CSE 330: Operating Systems

Class: 09 Date: 9/15

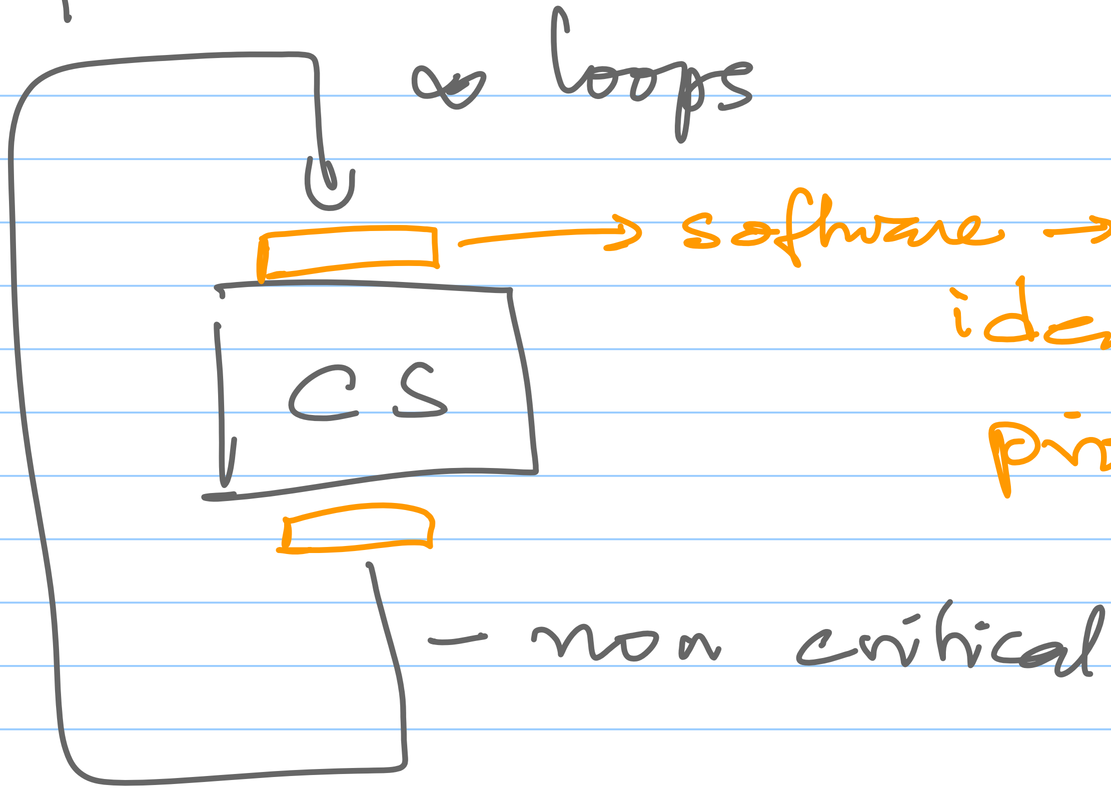
Fall 2016

Note Title

Critical sections

- mutex
- progress → if no process in cs & 1 or more want to enter → select from those that want to enter + do not delay for ever
- bounded waiting

2 processes



software → is almost identical in both processes

2 processes, i & j
or 0 & 1

```
flag[i] = 1
while (flag [j] == 1);

    CS

  flag [i] = 0
```

→ mutex ✓

→ progress ✗
(deadlock)

```
flag[i] = 1
while (flag[j] == 1) { flag[i] = 0;
                       flag[i] = 1 }

CS

flag i = 0
```

<u>no deadlock</u>

→ starvation
→ livelock

mutex ✓

✓ progress ✗ may delay

✗ bounded waiting

```
while flag[j] == 1 ;
flag[i] = 1
    CS
  flag[j] = 0
```

turn = i or j // init

```
while (turn == j);
  CS
  turn = i
```

process i

mutex ✓

progress ✗

Dekker's Solution

1965

```
flag[0] = 1
while flag[1] == 1 {
    if turn ≠ 0 {
        flag[0] = 0;
        while turn ≠ 0 {
        }
        flag[0] = 1
    }
}

// critical section
...
turn = 1;
flag[0] = 0;
// remainder section
```

Peterson's Solution 1981

Process[0]

flag[0] = 1;

turn = 1;

while (flag[1]==1 && turn == ~~j~~ 1);

Critical Section

----

flag[0] = 0;

Process[1]

flag[1] = ~~true~~; 1

turn = 0;

while (flag[0]==1 && turn == 0);

Critical Section

flag[1] = 0;

multiple processes . . .

