Little Book of Semaphores, Chapter 2

Geoffrey Matthews
Western Washington University

January 11, 2016

Semaphores

- An integer variable, initialized to any value.
- Only **increment** and **decrement** available.
- When a thread decrements, if the result is negative the thread blocks.
- When a thread increments, if there are threads waiting, one gets unblocked.
- Increment and decrement are atomic.

Semaphores

- An integer variable, initialized to any value.
- Only **increment** and **decrement** available.
- When a thread decrements, if the result is negative the thread blocks.
- When a thread increments, if there are threads waiting, one gets unblocked.
- Increment and decrement are atomic.
- No getter. Why?

Notation for Semaphors

Decrement	Increment
P	V
Wait	Signal
Wait	Post
Acquire	Release
Get	Release

Semaphore notation in cleaned up Python and Racket

```
fred = Semaphore(1)
fred.signal()
fred.wait()
```

```
(define fred (make-semaphore 1))
(semaphore-post fred)
(semaphore-wait fred)
```

Remarks

- There is no way to know before decrementing if a thread will block.
- After a thread increments a semaphore with waiting threads, there will be at least two active threads. There is no way to know which one will continue immediately.
- When you signal a semaphore, there is no way to know if threads are waiting.

Why semaphores?

- Semaphores impose constraints that help programmers avoid errors.
- Solutions using semaphores are often clean and organized.
- Semaphores can be implemented easily in hardware, so solutions are portable.

Why semaphores?

- Semaphores impose constraints that help programmers avoid errors.
- Solutions using semaphores are often clean and organized.
- Semaphores can be implemented easily in hardware, so solutions are portable.
- However, they can get complex quickly.