CS240 Operating Systems, Concurrency and Communications

The Mutual Exclusion Problem

Recall - Thread Benefits

- Simple mechanism to add concurrency
- Can be added to an existing language
- Lightweight on Resources, Quick to Create

Need for coordination of thread activities

- Sequencing tasks between threads
- Coordinating access to resources

CS240 Operating Systems, Concurrency and Communications

Mutual Exclusion Problem Statement

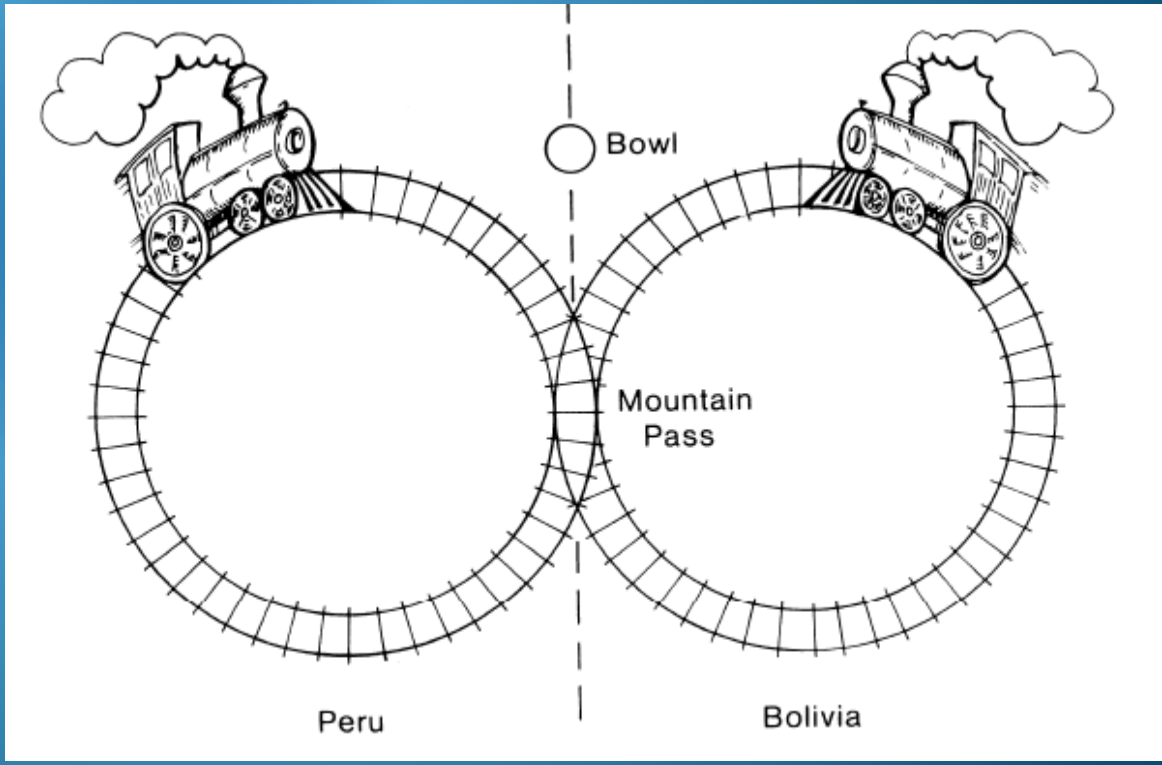
Consider a system of N cooperating threads where each thread might share a segment of code which requires exclusive access to complete.

No two threads must be able to execute this **critical section** of code at the same time.

CS240 Operating Systems, Concurrency and Communications

Train Drivers Problem

We use the following analogy to develop an understanding of what is needed to solve the mutual exclusion problem.



