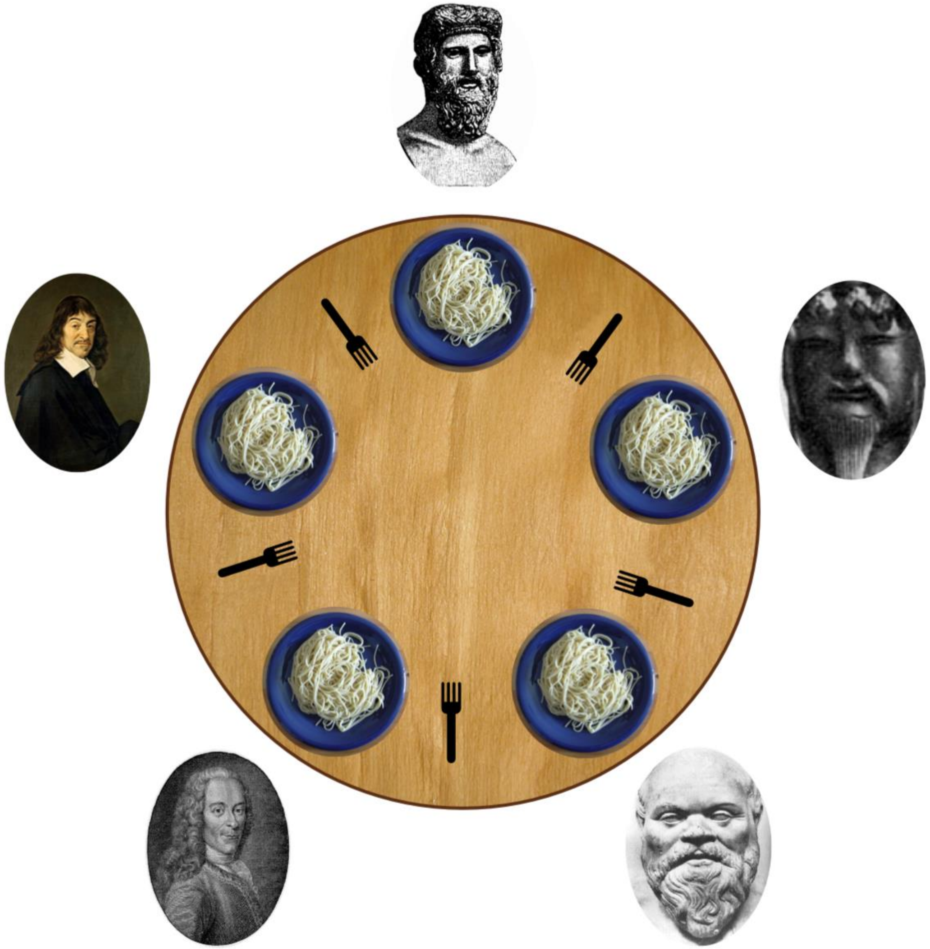
Part 9: The Dining Philosophers Problem

A famous concurrency problem



The Setup

Crux of the problem: Devise a solution such that:

➢

➢

➢

A Naïve Solution (High-level)

```
while (1) {
        while (left fork not available)
                think();
        pick-up(left-fork);

        while (right fork not available)
                think();
        pick-up(right-fork);

        eat();

        put-down(right-fork);
        put-down(left-fork);
}
```

