Classic IPC Problems

2.4 in Tanenbaum, 2nd ed.

Dining Philosophers Problem

Classic problem to demonstrate both deadlock and starvation.

Dijkstra (1965) posed and solved this problem.

Five philosophers seated around a table with five chopsticks and five rice bowls. Need two chopsticks to eat rice. Philosophers alternate between thinking and eating. Want to grant chopsticks while avoiding both deadlock and starvation.

What happens with the following straightforward solution:

```
Philospher(which)
int which;        /* index of philospher */
Resource LeftChopstick, RightChopstick;
{
    while (1) {
        Think;        /* for some amount of time */
        GetResource(LeftChopstick);
        GetResource(RightChopstick);
        Eat;
        ReleaseResource(LeftChopstick);
        ReleaseResource(RightChopstick);
    }
}
```

Compare Italian vs. Chinese version (fork/plate vs. chopsticks/rice bowl)

Solutions:

- if can't get right chopstick then put it back and try again (could get in lock step)
- a single semaphore that only one philospher can eat at a time (serialization)
- acquire both chopsticks at once (both or none)—could lead to starvation. Philosophers 1 and 3 might alternate between eating and thinking, but if they never think at the same time, philosopher 2 goes hungry.
- Fig 2-20 (Tanenbaum solution)

The Readers and Writers Problem

Disk file, airline reservation system—multiple processes reading and writing.

How to control access.

Can put a mutex around the resource (one read or write at a time). But multiple reads are ok. A solution?

Fig. 2-34, although the writer could starve if readers keep arriving.

The Sleeping Barber Problem

customers arrive to a barber, if there are no customers the barber sleeps in his chair.

If the barber is asleep then the customers must wake him up.