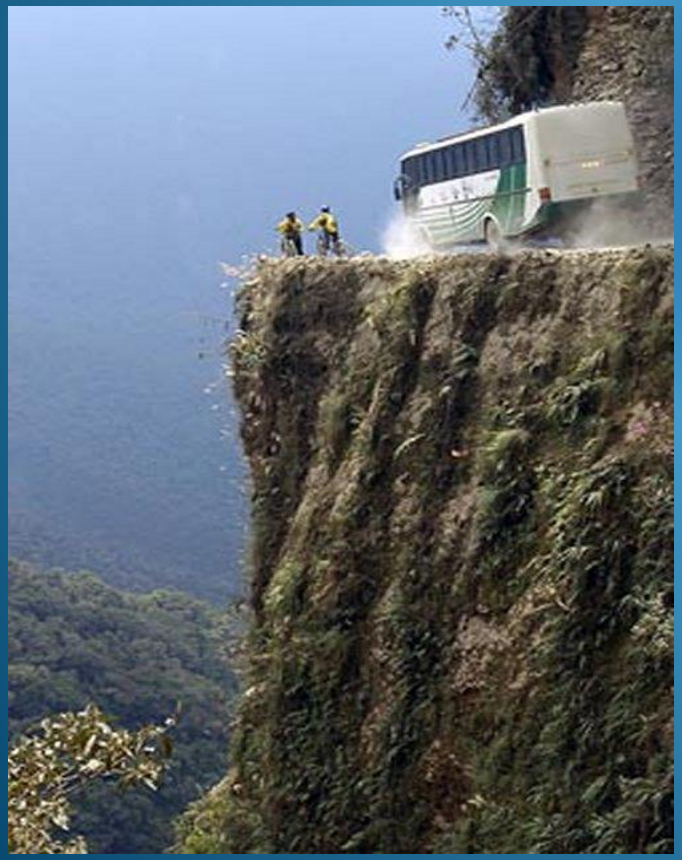
CS240 Operating Systems, Concurrency and Communications

Train Drivers Problem

Andes Mountains have one of the most important copper, tin, silver, lead, lithium and zinc mineral deposits in the world.

CS240 Operating Systems, Concurrency and Communications

Train Drivers Problem



The Road of Death

The **North Yungas Road** is a 69 kilometre stretch leading from La Paz to Coroico in Bolivia, popular with thrill seekers and cyclists.

