc57057726)

[7.2. Solution 16](#_Toc57057727)

[Tài liệu tham khảo 18](#_Toc57057728)

# Danh mục hình ảnh

[*Hình 1: Đường ray xe lửa ở Andes* 5](#_Toc57057697)

[*Hình 2: Vấn đề của người hút thuốc lá* 7](#_Toc57057698)

[*Hình 3: Bài toán ông già Noel* 9](#_Toc57057699)

# Lời mở đầu

Trong khoa học máy tính, đồng bộ hóa (**synchronization** ) được đề cập đến một trong hai khái niệm tuy khác biệt nhưng có liên quan đến nhau. Đó là đồng bộ hóa tiến trình hay điều độ tiến trình (**process** **synchronization**) và đồng bộ hóa dữ liệu (**data** **synchronization**). Trong đó, đồng bộ hóa tiến trình hướng đến ý tưởng rằng nhiều tiến trình sẽ được thực hiện đồng thời tại một thời điểm nhất định để đạt được thỏa thuận hay kết quả với 1 chuỗi hành động nhất định. Việc đồng bộ hóa tiến trình luôn là một bài toán xuyên suốt trong nguyên lý thuyết kế hệ thống và thiết kế các hệ điều hành, vì các tài nguyên phần cứng đang dần được nghiên cứu, cải tiến để đáp ứng được việc chạy đa tiến trình (vi xử lý đa lõi, bộ nhớ lớn truy cập nhanh, …).

Tài liệu báo cáo này xin trình bày một vài bài toán kinh điển về điều độ tiến trình:

* Railways in the Andes, a practical problem.

(Jerry Breecher – CS3013, Operating Systems Process Synchronization, page 25 )

* The Sleeping Teaching Assistant problem.

(Abraham Silberschatz, Peter Baer Galvin, Greg Gagne – Operating System Concepts 7th edition, page 251)

* The Cigarette Smokers' problem

( David Lorge, Parnas Carnegie Mellon University, On a solution to the cigarette smokers' problem )

* The Santa Claus problem

( William Stallings, Operating Systems: Internals and Design Principles. Prentice Hall, sixth edition )

* River Crossing Problem

(A Little Book of Semaphores, Allen B. Downey, Version 2.2.1, page 148)

* The Dining Savages Problem

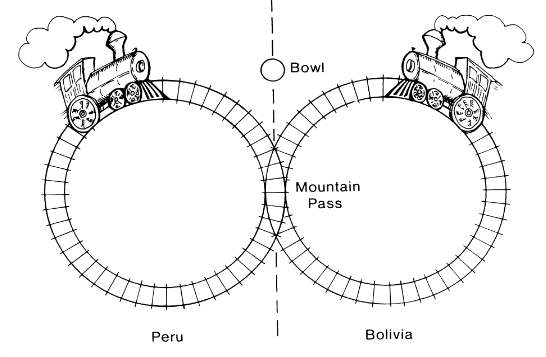
(A Little Book of Semaphores, Allen B. Downey, Version 2.2.1, page 115)

* The Roller Coaster Problem

(A Little Book of Semaphores, Allen B. Downey, Version 2.2.1, page 153)

# Bài toán đường ray xe lửa ở Andes – Rayways in the Andes

## Bài toán thực tiễn



*Hình 1: Đường ray xe lửa ở Andes*

Một dãy núi cao ở Andes, ở đó có 2 tuyến đường sắt tròn. Một đường ở Peru, một cái khác ở Bolivia. Họ cùng dùng chung , một phần của đường ray, nơi mà các đường băng qua một đèo núi nằm trên biên giới quốc tế (Gần hồ Titicaca).

Thật không may mắn, các đoàn tàu của Peru và Bolivian thỉnh thoảng va chạm nhau khi đồng thời đi vào phần chung của đường ray (đèo núi). Vấn đề là những người lái xe của cả hai tàu đều bị mù và điếc, do đó họ không thể nhìn thấy hay nghe thấy nhau.

