So does that mean there are more than two operations?

Monitors - Collection of procedures and variables and data structures grouped together in special kind of module
- Programming language construct.
- Compiler knows monitor is special
- Processor can procedure inside monitor at any time but cannot access internal data structures.
- only one process is allowed to be active in the monitor at a time.
- Maintains Mutual exclusion

Message Passing

Pseudo Code consumer producer problem

```
Producer() {
  while (true) {
    produce.item()
    receive(Consumer, m)
    put.item.imessage(m)
    send (consumer, m)
```

```
Consumer() {
  int i = 0;
  Message m;
  for(i=0; i<n; i++)
    send(producer, m)
  while (true)
```

```
    }
}
```

```
    { receiver (producer)
      Extract (m)
      send (producer, m)
      Consume. item ()
    }
}
```

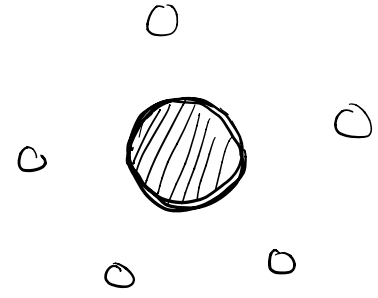
Console. Readline() is a blocking receive.

. . . . . .

Token ring can be used to share variables or find mutex exclusion

Deadlock in a distributed environment.

The dining philosopher



O = plate

//// food

```
Philosopher {
    while (true) {
        think()
        Pickup. fork(right)
        Pickup. fork (Left)
```

```
    eat()
    put down forks(right)
    put down forks(left)
}
```

Condition for deadlock

1. Mutual exclusion
2. Hold & wait - process is allowed to hold resources while waiting on other resources
3. No preemption
4. Circular wait

Condition for deadlock

Necessary not sufficient

1. Mutual exclusion
2. Hold & wait - process is allowed to hold resource while waiting on other resources
3. No preemption
4. Circular wait - actual deadlock
   - this is really what causes deadlock

Dealing with deadlock

- Prevention - really conservative.
  - if we get rid of any of the four above we cannot have deadlock occur.
  - we can get rid of mutual exclusion if they are files that are read only.
  - Easier to get rid of hold and wait.
    - Possible solutions
      - pick up all resources at once
      - put the held resources back that you are holding. re request the data. based on time
        - sucks if the data is not easy to retrieve.
  - non-preemption
    - allow some resources to be preempted.
    - allow higher priority areas to steal

- Circular wait
  - don't allow processes to hold data out of order.

- Avoidance
  - allow all conditions to exist but try and avoid deadlock
    - resource allocation denial
      - Bankers algorith by Dijkstra
      - suppose 3 processes A, B, and C
        10 units of resources
        we know the total need of each process
        A needs 6 units
        B needs 3 units
        C needs 5 units
        keep track of current state

Safe state

processes	Allocation
A	3
B	1
C	3
Available	3

we cannot give up units to programs if it could stop other programs from competing.

- information on the system can be hard

- Process creation denial

- Detection
  - have a monitor that looks over alist and if two processor are holding each other data. we can only look at blocked processor.
  - OS has routine that looks for deadlocks

10/5/18 Day 19

- Circular wait

Deadlock detection

Suppose we have 4 processes, P_1, P_2, P_3, P_4

5 resources, r_1, r_2, r_3, r_4, r_5

	r_1	r_2	r_3	r_4	r_5
available	2	1	1	2	1

Current allocation

	r_1	r_2	r_3	r_4	r_5
P_1	1	0	1	1	0
P_2	1	1	0	0	0
P_3	0	0	0	1	0
P_4	0	0	0	0	0

deadlocked

Requested chart

	r_1	r_2	r_3	r_4	r_5
P_1	0	1	0	0	0
P_2	0	0	1	0	1
P_3	0	0	0	0	1
P_4	1	0	1	0	1

Available resources

r_1	r_2	r_3	r_4	r_5
0	0	0	0	1

Algorithm

1. Mark each process that has a row in matrix is all zero
2. initialize a temporary vector equal to available vector
3. Find i such that process i currently unmarked, and the i^{th} row request matrix is less equal to w with new element by element each element if no such element found terminate algorithm.
4. If such a row is found mark process and add corresponding row of allocation matrix w go to step 3:
   if there are unmarked processes left they are in deadlock

Not collecting

- End of chapter 6
  5, 11, 12, 14, 17, 18

Memory management - Part 3 in text

the way it used to be.

Terms

relocation

protection

sharing

logical organization

physical configuration

Mono programming