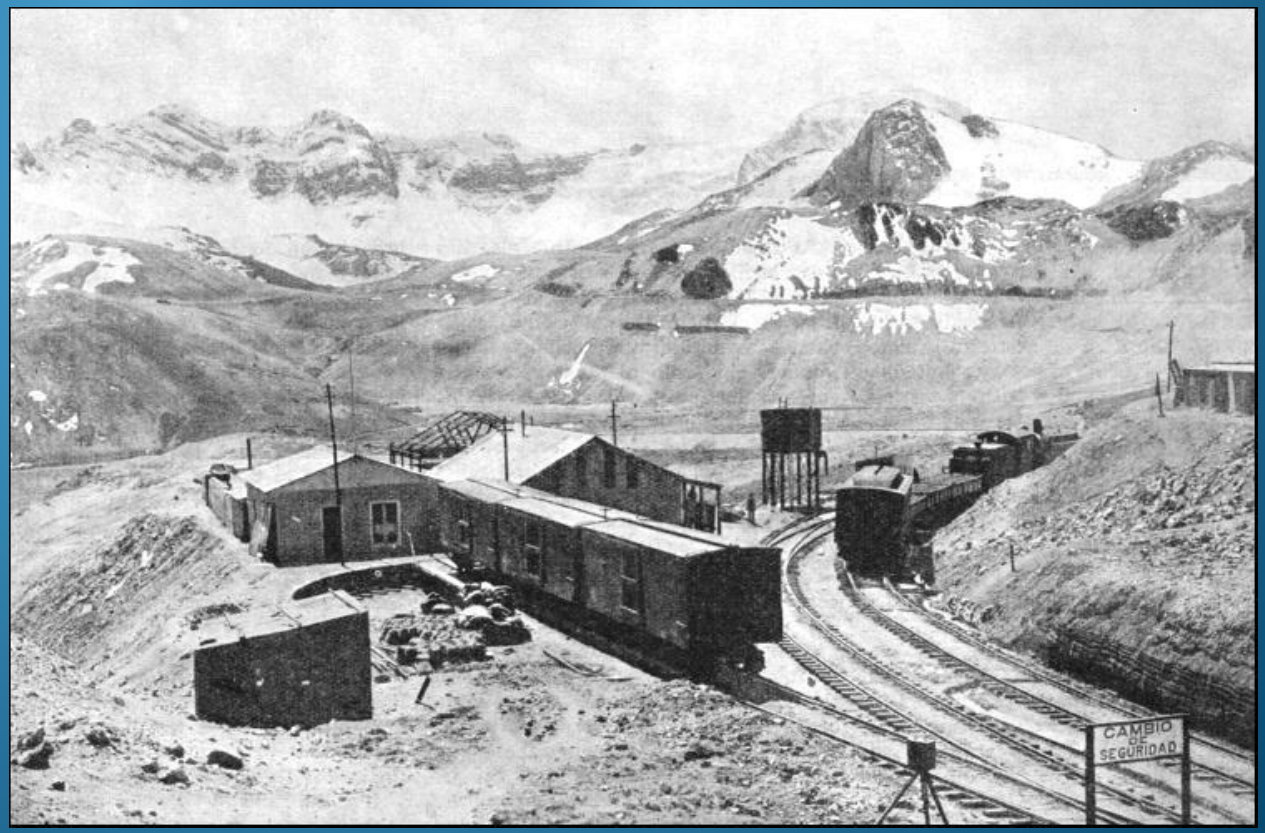
Top Gear on the Road of Death

CS240 Operating Systems, Concurrency and Communications

Train Drivers Problem

Ticlio in Peru.
Used to be the highest railway station in the world (15,610 ft) in the Andes Mountains.

Tanggula Railway Station opened in 2006 in China is 1,000ft higher.



CS240 Operating Systems, Concurrency and Communications

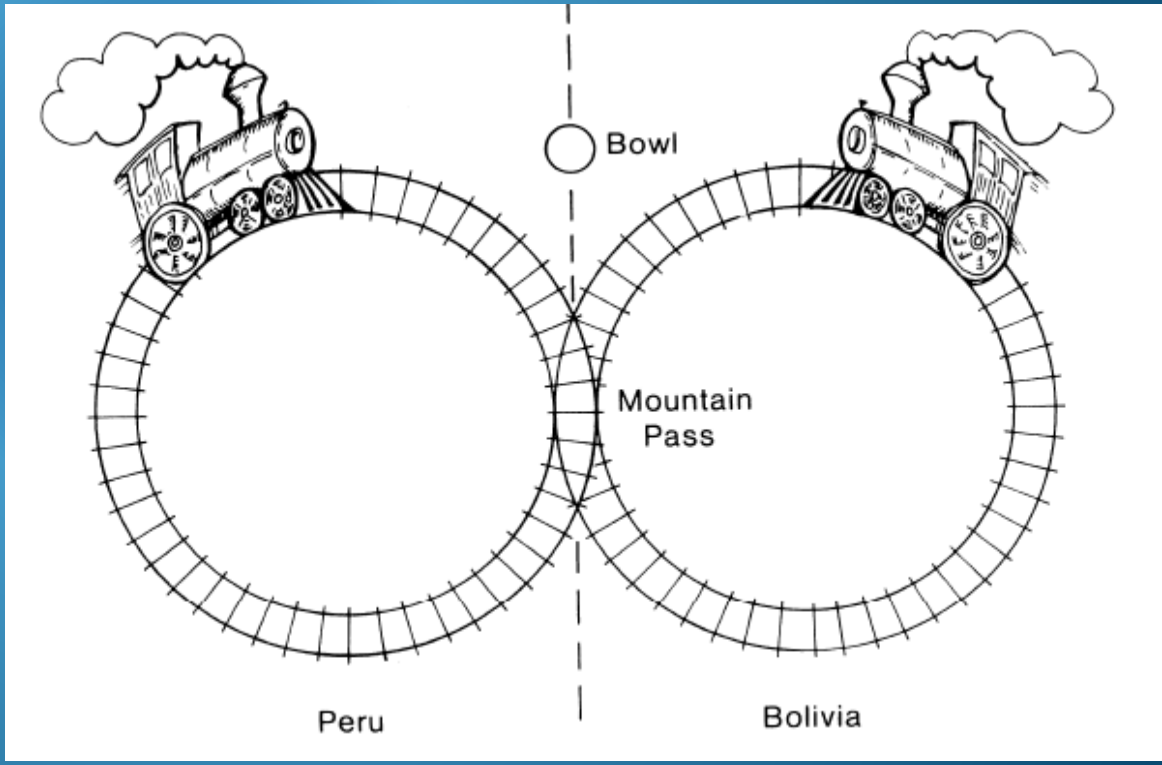
Train Drivers Problem

This is where our scene is set!

