

# Monitors

P. J. Denning  
For CS471 / CS571

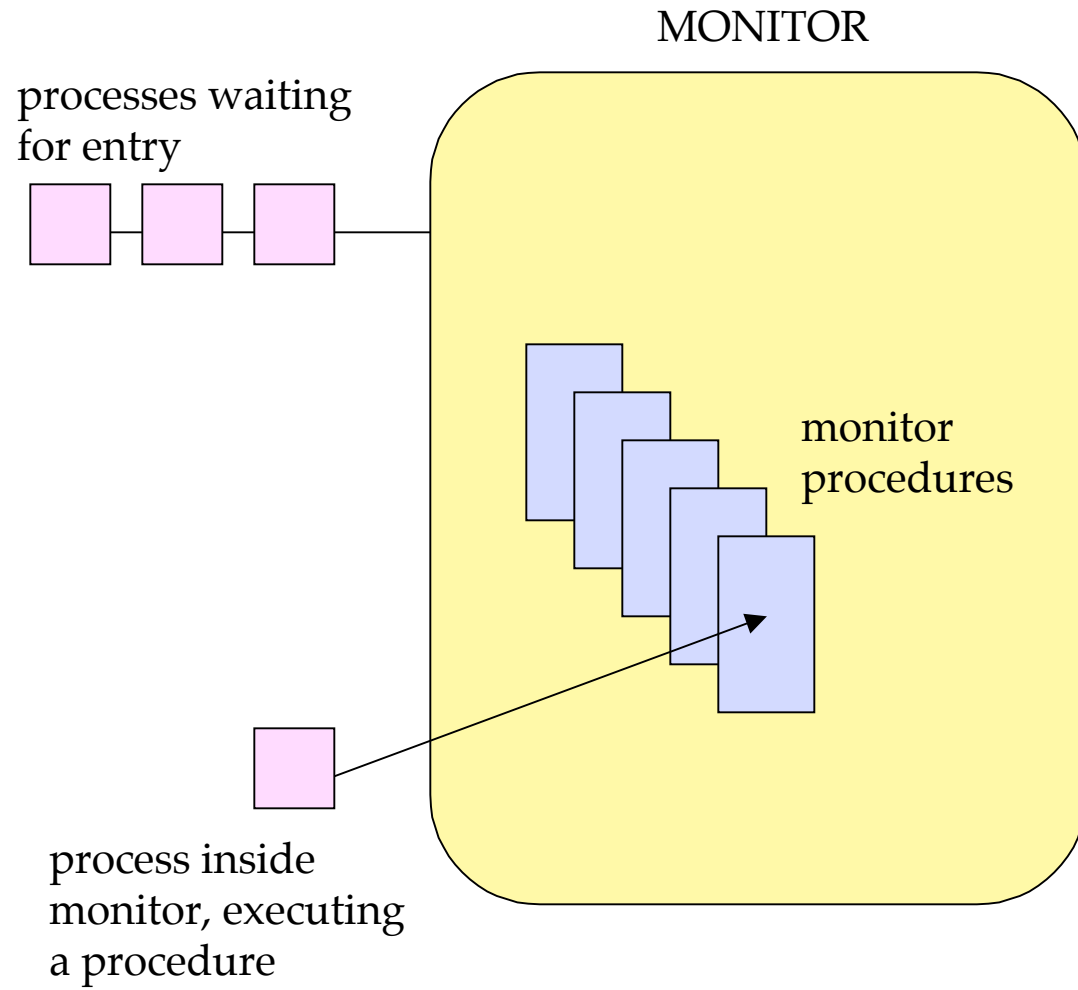
© 2001, P. J. Denning

# Monitors

- A high level language synchronization structure (Hoare 1978)
- Compiler translates monitor into proper semaphore patterns
- Much improved programming reliability

# Monitors

- High level view: monitor is a package of procedures (and data structures); when process enters by calling one of the procedures, the entire monitor is locked to entry by other processes.
- Provides mutually indivisible set of operations on common data.