Hai người lái tàu đã thống nhất về phương pháp ngăn ngừa va chạm sau đây. Họ đặt một bát lớn ở lối vào đèo. Trước khi vượt qua, người lái xe phải dừng tàu, đi qua bát, và chạm vào nó để thấy nó có chứa một tảng đá hay không. Nếu bát trống, người lái xe phải tìm một hòn đá và bỏ nó vào bát, cho biết chuyến tàu của mình đang đi qua; Một khi đoàn tàu của ông đã vượt qua được, ông phải đi bộ trở lại bát và loại bỏ đá của ông, để báo hiệu rằng chỗ dùng chung đó không có ai sử dụng nữa. Cuối cùng, anh ta quay trở lại tàu và tiếp tục lái xe.

Nếu người lái xe kia lái xe đến đèo tìm thấy trong bát có một hòn đá thì người đó phải rời chỗ chung đường ray đó để tránh va chạm. Do vậy anh ta phải cho đoàn tài nghỉ ngơi và kiểm tra lại cái tô cho đến khi thấy nó trống rỗng. Sau đó anh ta bỏ 1 hòn đá vào tô và cho tàu anh ta vào đèo. Một sinh viên thông minh đến từ đại học LaPaz (Bolivia) cho rằng cái cách này có thể làm cho chặn đường tàu mãi mãi. Nhưng người lái xe ở Bolivia chỉ cười và nói rằng điều đó không thể đúng bởi vì nó không bao giờ xảy ra.

Thật không may, vào một ngày cả 2 đàu đâm vào nhau. Sau vụ tai nạn, sinh viên đó được gọi đến để tư vấn để đảm bảo rằng sẽ không xảy ra tai nạn nữa. Anh ấy nói rằng cái bát đã được sử dụng sai cách. Người lái xe ở Bolivia phải đợi tại lối vào đèo cho đến khi cái bát trống, lái xe qua đèo và đi bộ trở lại để đặt một tảng đá vào bát. Người lái xe người Peru phải chờ đến khi bát có chứa đá, lái xe qua đèo và đi bộ để lấy tảng đá khỏi bát. Chắc chắn, phương pháp của ông ngăn ngừa tai nạn. Trước khi sắp xếp này, tàu của Peru chạy hai lần một ngày và tàu Bolivian chạy mỗi ngày một lần. Người Peru rất không hài lòng với cách sắp xếp mới.

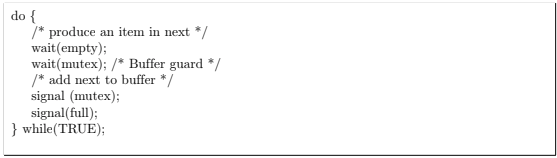
Do đó, sinh viên này được kêu gọi trở lại và được yêu cầu tìm giải pháp khác để ngăn ngừa được tai nạn và tránh được vấn đề phương pháp trước đây. Sau một thời gian suy nghĩ, sinh viên này đã đề nghị rằng nên sử dụng 2 cái bát, mỗi một bát cho mỗi người lái tàu, và cách thức hoạt động như sau: khi lái xe đến chỗ vào đầu tiên anh ta thả hòn đá vào bát của mình sau đó kiểm tra cái bát khác để xem nó có hòn đá nào không. Nếu nó trống thì anh ta lái xe của mình thông qua đường chuyền. Sau đó dừng lại và đi trở lại để loại bỏ hòn đá của mình. Nhưng nếu anh ta tìm thấy một hòn đá trong bát khác thì anh ta trở lại bát của mình và lấy đá của mình ra. Tiếp theo, anh ta nghỉ ngơi và đợi cho đến lúc cho cái bát kia trống rỗng và bỏ hòn đá vào bát của mình... Phương pháp này hoạt động tốt cho đến cuối tháng 5, khi hai chuyến tàu đồng thời bị chặn tại cửa khẩu.