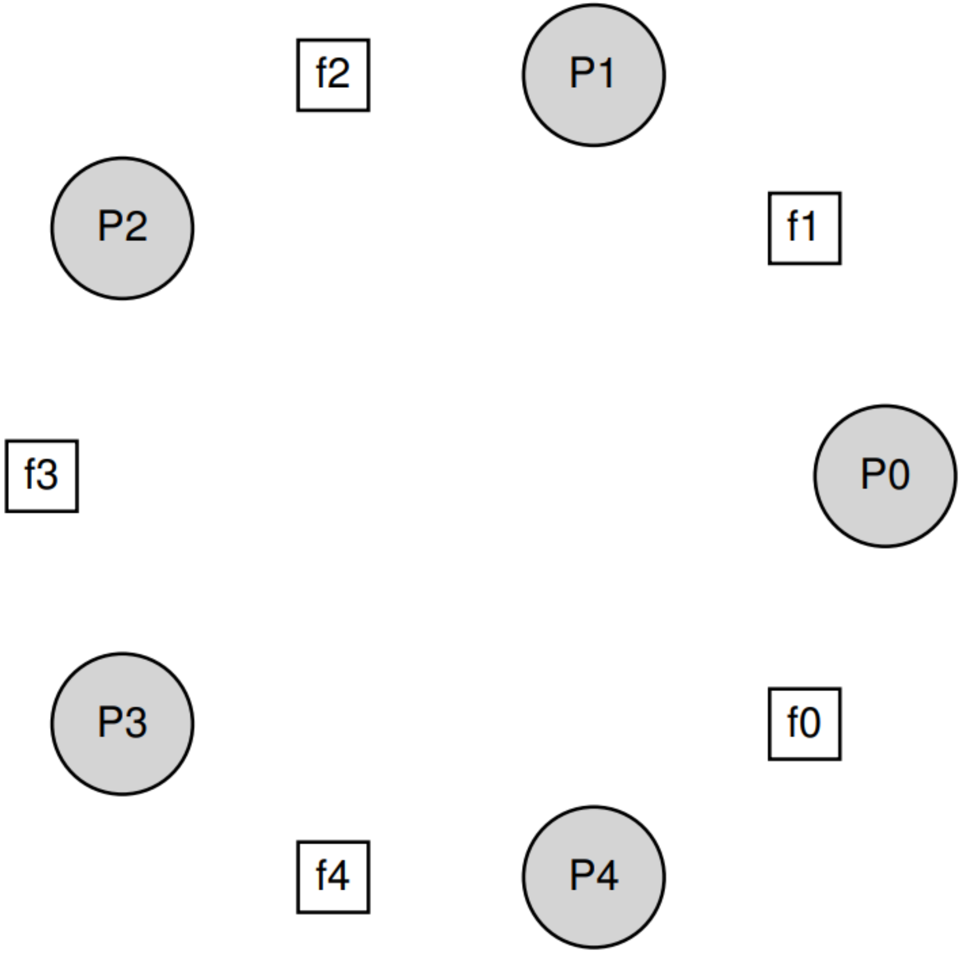
A Naïve Solution (Code)

```
while (1) {
        lock(left-fork);
        lock(right-fork);

        eat();

        unlock(right-fork);
        unlock(left-fork);
}
```

Circular Wait



Conditions for Deadlock

1. Mutual Exclusion
2. Hold-and-Wait
3. No Pre-emption
4. Circular Wait

Three Methods of Attack

1. Deadlock Prevention

   ➢

2. Deadlock Avoidance

   ➢

3. Deadlock Detection & Recovery

   ➢

Deadlock Prevention: Remove One

1. Mutual Exclusion
2. Hold-and-Wait
3. No Pre-emption
4. Circular Wait

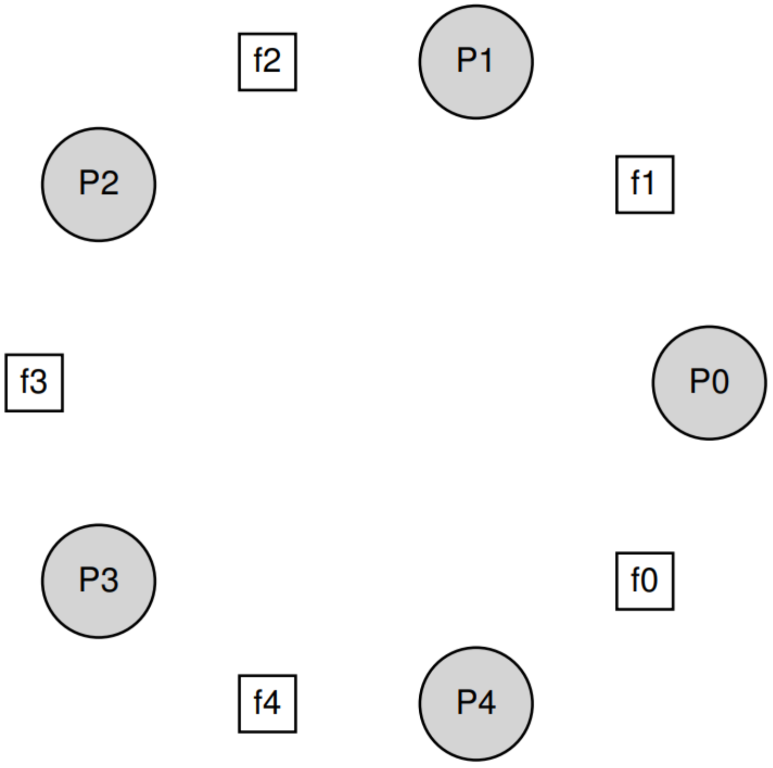
Hold-And-Wait (the “Waiter” solution)

```
while (1) {
        lock(waiter);
        lock(left-fork);
        lock(right-fork);
        unlock(waiter);


        eat();

        unlock(right-fork);
        unlock(left-fork);
}
```

Circular Wait ("Numbered Forks")



Numbered Forks Code

```
while (1) {
        // philosopher 4 goes right-left
        if (me == 4) {
                lock(right-fork);
                lock(left-fork);
        } else { // everyone else is left-right
                lock(left-fork);
                lock(right-fork);
        }

        eat();

        unlock(right-fork);
        unlock(left-fork);
}
```

Part 10: Linux

How does Linux handle all of this?

Processes vs Threads

In general/in theory…

Process: A unit of protection/isolation

Thread: A point of execution/what is scheduled on the CPU

In Linux? Tasks.

