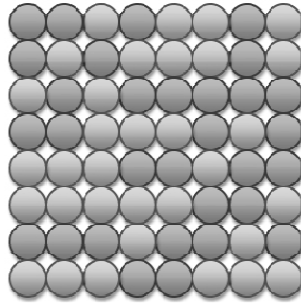




UNIT 10C

Concurrency: Multitasking & Deadlock

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

1

Multitasking & Operating Systems

- **Multitasking** - The coordination of several computational processes on one processor.
- An **operating system** (OS) is the system software responsible for the direct control and management of hardware and basic system operations.
- An OS provides a foundation upon which to run application software such as word processing programs, web browsers, etc.

from Wikipedia

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

2

Operating System “Flavors”

- UNIX
 - System V, BSD, Linux
 - Proprietary: Solaris, Mac OS X
- Windows
 - Windows XP
 - Windows Vista
 - Windows 7

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

3

Multitasking



- Each (un-blocked) application runs for a very short time on the computer's processor and then another process runs, then another...
- This is done at such a fast rate that it appears to the computer user that all applications are running at the same time.
 - How do actors appear to move in a motion picture?

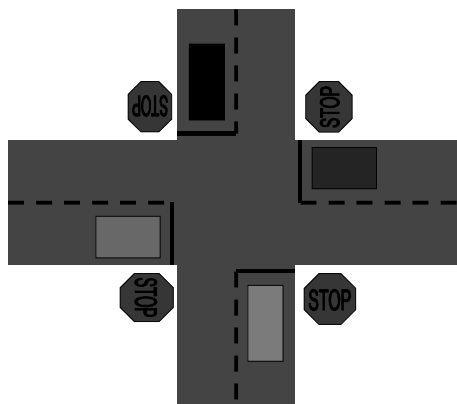
15110 Principles of Computing, Carnegie
Mellon University - CORTINA

4

Critical Section

- A **critical section** is a section of computer code that must only be executed by one process or thread at a time.
- Examples:
 - A printer queue receiving a file to be printed
 - Code to set a seat as reserved
 - Web server that generates and returns a web page showing current registration in course

A Critical Section



Cars may not make turns through this intersection. They may only proceed straight ahead. When a car stops at this intersection, it can proceed straight ahead if there is no car to its left and no car to its right. Otherwise it must wait some random amount of seconds and then check again to see if it's safe to proceed. If not, it waits again, and so on.

Shared Computing Resources

- memory
- tape drives
- disk drives
- printers
- communication ports
- input devices (keyboard, mouse)

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

7

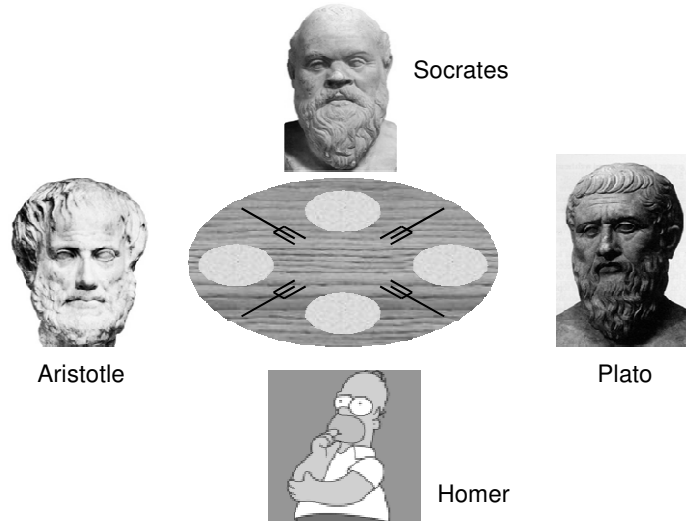
Deadlock

- Deadlock is the condition when two or more processes are all waiting for some shared resource that other processes of the group hold, causing all processes to wait forever without proceeding.
- How can deadlock occur at the intersection with the 4-way stop?