## Thuật toán giải quyết bài toán

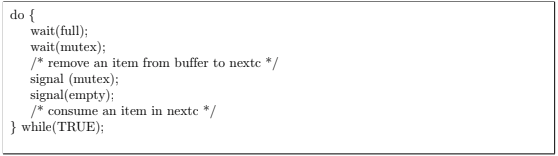
Sau đây chúng ta cùng tìm hiểu thuật toán của bài toán này sao cho hiệu quả nhất. Thực ra, ta thấy bài toán này hoàn toàn tương tự bài toán **Producer and Consumer**. Bài toán giải bằng phương pháp đèn báo với thuật toán như sau:

BINARY\_SEMAPHONE mutex=1  
COUNTIN\_SEMAPHORE empty = n; full=0;

**Producer:**



**Consumer:**



Ta xem **mutex** là đèn báo và biến **full** và **empty** để báo hiệu trạng thái hàng đợi là đầy hay rỗng.

# Vấn đề của người hút thuốc lá – Cigarette Smoker’s problem

## Bài toán

* Bài toán lần đầu được mô tả bởi Suhas Patil năm 1971.

Có 4 người trong vấn đề này: 3 người hút thuốc lá và một người đại lý. Mỗi người hút thuốc lá sẽ trải qua 2 quá trình là làm thuốc lá và hút thuốc. Để làm ra một điếu thuốc thì yêu cầu phải có thuốc lá (tobacco), giấy và diêm. Mỗi người hút thuốc lá có vô hạn 1 trong 3 cái: Thuốc lá, hoặc giấy hoặc diêm . Người đại lý có một nguồn vô tận của cả ba. Người đại lý sẽ chọn ngẫu nhiên 2 trong 3 món đồ đó đặt lên trên bàn. Còn người hút thuốc có thể lấy 2 cái trên bàn đó , với điều kiện là 2 cái đó khác với cái mình đang có để làm ra 1 điếu thuốc và sau đó hút.

Ở đây thì người đại lý bán thuốc đại diện cho hệ điều hành, còn người hút thuốc đại diện cho tiến trình. Người bán thuốc muốn phân bố các nguồn nhân lực cần thiết cho các quá trình và tránh trường hợp rơi vào bế tắc.

****

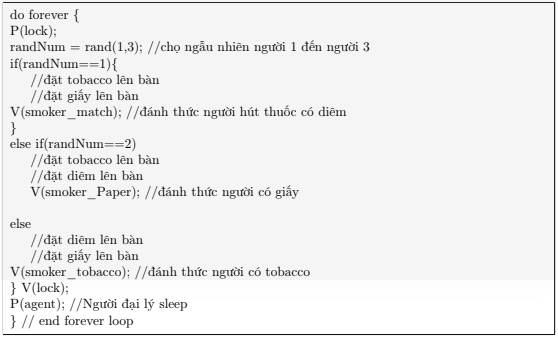
*Hình 2: Vấn đề của người hút thuốc lá*

Ví dụ: Người A có giấy, người B có thuốc lá, và người C có diêm. Nếu người đại lý đưa ra giấy và thuốc (tobacco) thì người A và B sẽ không tạo được thuốc lá do thiếu diêm.

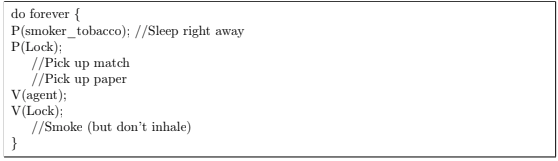
## Thuật toán giải quyết bài toán

Sau đây là một giải pháp khá dễ dàng . Cả 3 người sẽ làm được điếu thuốc lá và hút nó. Nếu người nào không làm được điếu thuốc thì sẽ ngồi nghỉ 1 chỗ nào đó (Sleep). Khi người đại lý bán hàng đặt hai vật lên trên bàn, và sẽ kêu (Wake up) người tương ứng có thể tạo ra thuốc để hút với hai nguyên liệu kia, sau đó thì đi ngồi nghỉ tiếp. Sau đây sẽ là thuật toán giải quyết vấn đề này:

* Thuật toán cho người đại lý bán hàng:



* Thuật toán cho người hút thuốc



Người hút thuốc ngay lập tức ngủ, khi người đại lý đưa 2 vật lên bàn, sau đó đại lý sẽ đánh thức người hút thuốc tương ứng (P.smocker\_tobacco). Người hút thuốc sau đó sẽchờ đến lượt truy cập vào tài nguyên găng lấy đồ vật và đánh thức các đại lý. Trong khi người hút thuốc lá đang hút, người đại lý có thể đặt hai vật lên bàn và đánh thức một người khác (Nếu vật người đó có không giống với 2 đồ vật trên bàn). Người đại lý ngủ ngay lập tức sau khi đặt các vật ra.

Giải thuật này giống với giải thuật bài toán Producer – Consumer. Ngoại trừ Producer chỉ có thể sản xuất 1 mặt hàng (ở đây là lựa chọn 3 mặt hàng – 3 hành động) cùng 1 lúc.

# Bài toán ông già Noel – Santa Claus Problem



*Hình 3: Bài toán ông già Noel*

## Bài toán

* Đây là một bài toán được đưa ra trong quyển Operation Systems của William Stallings. Nhưng được cho là của ông John Trono, ở trường đại học Michael ở Vermont.

Santa Claus đang ngủ ở trong cửa hàng của ông ấy ở Bắc Cực, ông ấy chỉ có thể được đánh thức bởi:

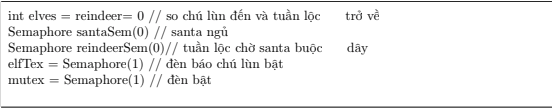
1. Cả 9 con tuần lộc trở về từ kì nghỉ hè ở Thái Bình Dương
2. 3 trong số các chú lùn gặp khó khăn trong việc làm đồ chơi đến nhờ ông già Noel giúp.  
   Một số lưu ý trong bài toán này:
   * Khi 3 chú lùn đang nhờ ông già Noel giúp, bất kì chú lùn nào đến gặp ông già Noel phải đợi những chú lùn kia trở về.
   * Nếu ông già Noel thức dậy nhìn thấy 3 chú lùn ở ngoài cửa cùng với con tuần lộc cuối cùng trở về từ vùng nhiệt đới thì những chú lùn có thể phải đợi đến sau giáng sinh, bởi vì việc quan trọng lúc này là chuẩn bị các xe trượt tuyết.
   * Giả thiết rằng các con Tuần Lộc không muốn rời khỏi vùng nhiệt đới, muốn ở lại đó đến giây phút cuối cùng có thể.
   * Con Tuần Lộc cuối cùng đến nơi phải đón ông già Noel trong khi những con khác chờ đợi trong một túp lều nóng lên trước khi được đưa vào xe trượt tuyết.

## Phân tích bài toán

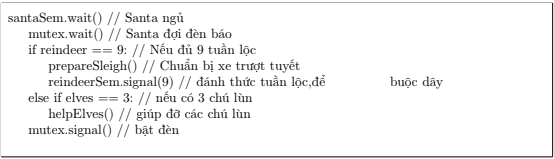
* Sau khi con tuần lộc thứ 9 đến, ông già noel phải chuẩn bị xe trượt tuyết (prepareSleigh) , và sau đó cả 9 con tuần lộc phải được buộc dây thừng (getHitched)
* Nếu trường hợp ông gìa Noel thức giấc chỉ có 3 chú lùn, ông già Noel phải giúp đỡ các chú lùn (helpElves), và điều đó cũng có nghĩa là các chú lùn được giúp đỡ (getHelp).
* Tất cả 3 chú lùn phải được giúp đỡ trước khi chú lùn nào khác được bổ sung.