Peterson's solution

cannot be extended

Bakery Algorithm (1974)

process i (by Lamport)

```
~~choosing[i] = 1;~~
num[i] = max(num[0],num[1]...num[n-1])+1;
~~choosing[i] = 0;~~
for j = 0 to n-1 {
    ~~while (choosing[j]==1) <no-op>;~~
     while (num[j]<>0 and (num[j],j)<(num[i],i) <no-op>;
};
```

!=

(numj < num[i])

n processes

fixed #

n items

num | 0 | 0 | 2 | 3 | 0 | 3 |

lower index

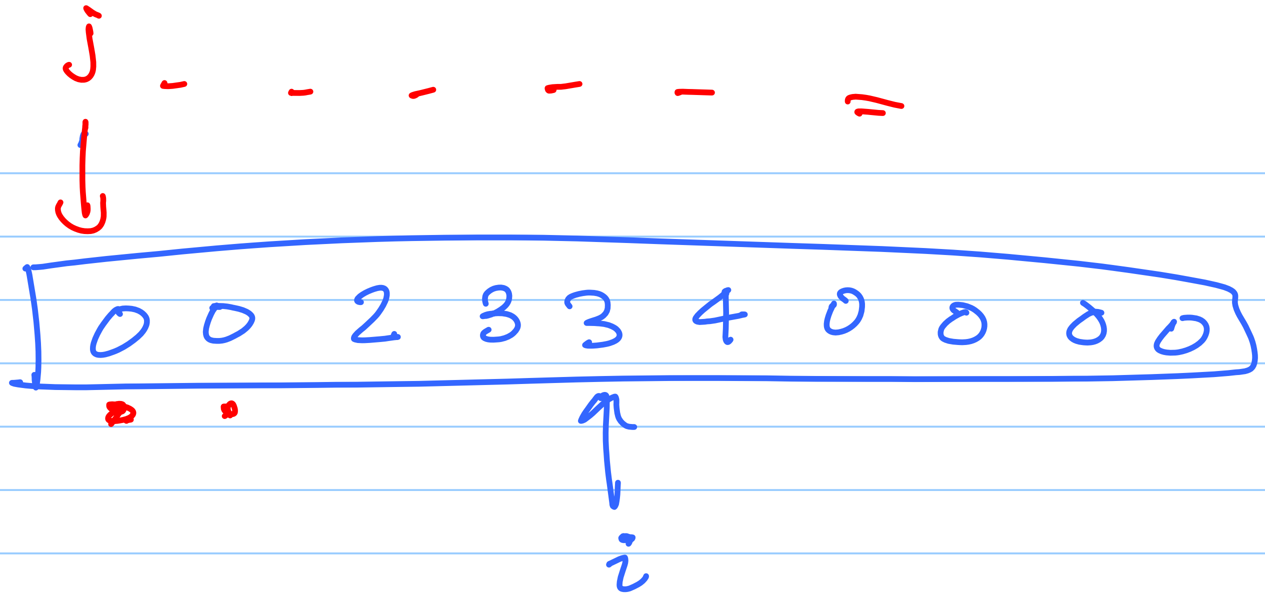
Critical Section

```
num[i] = 0;
```

$(\text{num}(j), j) < (\text{num}(i), i)$

$\Rightarrow$ if (num[j] < num[i]) then true

else if num j == num i then i < j $\rightarrow$ T


j
0 0 2 3 3 4 0 0 0 0
i

Practical solutions ...

Peterson - 2 processes only

Lamport - N is fixed & cannot be changes

↳ Both use busy waiting

→ error prone, hard to comprehend

Hardware based

- UNIProcessor Systems

Disable all interrupts

$\boxed{\text{Critical Section}}$

Enable all interrupts

Disabling Interrupts

- Overkill
- used for VERY short critical sections (order of microsecs)

↳ used to build longer critical sections - using other techniques.

- CANNOT be used on multicore