15110 Principles of Computing, Carnegie
Mellon University - CORTINA

8

Dining Philosopher's Problem



15110 Principles of Computing, Carnegie
Mellon University - CORTINA

9

The Dining Philosopher's

- Each philosopher thinks for a while, then picks up his left fork, then picks up his right fork, then eats, then puts down his left fork, then puts down his right fork, thinks for a while...
 - We assume here that each philosopher thinks and eats for random times, and a philosopher cannot be interrupted while he picks up or puts down a single fork.
- Each fork models a "resource" on a computer controlled by an OS.
- Original problem proposed by Edsger Dijkstra.

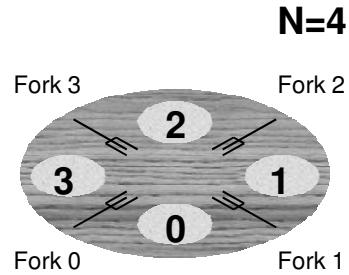
15110 Principles of Computing, Carnegie
Mellon University - CORTINA

10

Dining Philosopher's Problem

- There are N philosophers.
- Philosopher i does the following:

1. THINK
2. Pick up fork i .
3. Pick up fork $(i+1) \text{ modulo } N$.
4. EAT
5. Put down fork i .
6. Put down fork $(i+1) \text{ modulo } N$.
7. Go to step 1.



NOTE: $(i+1) \text{ modulo } N = i+1$, if $0 \leq i < N-1$
 $(i+1) \text{ modulo } N = 0$, if $i = N-1$

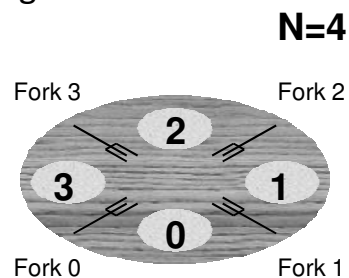
15110 Principles of Computing, Carnegie Mellon University - CORTINA

11

Dining Philosopher's Problem

- There are N philosophers.
- Philosopher i does the following:

1. THINK
2. Pick up fork i .
3. Pick up fork $(i+1) \text{ modulo } N$.
4. EAT
5. Put down fork i .
6. Put down fork $(i+1) \text{ modulo } N$.
7. Go to step 1.



How can deadlock occur here?

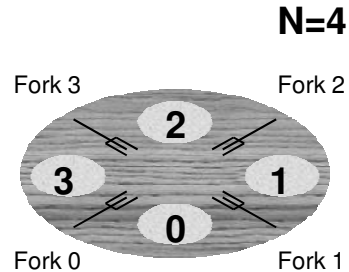
15110 Principles of Computing, Carnegie Mellon University - CORTINA

12

Removing Deadlock

- Philosopher i performs the following:

1. THINK
2. If i is not equal to $N-1$:
 - a. Pick up fork i
 - b. Pick up fork $i+1$
3. If i equals $N-1$:
 - a. Pick up fork 0
 - b. Pick up fork $N-1$
4. EAT
5. If i is not equal to $N-1$:
 - a. Put down fork i
 - b. Put down fork $i+1$
6. If i equals $N-1$:
 - a. Put down fork 0
 - b. Put down fork $N-1$
7. Goto step 1



Can a philosopher still starve?

15110 Principles of Computing, Carnegie Mellon University - CORTINA

13

Semaphores



- A (binary) semaphore S is a shared variable that holds an integer that is initialized to 1.
- We can use a semaphore to protect a critical section so that only one process accesses this section at a given time:

non-critical program code

Request S

CRITICAL SECTION

Release S

non-critical program code

Initially, semaphore S is 1. The first process to request S sets S to 0, blocking other processes from proceeding once they get to their request operations. Once the first process finishes running the critical section, it releases S , causing S to increase to 1, allowing another process to proceed into the critical section.

15110 Principles of Computing, Carnegie Mellon University - CORTINA

14

Summary

- Concurrent computing and multitasking are very challenging problems in computer science.
- The simple rules of sequential computation no longer apply when we compute
 - in a distributed manner across networks
 - in a single computer system in a multitasking environment