Some Classic Concurrency Problems

Reusable Barriers

Joseph Kehoe[1]

[1] Department of Computing and Networking
Institute of Technology Carlow

CDD101, 2017

The Problem

- The previous barrier solution will not work in a loop
  - Why?
- How do we solve this problem?
- We need a reusable barrier that locks itself after all the threads have passed through
- Try it now!

Why is this a bad solution?

```
mutex.wait ()
    count += 1
mutex.signal ()

if count == n : turnstile.signal ()
turnstile.wait ()
turnstile.signal ()
//CRITICAL POINT

mutex.wait ()
    count -= 1
mutex.signal ()
if count == 0: turnstile.wait ()
```

Barrier Hint

```
turnstile = Semaphore (0)
turnstile2 = Semaphore (1)
mutex = Semaphore (1)
```

- Use two turnstiles
- One turnstile is always locked!
- Sometimes called a two phase barrier