Oh yes, one more thing....

The Drivers are Blind and Deaf

(and we need a method of preventing collisions).



CS240 Operating Systems, Concurrency and Communications

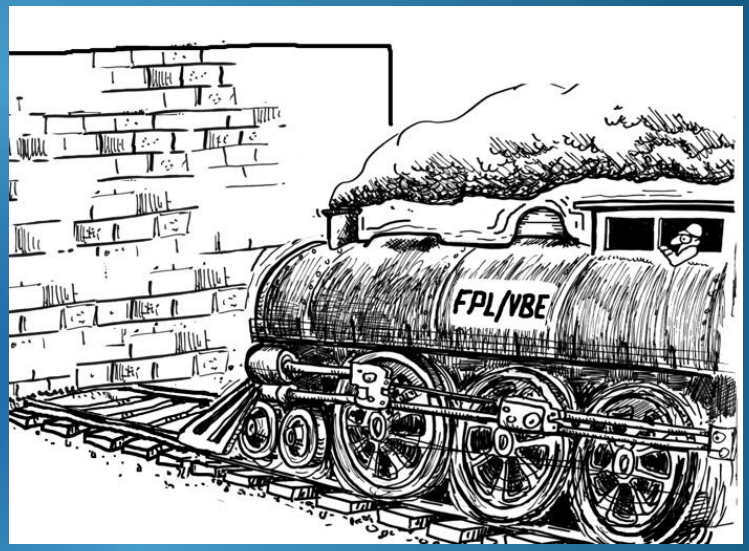
Train Drivers Problem - Method 1

They set up a bowl at the entrance to the pass. Before entering the pass, a driver must stop his train, walk over to a bowl and reach into it to see if it contains a rock. If the bowl is empty, he searches around for a rock and drops it in the bowl, indicating that his train is using the pass, then he drives the train into the pass.

If a driver arriving at the pass finds a rock in the bowl, he leaves it there. He repeatedly takes a siesta and rechecks the bowl until he finds the bowl empty.

When the train has cleared the pass, he must walk back to the bowl and remove the rock.

CS240 Operating Systems, Concurrency and Communications



Train Drivers Problem - Method 1

Problems

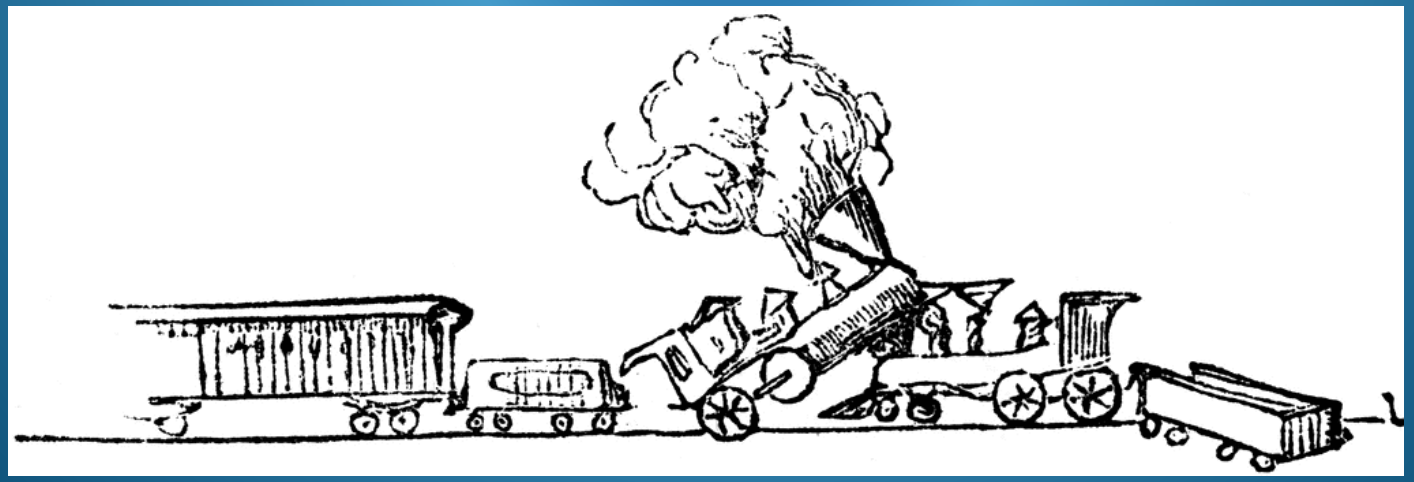
1. It is claimed that subversive schedules made up by the Peruvian officials could block the Bolivian train forever. Explain.

