

## The Monkeys and Bananas Problem

Consider the following synchronization problem. A boat delivers bananas to a desert island which is inhabited only by monkeys. The boat delivers one load of bananas at a time, but the load has a variable number of bananas in it; when it arrives, it calls the `deliver(int nbananas)` function, where `nbananas` specifies how many are delivered (at least one is always delivered). Bananas are placed in a crate on the beach.

When a monkey is hungry, it goes to the crate. If there are enough bananas in the crate, it takes what it needs; otherwise, it joins a queue of monkeys waiting for the boat. The monkeys are greedy and are not always satisfied with just one banana. A monkey wanting to get `nbananas` bananas calls `monkey(int nbananas)`. Calling `monkey` should cause it to wait in the queue until it gets `nbananas` bananas, then leave. A monkey will stay at the head of the queue until it has all the bananas it wants, rather than, for instance, grabbing 3 out of 4 bananas and then going to the back of the queue to wait for its final banana.

Devise a solution for the monkeys-and-bananas problem which follows the rules given above, and incorporates correct concurrency control using only semaphores.

We can use a semaphore as a counter to count the number of bananas. The head monkey needs to be treated specially, since it has to stay at the head of the queue until it gets all the bananas it wants. Therefore we make the head monkey wait on the bananas semaphore and create a separate semaphore for the rest of the monkeys, which we call `queue`. The head monkey signals `queue` when it has all the bananas it wants, as long as at least one monkey is waiting in the queue. Since we cannot determine the `queue` value ourselves, we need a separate variable to count how many monkeys are waiting by the crate, and we protect this variable with the semaphore `mutex`. Here's a solution:

```
sem bananas = 0; // could initialise to any value
sem queue = 0; // all monkeys except head monkey wait here
int monkeys = 0;
sem mutex; // to protect "monkeys"

// put "nbananas" bananas in the crate
void deliver(int nbananas) {
    for (int i=0; i<nbananas; i++)
        signal(bananas);
}

// monkey wants "nbananas" bananas
void monkey(int nbananas) {
    wait(mutex);
    monkeys++;
    if (monkeys > 1) { // i.e. there was already a head monkey
        signal(mutex);
        wait(queue);
    }
    else
        signal(mutex);
    // now I am the head monkey
    for (int i=0; i<nbananas; i++)
        wait(bananas);
    // wake up the next monkey
    wait(mutex);
    monkeys--;
    if (monkeys > 0)
        signal(queue);
    signal(mutex);
}
```