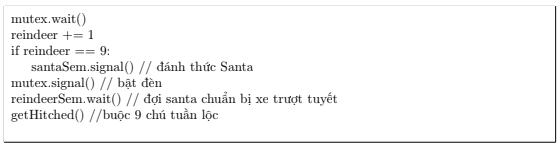
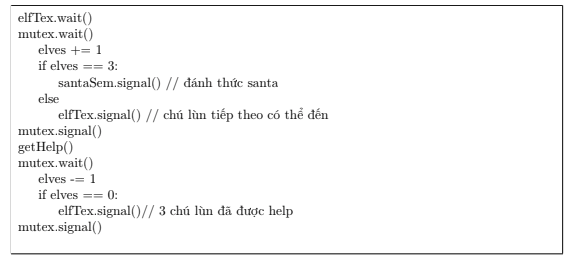
## Thuật toán giải quyết

**Khởi tạo:**



**Ông già Noel**



**Reindeer**  
  
**Elves**  


# Bài toán người trợ giảng ngủ gật – The Sleeping Teaching Assistant Problem

## Bài toán

Bài toán người trợ giảng ngủ gật là một biến thể của bài toán Người thợ cắt tóc ngủ gật, được viết trong quyển sách Operation Systems Concept của Silberschatz and Galvin. Bài toán như sau :

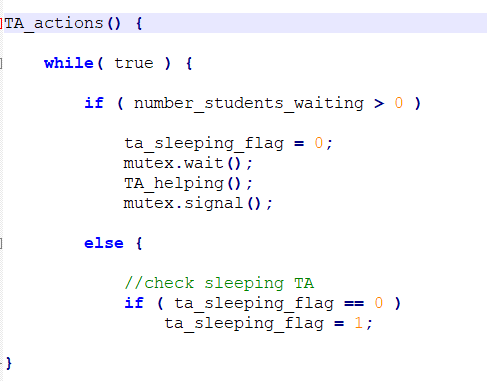
Một khoa Khoa học máy tính của một trường đại học có 1 người trợ giảng (Teaching Assistant) để giúp các sinh viên đại học trong những bài tập lập trình của họ trong suốt giờ hành chính thường ngày. Văn phòng của người trợ giảng khá nhỏ, chỉ có duy nhất 1 chiếc bàn với 1 ghế và máy vi tính. Bên ngoài hành lang, người ta đặt 3 chiếc ghế để sinh viên có thể ngồi và đợi nếu trong phòng người trợ giảng đang giúp 1 bạn sinh viên khác. Khi không có bạn sinh viên nào cần giúp đỡ trong giờ hành chính, người trợ giảng sẽ ngồi xuống chiếc ghế và chợp mắt 1 lúc. Nếu 1 sinh viên đến vào giờ hành chính và gặp tình huống người trợ giảng đang ngủ, câu ta phải đánh thức người trợ giảng kia để yêu cầu được giúp đỡ. Trong trường hợp sinh viên đến lúc người trợ giảng đang giúp đỡ sinh viên khác, cậu ta phải ngồi xuống 1 trong những chiếc ghế ngoài hành lang và chờ đợi. Nếu không có chiếc ghế nào trống cả, cậu sinh viên sẽ phải quay trở lại vào lần sau.