CS240 Operating Systems, Concurrency and Communications

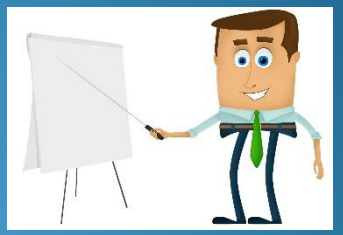
Train Drivers Problem - Method 1

Problems

2. Unfortunately, one day the two trains crashed. Explain.



CS240 Operating Systems, Concurrency and Communications



Train Drivers Problem - Method 2

Following the collision, a consultant was called in who concluded that the bowl was being used in the wrong way.

The Bolivian driver must wait at entry to the pass until the bowl is empty, then drives through the pass and walks back and puts a rock in the bowl. The Peruvian driver must wait at the entry until the bowl contains a rock, then he drives through the pass and walks back to remove the rock from the bowl.

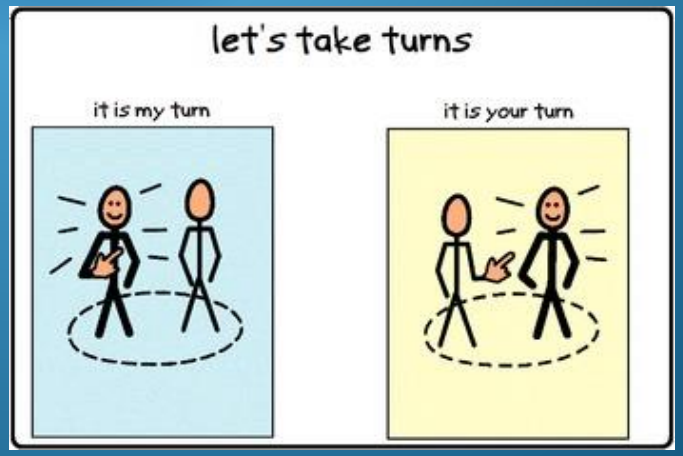
Sure enough, there were no more crashes.

CS240 Operating Systems, Concurrency and Communications

Train Drivers Problem - Method 2

Problem

Prior to this the Peruvian Train ran twice a day and the Bolivian Train ran once a day. The Peruvians were very unhappy with the new arrangement. Why?



CS240 Operating Systems, Concurrency and Communications

Train Drivers Problem - Method 3

The consultant was called back to come up with a solution that solved the Peruvian's problem as well as preventing crashes. He suggested using two bowls, one for each driver.

When a driver wishes to enter, he first drops a rock in his own bowl first as a signal that he wishes to use the pass. He then checks the other driver's bowl to see if it is empty. If it is, he drives his train through, walks back and removes the rock from his own bowl.

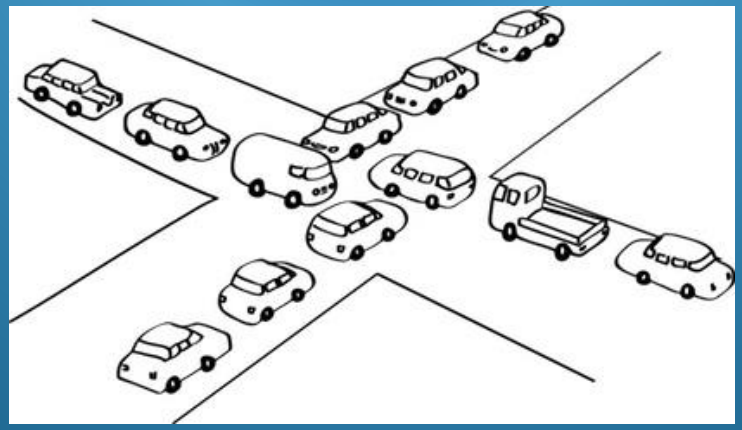
If the other driver's bowl does contain a rock however, then he lets that driver go through the pass first by taking a siesta and retrying the other driver's bowl periodically until the other driver removes the rock after he clears the pass.

