Shared memory

shared_x

17

Process A

x

Process B

y

α
Semaphore

int

queue of sleeping processes
Initialize Sem.

- value: 1
- sleepers: [Empty]

Two ops.
- Down
- Up

(Share among some processes)
Process A

1. Down(s);
   critical region
   Up(s);

2. if s.value > 0
   s.value --
else
   put calling process to sleep
   add calling process to s.sleepers
A is in its critical region

process B

\[ \text{Down}(s); \]

\[ \text{Up}(s); \]
A is in its critical region

Process C

Down(s)

Up(s)
wake up little B!
if S.sleepers is non-empty
  wake up the front sleeper
  (and remove from S.sleepers)
else
  S.value++;

A
Down(s);
critical region
Up(s);
A Down  O, empty  (A in CR)
B Down  O, B  (B asleep; A in CR)
C Down  O, B, C  (B, C sleep; A in CR)
A Up  O, C  (C sleep; B, C - CR)
B Up  O, X  (-; C in CR)
A Down  O, A  (A sleep; A, C in CR)
C Up  O, X  (-; A in CR)
A Up  1, X  (-; -)
A "mutex" (enforces "mutual exclusion")

"Binary semaphore" (lock, unlock, operations)

Weird problems "Dining Philosophers"

(check Wikipedia)