➢

➢

➢

Tasks Have…

Tasks *combine* elements of “process” and “thread”

- State
- Various IDs
- IPC
- Scheduling information
- Virtual address space
- Registers & stack information
- …etc…

Task State Diagram

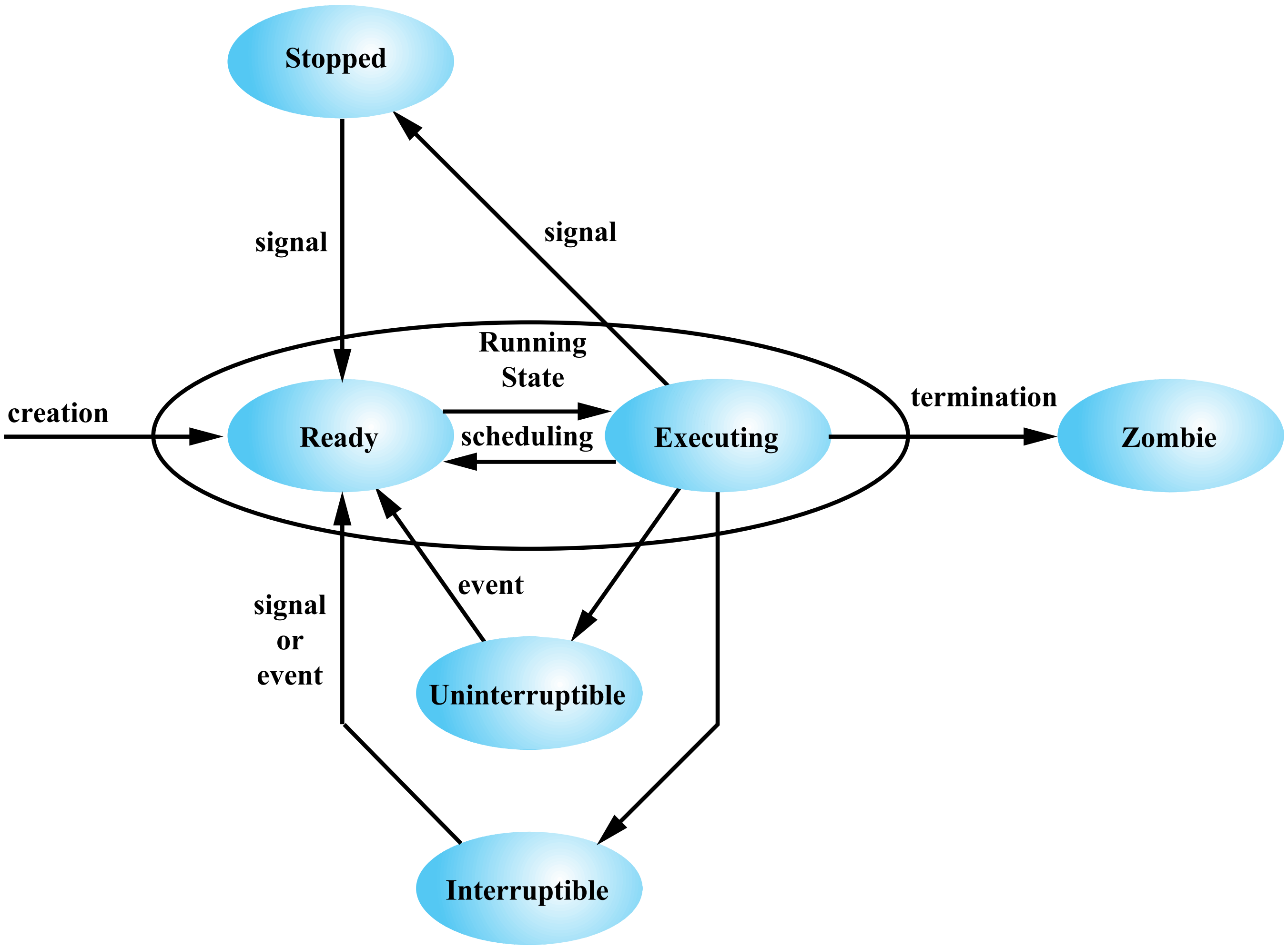


Figure 4.15 Linux Process/Thread Model

What about Concurrency?

Linux uses:

- `libc` (`glibc`) for `pthreads`
- “`futex`”
- Deadlock prevention
- Deadlock detection

Closing Thoughts

If You Like This Stuff…

…and want to learn more, here's more concurrency solutions:

- (Locks)
- Condition Variables (called "Monitors" when used w/ locks)
- (Semaphores)
- Message Passing; e.g., `MPI`
- Event-based; e.g., `nodejs`
- Lock-free Data Structures

If You Like This Stuff…

…and want to learn more, here's some more "famous" problems

- (Producer/Consumer)
- Readers/Writers problem
- (Dining Philosophers)
- Cigarette Smoker's problem
- Sleeping Barber problem
- ABA problem

Learning Group Time

HW 5