CS240 Operating Systems, Concurrency and Communications

Train Drivers Problem - Method 3

Problem

The solution worked for a long time, until one day both trains were simultaneously blocked at the entry. Why?



CS240 Operating Systems, Concurrency and Communications

Train Drivers Problem - Method 4

(Solution)

A single bowl is used, but initially it contains a rock. If either driver finds a rock in the bowl, he removes it and drives through the pass. He then walks back to put a rock in the bowl. If a driver finds no rock in the bowl when he wishes to enter the pass, he waits.

Why does this solution work? How does it differ from Method 1?

CS240 Operating Systems, Concurrency and Communications

In Software, **Method 1** is equivalent to both threads executing the following code:-

```
int flag = 0;
```
Empty Bowl

```
/* Shared variable between the threads
   Initially 0   */
```

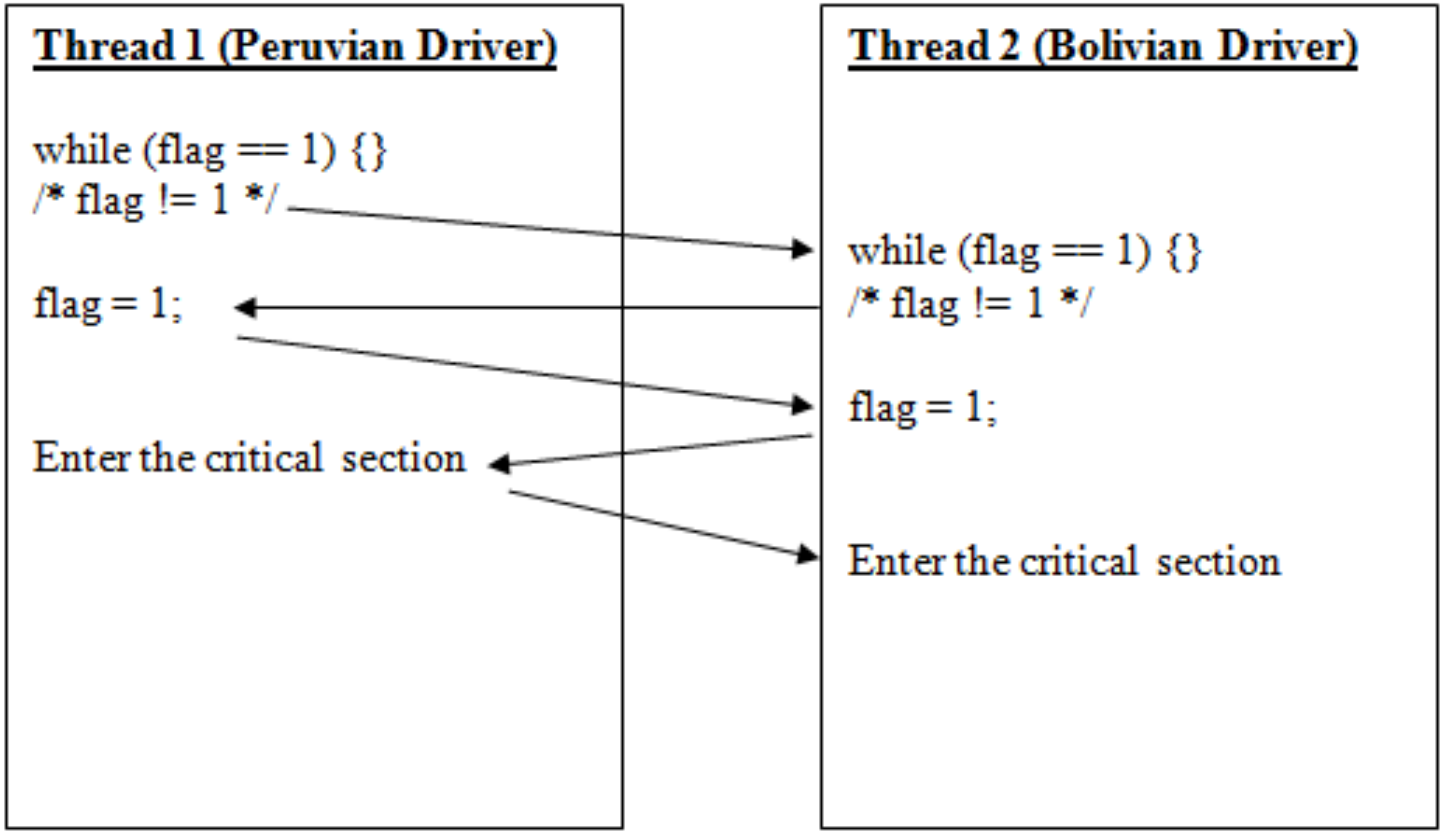
While bowl contains a rock, do nothing

```
while (flag == 1) {}  /*Do nothing while flag=1 */

flag = 1;
Enter the Pass
...
Leave the Pass
flag = 0;
```

