

Synchronization

(Peterson's Solution)

For Z threads: `int turn; // 0 or 1`
`int interested[Z]`

(can be generalized for >2 threads)

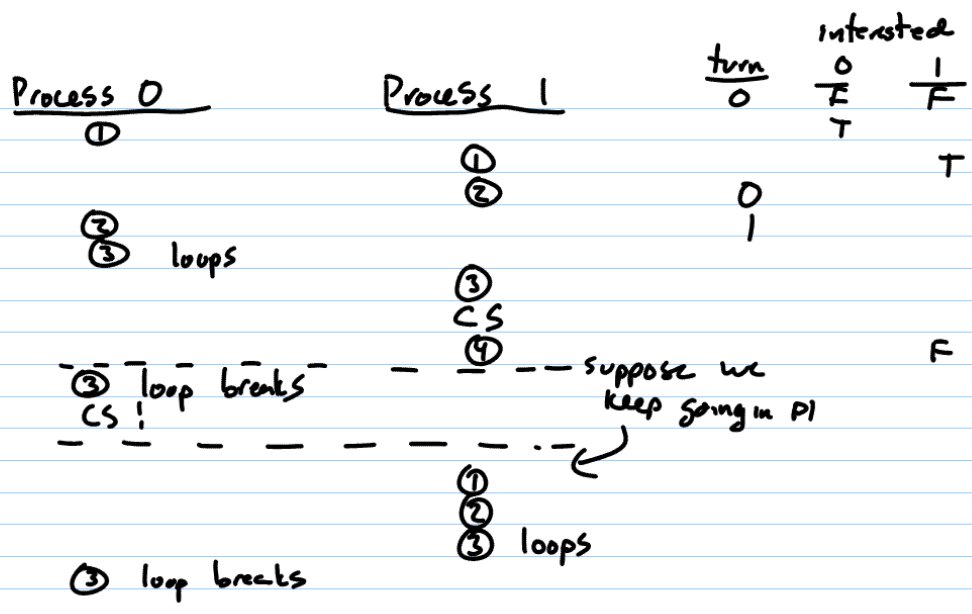
enter_cs: ① `interested[me] = T`
 ② `turn = other`
 ③ `while (turn != me && interested[other] == T)`
 `; // or yield`

leave_cs ④ `interested[me] = F`

`enter_cs();`

`// CRITICAL SECTION`

`leave_cs();`



monitor: collection of data and procedures
 (programming language construct) data can only be accessed by procedures
 procedures are mutually exclusive

condition variable: associated with some boolean condition

monitor
 proc A
 proc B
 proc C
 ⋮

3 operations

- 1) wait - blocks calling process thread is in proc A no other thread can be in A, B, or C
- 2) signal - unblock one blocked process
- 3) broadcast - unblocks all blocked processes

typical use: - if (something important is F)
 wait
 - modify data
 if (modification made something important T)
 signal / broadcast

Producer / Consumer

- 1 thread (producer) puts things into a fixed-length buffer
web crawler putting HTML code in
- 1 thread consumer takes things out
scraper harvesting e-mail addresses

Java: methods in monitor tagged w/ synchronized

every object has a condition variable

operations: wait / notify / notifyAll