## Thuật toán giải quyết bài toán

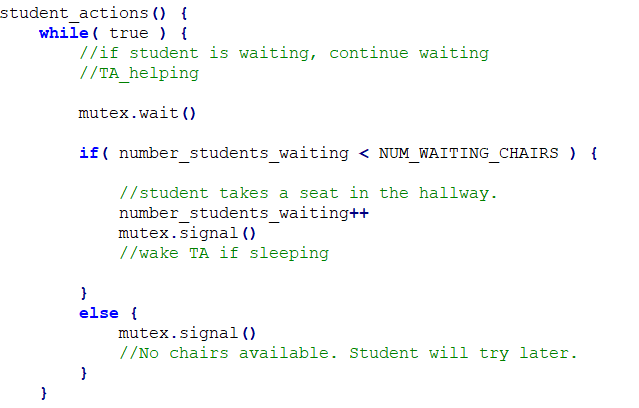
Bài toán sử dụng ký thuật đèn báo để giải quyết;

mutex = 1;

Thuật toán cho TA:



Thuật toán cho sinh viên:



# Bài toán vượt sông – River Crossing Problem

## Bài toán

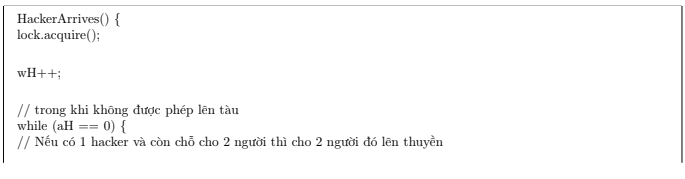
* Đây là vấn đề được viết bởi Anthony Joseph ở U.C.Berkeley . Nhưng không xác định chính xác ai là tác giả của bài toán này. Nó tương tự như vấn đề của bài toán tạo phần tử nước theo nghĩa nó là loại rào cản đặc biệt chỉ cho phép các luồng truyền qua các kết hợp nhất định.

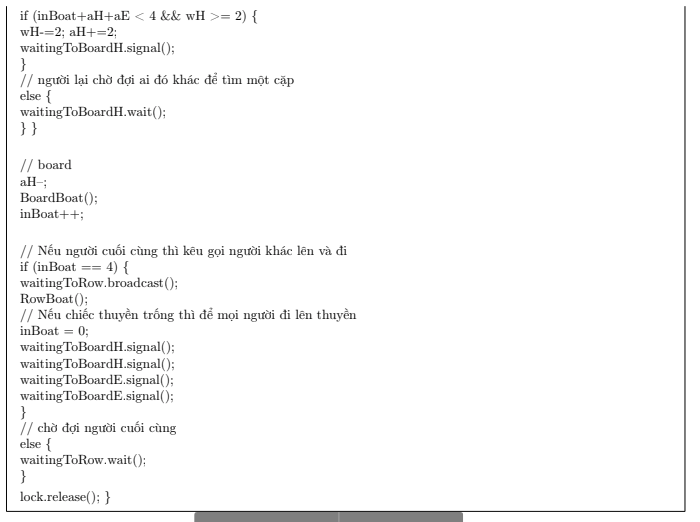
Bài toán như sau: Một nơi nào đó gần Redmond, Washington có một chiếc thuyền buồm chở hàng được sử dụng bởi cả tin tặc (Hacker) làm việc cho Linux và nhân viên của Microsoft để chuyển họ qua sông. Phà có thể giữ 4 người , nó sẽ không rời bờ với nhiều hoặc ít hơn số lượng vừa nói trên. Để đảm bảo an toàn cho khách hàng thì không cho phép đưa một hacker vào thuyền với ba hacker. Bất kỳ cách kết hợp khác là an toàn.

