Operating System Concepts Weekly

Week 5

Lucas van der Laan
s1047485

1

- The sem sempahore is for the shelves and the items semaphore is for after something has been added to the shelves.
- If the customer executes first, we already have a deadlock because the baker can never add something to the shelf. If the baker executes first, we first signal to the items instead of first signaling to sem and if we have a binary semaphore, this will make the customer still wait on items.wait();
- The improvement I would make is that we do sem signal before we do items signal in the baker, this will no longer deadlock with binary semaphores.

Dotted lines are waits, filled lines are signals.

$$\begin{tikzcd}
& C \arrow[dl, bend right, "4"'] & \\
\boxed{S} \arrow[ur, dashed, "1"] \arrow[dr, dashed, "3"] & & \boxed{I} \arrow[ul, dashed, "2"'] \\
& B \arrow[ul, bend left, "6"] \arrow[ur, "5"'] &
\end{tikzcd}$$

2

- There is no deadlock possible, as they will more or less alternate between each other. When, for example, P0 signals f0 there is a chance of it going to P2 or P0, if it goes to P2, P1 and P2 will then fight for f1. If P2 gets f1, it will then eat and signal f1 and f0, P0 and P2 will then fight for f0 again. If P2 gets f0, P1 and P2 will fight for f1, if P0 gets f0, P1 will go further with f1, signal f1 and then signal f2 for a fight between P1 and P0 again. There is a constant struggle for who gets the fork, but not a single fight leads to a deadlock.

- If we swap the waits in P0 and P1 we can get a deadlock. This is because they will all be waiting: P0 on f0, P1 on f2, P2 on f1.
- No, it doesn't matter in which order the forks are released, because the fork will be released at some point. As long as the fork is released, a fight can be had about the fork again.

3

```
void smokerA() {
  while (true) {
    agentSem.signal();
    paper.wait();
    tobacco.wait();
  }
}

void smokerB() {
  while (true) {
    agentSem.signal();
    tobacco.wait();
    match.wait();
  }
}

void smokerC() {
  while (true) {
    agentSem.signal();
    match.wait();
    paper.wait();
  }
}
```

No deadlocks.