Problem Solving Session

Semaphores

Prepared by Dhruv Singhal

Recap: Semaphore

Semaphore is a data structure that encapsulates an integer.

It has two operations P and V.

P decrements the integer and V increments it.

From the user's perspective, the integer is never allowed to become negative.

Attempting to decrement below 0 will block the running thread until another thread increments the count.

The Senate Bus Problem

Riders come to a bus stop and wait for a bus.

When the bus arrives, all the waiting riders invoke `boardBus`, but anyone who arrives while the bus is boarding has to wait for the next bus.

The capacity of the bus is 50 people; if there are more than 50 people waiting, some will have to wait for the next bus.

When all the waiting riders have boarded, the bus can invoke `depart`.

If the bus arrives when there are no riders, it should depart immediately.

Write synchronization code that enforces all of these constraints.

Solution 1

```
#-- Initialization ------------------
riders = 0
mutex = Semaphore(1)
multiplex = Semaphore(50)
bus = Semaphore(0)
allAboard = Semaphore(0)

#-- Bus -----------------------------
mutex.P()
    if riders > 0:
        bus.V()
        allAboard.P()
mutex.V()
depart()
```

```
#-- Riders --------------------------
multiplex.P()
    mutex.P()
        riders += 1
    mutex.V()
    bus.P()
multiplex.V()
boardBus()
riders -= 1
if riders == 0:
    allAboard.V()
else:
    bus.V()
```

Solution 2

```
#-- Initialization -------------------
waiting = 0
mutex = new Semaphore(1)
bus = new Semaphore(0)
boarded = new Semaphore(0)

#-- Riders ---------------------------
mutex.P()
     waiting += 1
mutex.V()
bus.P()
board()
boarded.V()
```

```
#-- Bus ----------------------------
mutex.P()
     n = min(waiting, 50)
     for i in range(n):
          bus.V()
          boarded.P()
     waiting = max(waiting-50, 0)
mutex.V()
depart()
```