## Solution

1. Ở đây với mọi người lên thuyền mình sử dụng hàm BoardBoat() Các tình huống đó với 3 nhân viên và 1 hacker, hoặc ngược lại.
2. Khi chiếc thuyền chở đúng số người , thì mỗi người ở trên đó gọi là hàm RowBoat().
3. Chỉ một người có thể truy cập thuyền tại một thời điểm nhất định.
4. Hacker và nhân viên ở bên ngoài thuyền đợi, ta sẽ sử dụng các biến để biểu diễn các sự kiện này waitingToBoardH và waitingToBoardE. Người ở trong thuyền sẽ đợi tàu rời đi, ta sử dụng biến waitingToRow để biểu diễn sự kiện này.
5. Thứ nhất, cần phải biết số lượng hackers đang chờ đợi để được chỉ định cho một thuyền (wH) và số lượng nhân viên chờ đợi để được giao cho một thuyền (wE). Cũng cần phải biết số lượng tin tặc chưa lên tàu nhưng đã được chỉ định cho thuyền (aH) và số lượng nhân viên đó (aE). Cuối cùng, người cuối cùng trong thuyền cần phải xếp hàng , vì vậy cần phải biết số người trên thuyền (int inBoat). Tất cả các biến này đều được khởi tạo thành 0.

Thuật toán giải quyết:





# Bài toán tàu lượn siêu tốc – The Roller Coaster Problem

## Bài toán

* Bài toán này được trình bày trong quyển Concurrent Programming của Andrew, nhưng ông quy nó về những luẩn điểm của J. S. Herman’s Master.

Bài toán: Giả sử có n luồng hành khách và một luồng xe. Các hành khách liên tục chờ đợi để được đi trên xe, điều này có thể giữ C hành khách, trong đó C < n. Xe chỉ có thể đi quanh đường ray khi nó đầy.

Dưới đây là một số chi tiết bổ sung:

• Hành khách có 2 lựa chọn lên tàu hoặc không lên.

• Xe có các hành động tải, chạy và dỡ hàng.

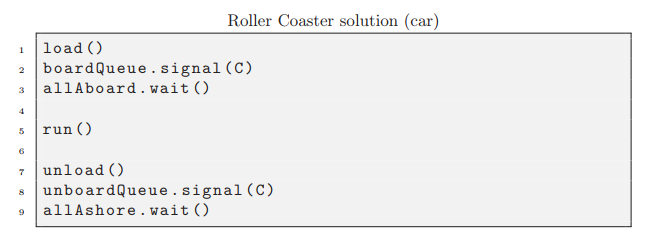
• Hành khách không thể lên xe cho đến khi xe đã đủ tải

• Xe không thể khởi hành cho đến khi C hành khách đã lên.

• Hành khách không thể dỡ hàng cho đến khi xe bắt đầu dỡ hàng.

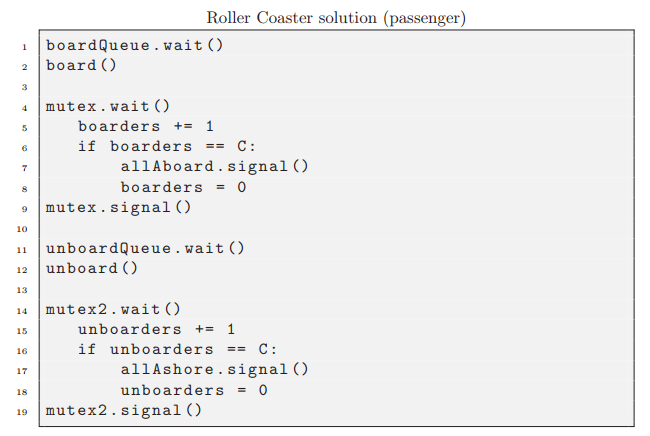
## Solution

Thuật toán cho tàu lượn:



Khi tàu đến, nó sẽ báo hiệu cho C hành khách, là đợi cho đến hành khách cuối cùng báo hiệu tín hiệu allBoard. Sau khi nó khởi hành, xe sẽ cho phép C hành khách xuống tàu và đợi tín hiện allAshore.

Thuật toán cho hành khách:



Hành khách sẽ đợi tàu trước khi lên theo 1 cách tự nhiên. Và đợi tàu dừng lại trước khi rời đi. Hành khác cuối cùng lên tàu và báo hiệu cho tàu và đặt lại biến đếm hành khách.