CS240 Operating Systems, Concurrency and Communications

The crash occurs whenever an interleaved execution of the threads, similar to that below occurs.



CS240 Operating Systems, Concurrency and Communications

Method 2 is equivalent to:-

```
int turn = 0;
/* Shared variable between the processes,
initially 0 */
```

```
Peruvian Process
while (turn == 0) {}
Enter Pass
...
Leave Pass
turn = 0;
```

```
Bolivian Process
while (turn == 1) {}
Enter Pass
...
Leave Pass
turn = 1;
```

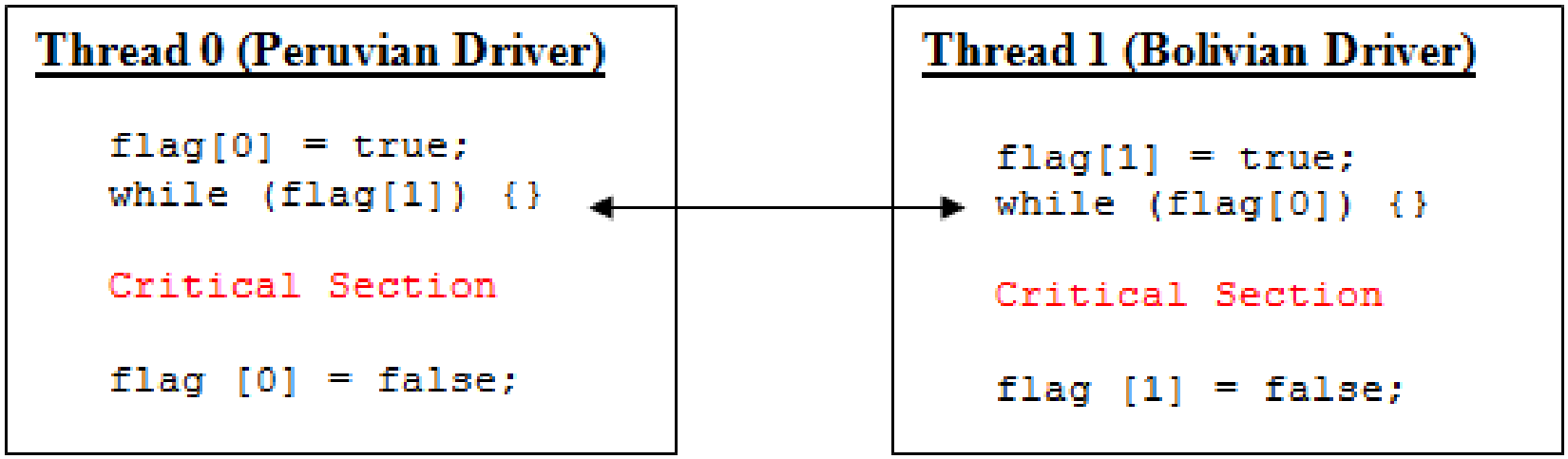
While mutual exclusion is not violated, the execution is lock step.

CS240 Operating Systems, Concurrency and Communications

Method 3

The two bowls are represented as elements of a boolean array, both are intially empty.

```
/* Shared variable between the threads*/
boolean[] flag = {false, false};
```



A deadlock occurs if both threads set their flags simultanously, neither can proceed as it is waiting on the action of the other.

