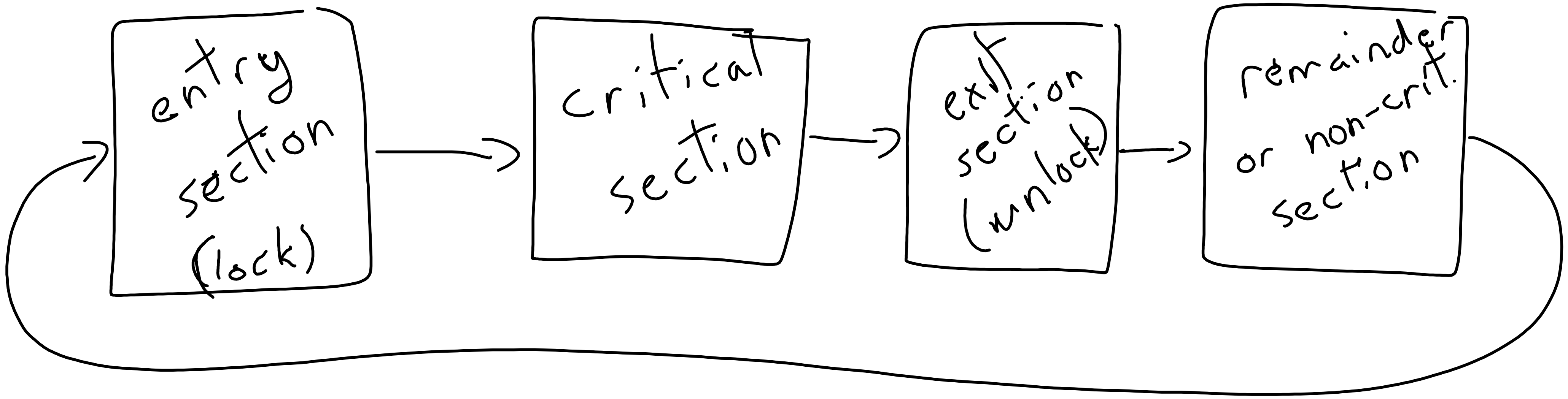
Critical Region/Section Problem

- Processes access a shared resource.



Critical Regions

Requirements to avoid race conditions:

1. No two processes may be simultaneously inside their critical regions.
2. No assumptions may be made about speeds or the number of CPUs.
3. No process running outside its critical region may block other processes.
4. No process should have to wait forever to enter its critical region.

1. mutual exclusion
2. processing

3. progress
4. bounded wait
(starvation)

Mutual Exclusion with Busy Waiting: Strict Alternation

wait for turn 0 →

```
while(TRUE){
 while(turn != 0);
 critical_region();
 turn = 1;
 noncritical_region();
}
```

```
while(TRUE){
 while(turn != 1);
 critical_region();
 turn = 0;
 noncritical_region();
}
```

← wait for input

- Unfortunately, this is yet another (non)solution:
  - Does not permit processes to enter their critical regions two times in a row
  - A process outside the critical region can effectively block another one

Mutual Exclusion with Busy Waiting: Peterson’s algorithm

```
#define N 2

int turn;
int interested[N];

void enter_region(int process){
 int other = 1 - process;
 interested[process] = TRUE;
 turn = process;
 while(turn==process && interested[other]==TRUE);
}

void leave_region(int process){
 interested[process] = FALSE;
}
```