# Bài toán bữa tối của bộ tộc hoang dã – The Dining Savages Problem

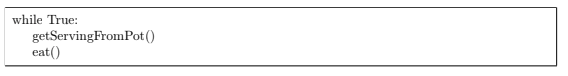
## Bài toán thực tế

* Bài toán này được trình bày trong quyển Concurrent Programming của Andrews.

Bài toán như sau: Một bộ tộc hoang dã ăn tối chung từ một cái nồi lớn có thể chứ được M khẩu phần ăn được nấu chín. Khi một người muốn ăn anh ấy sẽ tự lấy ở nồi nếu còn. Nếu nồi rỗng, người ấy đánh thức đầu bếp dậy và đợi cho đến khi đầu bếp cho thức ăn vào nồi.

## Solution

Tất cả các thành viên của bộ tộc đều sử dụng hàm lấy thức ăn:

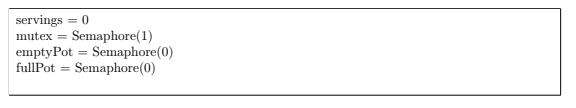


Một tiến trình nấu ăn sẽ chạy lệnh:

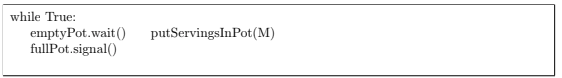


**Điều kiện ràng buộc giữa các tiến trình là:**

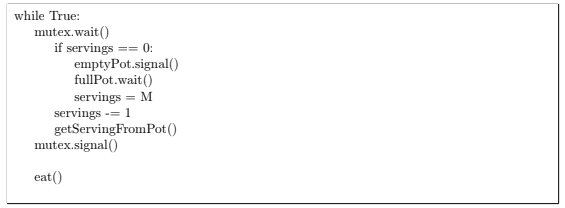
* Người không thể lấy thức ăn (chạy hàm getDervingFromPot()) nếu nồi rỗng.
* Đầu bếp chỉ có thể bỏ thêm thức ăn (chạy hàm putServingInPot())khi nồi rỗng. Chúng ta sử dụng biến đếm servings để đếm số lượng thức ăn còn lại trong nồi. Nếu servings =0 thì đầu bếp sẽ chạy hàm putServingInPot() . Đèn bào emptyPot để chỉ nồi rỗng và fullPot để chỉ nồi đã đầy.



Người nấu ăn:



Khí có tín hiệu báo nồi đã rỗng thì cook sẽ thực hiện hàm putServingsInPot() để đưa thêm thức ăn vào nồi. Khi nồi đầy sẽ chạy tiến trình fullPot.signal().// Savage (người trong bộ tộc)



Mỗi thành viên trong bộ tộc sẽ phải đợi đèn hiệu mutex để được lấy thức ăn. Nếu nhận thấy nồi rỗng thì sẽ đợi cho đến khi tín hiệu nồi đầy (fullPot) từ tiến trình Cook để thiết lập biến chỉ số lượng thức ăn trong nồi là servings = M và chạy tiếp chương trình. Mỗi tiến trình sẽ lấy một khẩu phần ăn từ nồi và giảm biến đếm đi 1 đơn vị . Sau khi lấy xong tiến trình này sẽ trả lại mutex cho các tiến trình đang đợi khác và thực hiện hàm eat().

# Tài liệu tham khảo

1. Operation System Process Synchronization, Jerry Breencher
2. A Little Book of Semaphores, Allen B. Downey, Version 2.2.1
3. Gagne, Silberschatz, and Galvin, Operating System Concepts (7th Edition)
4. On a solution to the cigarette smokers' problem, David Lorge - Parnas Carnegie Mellon University
5. Cigarette Smoker Problem, <https://www.youtube.com/watch?v=M_G6nDtQarg>
6. Satan Claus Problem, <https://www.youtube.com/watch?v=pqO6tKN2lc4>
7. Sleeping Teaching Assistant, <https://github.com/dpulsifer/Sleeping-TA-Problem>