CS240 Operating Systems, Concurrency and Communications

Method 4 is equivalent to both processes executing the following code:-

Bowl contains a rock to begin with

```
int flag = 1;
/* Shared variable between the processes
initially 1 */
```

While bowl is empty, do nothing

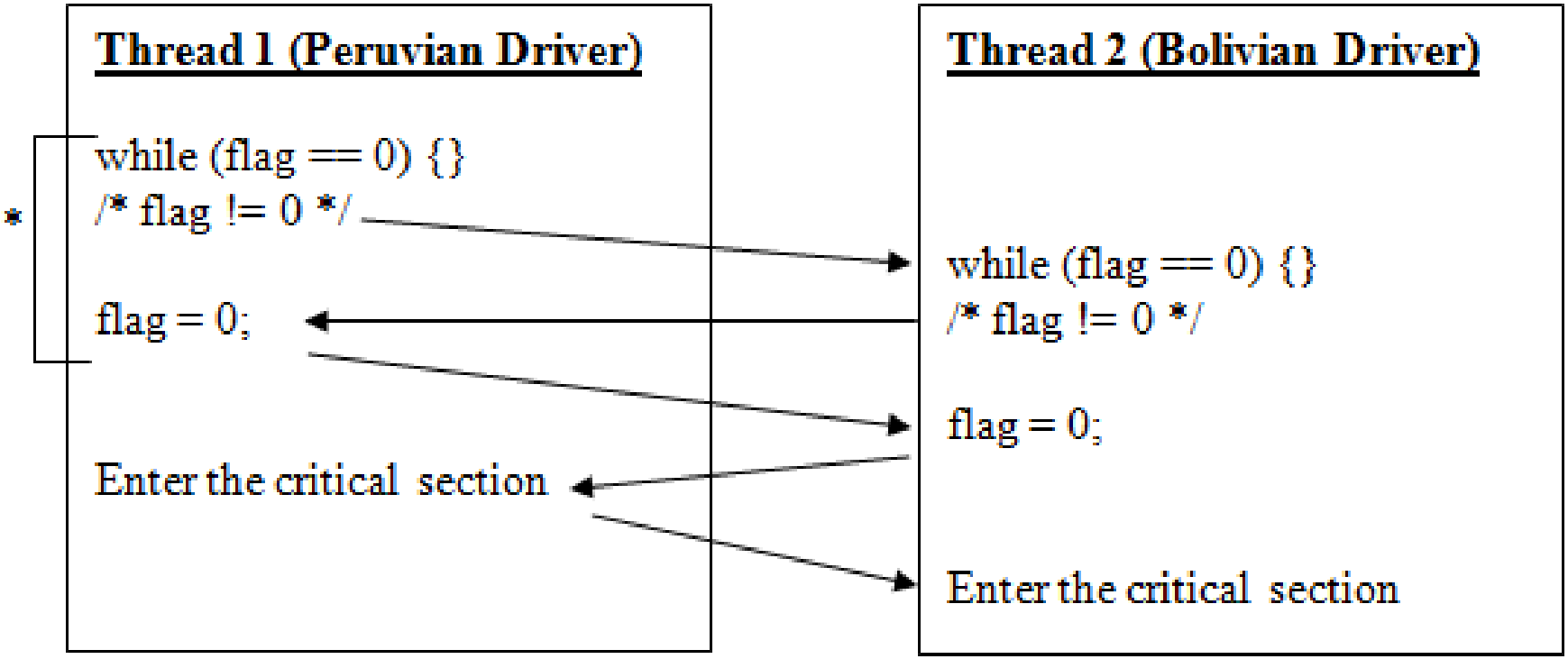
```
while (flag == 0) {}/*Do nothing while Flag=1*/
```

Remove the rock to use the pass

```
flag = 0;
Enter the Pass
...
Leave the Pass
flag = 1;
```

CS240 Operating Systems, Concurrency and Communications

Note that in software we could have the following interleaved sequence of execution which would violate the mutual exclusion requirement.



CS240 Operating Systems, Concurrency and Communications

Method 4 - This method worked for the train drivers. Why doesn't it work in software?
The reason is that removing a rock from the bowl was an indivisible operation. Only one driver could find AND take the rock at a time. In software however, our algorithm separately tests Flag and then sets it.

For the solution to solve the mutual exclusion problem, the statements marked by an asterisk in the figure would have to be indivisible. That is, somehow they must be executed completely or not at all, but interleaved execution with another process must be prevented.