**Operating Systems**
**Software Methods of Enforcing Mutual Exclusion**

---

These "solutions" are not all correct. Some are "correct" but are not desirable. They all have certain limitations. One such limitation is that these are implemented for providing mutual exclusion between two processes only.

## Solution #1

For this solution assume that `turn` is initially 0 and that there are only two processes requesting the critical section.

```
Process 0                    Process 1
while (true) {               while (true) {
  nonCriticalSection()        nonCriticalSection()
  while (turn == 1) {}        while (turn == 0) {}
  criticalSection()           criticalSection()
  turn= 1                     turn= 0
}                            }
```

Questions to ponder:

- What is happening?
- What could be done to make this work for $n$ processes?
- What are some problems with this method?

## Solution #2

For this solution assume that `p0inside` and `p1inside` are initially false.

```
Process 0                    Process 1
while (true) {               while (true) {
  nonCriticalSection()         nonCriticalSection()
  while (p1inside) {}          while (p0inside) {}
  p0inside= true               p1inside= true
  criticalSection()            criticalSection()
  p0inside= false              p1inside= false
}                            }
```

Questions to ponder:

- What is a problem with this "solution"?
- What would happen if `p0inside= true` and `p1inside= true` were moved *before* the while loop?

## Solution #3

For this solution assume that `p0wantsToEnter` and `p1wantsToEnter` are initially false and that `favoredProcess` is initially 1.

```
Process 0                       Process 1
while (true) {                  while (true) {
  nonCriticalSection()            nonCriticalSection()
  p0wantsToEnter= true            p1wantsToEnter= true
  favoredProcess= 1               favoredProcess= 0
  while (p1wantsToEnter &&        while (p0wantsToEnter &&
        favoredProcess==1) {}           favoredProcess==0) {}
  criticalSection()               criticalSection()
  p0wantsToEnter= false          p1wantsToEnter= false
}                               }
```

Questions to ponder:

- What is a problem with this "solution"?
- How would you extend this solution to work with $n$ processes?

## A Hardware/Software Solution

Suppose the hardware implements an indivisible `testAndSet` instruction that works like this:

```
int testAndSet(int flag)
{
  if (flag==0) {
    flag= 1;
    return 0;
  } else
      return 1;
}
```

Here is a possible solution for $n$ processes that uses such an instruction. Assume that flag is initially 0.

```
Process i
while (true) {
  nonCriticalSection()
  while (testAndSet(flag) == 1) {}
  criticalSection()
  flag= 0
}
```