Deadlock – Dining Philosophers

SWEN-220
When using shared, mutable resources, there are several access properties your system should exhibit.

• A **shared, mutable resource** (SMR) could be a shared mutable variable, or a device such as a communication channel, disk, or printer.

• Safety (job #1):
  Mutually exclusive access to shared, mutable resource (SMR)

• Liveness 1:
  If threads are trying to access an SMR, one eventually does.

• Liveness 2:
  A thread holding an SMR eventually releases it.

• Fairness (no starvation):
  If a thread is trying to access an SMR, it eventually gains access.
The classic Dining Philosophers can deadlock and leave the philosophers hungry.

- Informally – a set of threads blocked with no possibility of progress.
- Formally – a set of threads, each holding an SMR needed by another thread in the set and waiting to acquire a resource which is already held.
- Classic example: Dining Philosophers

What is a path to deadlock?

- Naïve Approach
  - Get right fork
  - Get left fork
  - Eat
The Path to Deadlock

1. Everyone attempts to start eating by reaching for the fork to their right.
   - P0 -> Fork 0
   - P1 -> Fork 1
   - P2 -> Fork 2
   - P3 -> Fork 3

2. Assuming they are all successful, they then attempt to get the fork to their left.

3. All the “left side” forks are taken by the Philosopher to their left, so they wait.

4. Since no one will put their right fork down until they get the left and finish eating – they will wait forever (starve).

DEADLOCK!
There are **four necessary and sufficient conditions** for deadlock to be possible.

- **Necessary** means all must hold for *deadlock to be possible*.
- **Sufficient** means if all hold *deadlock is possible*.

The four necessary and sufficient conditions for deadlock to be possible are

- Exclusive use of resources
  *(A philosopher requires both forks to eat)*
- No preemption of resource hold
  *(A philosopher is not prompted to put down his fork)*
- Serial acquisition of resources
  *(Forks are sequentially picked up by a philosopher)*
- Cyclic hold-and-wait graph or cyclic dependency
  *(If all philosophers have one fork, they are holding & waiting)*

- Having these four conditions guarantees that deadlock is possible. It does not guarantee that it will happen.
  - Do you want to trust your system with “it may not happen”?