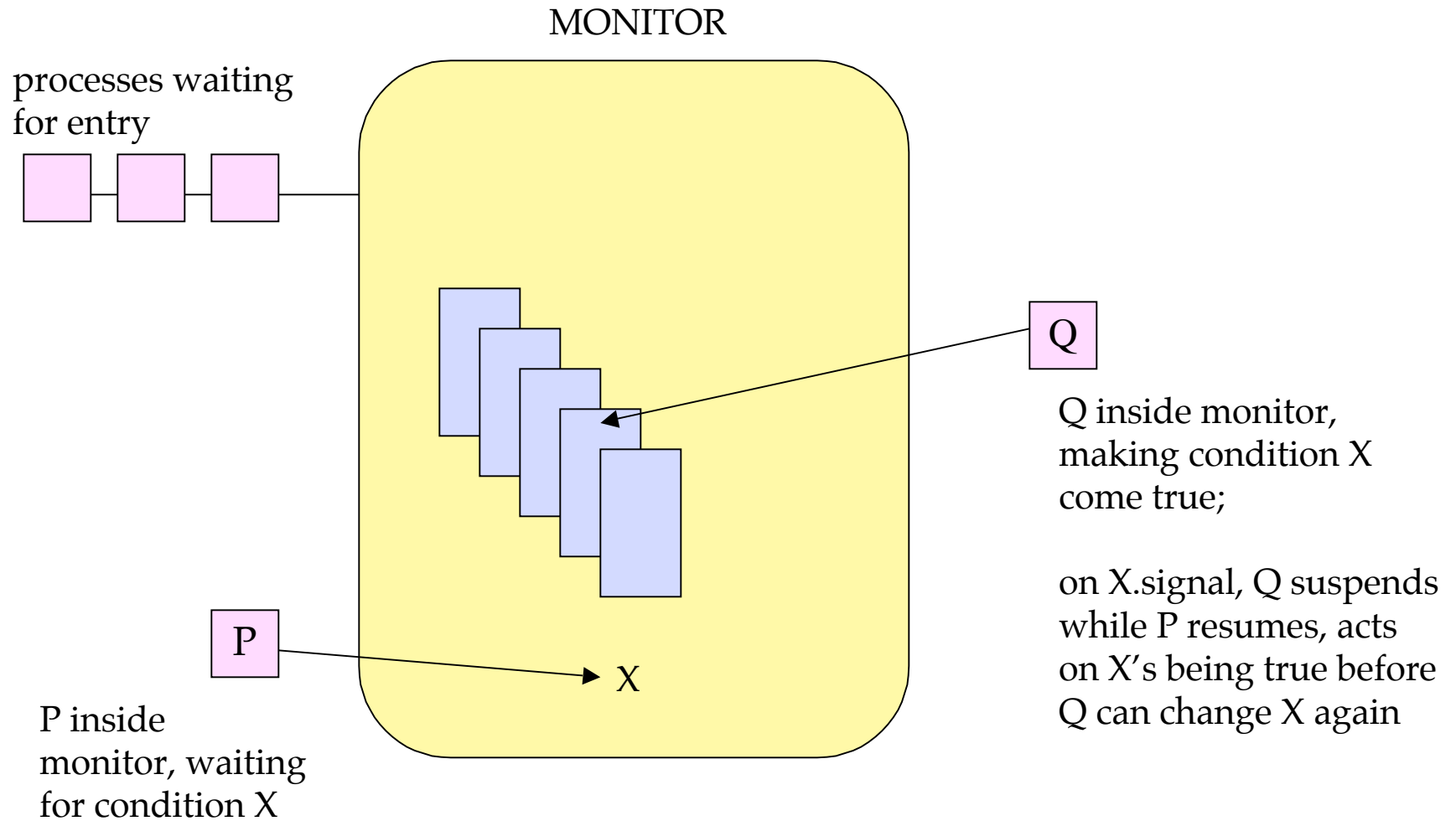


# Monitors

- What if process inside needs to stop and wait?
  - Ex: Pool manager monitor, process executes GetUnit when pool empty?
- How to release monitor exclusion and permit another process to enter and free the waiting one?
  - Ex: another process returns a unit, making it possible for waiting process to proceed.

# Monitors

- Condition variable  $x$ : denotes a boolean condition
- $x.\text{wait}$  -- stop and wait until the condition becomes true
- $x.\text{signal}$  -- let one waiting process (if one exists) know that the condition is true



```
monitor poolmgr
```

```
condition nonempty
```

```
GetUnit:{
```

```
  if poolsize=0 then nonempty.wait
```

```
  h = "remove unit from pool"
```

```
  return h }
```

```
ReturnUnit(h):{
```

```
  "link h back into pool"
```

```
  poolsize++
```

```
  if poolsize=1 then nonempty.signal
```

```
  return }
```

```
end monitor
```

## JAVA:

### synchronized object:

- input queue: threads wanting access for first time
- wait queue: threads previously in a method of the object that stopped to wait for a condition

### Operations:

- wait() -- thread stopped, placed in wait queue
- notify() -- one waiting thread released to input queue
- notifyAll() -- all waiting threads released to input queue

### for condition variable x denoting condition C:

- simulate x.wait with  
    while not C do {wait()}
- simulate x.signal with notify() or notifyAll()

```
synchronized GetUnit:{  
  while poolsize=0 do wait()  
  h = "remove unit from pool"  
  return h }  

```

```
synchronized ReturnUnit(h):{  
  "link h back into pool"  
  poolsize++  
  if poolsize=1 then notify()  
  return }  

```