Critical Section

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

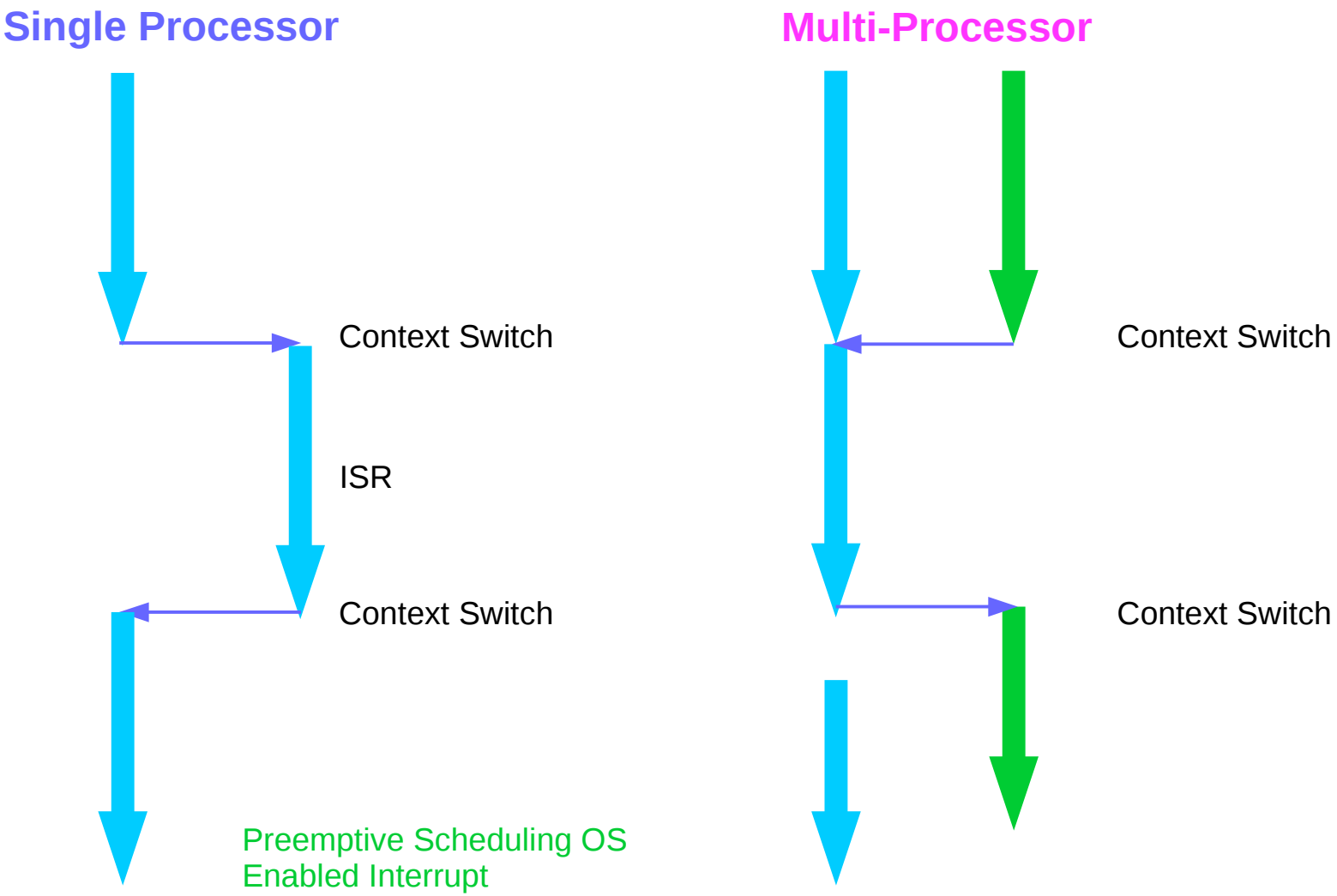
Useful Links

Uoc.gr
http://www.csd.uoc.gr/~hy586/material/lectures/cs586-Section2.pdf
Until Peterson's algorithm

UWisc, EECE6354,
http://pages.cs.wisc.edu/~remzi/OSTEP/threads-locks.pdf
Multitasking, RTK,

Uregina,
http://www2.cs.uregina.ca/~hamilton/courses/330/notes/synchro/node3.html

Single Processor v.s. Multi-processor



Try 1 – working examples

lock = false;

```
P0:
while (true) {
        while (lock) { ; }
        lock = true;
        critical section
        lock = false;
        non-critical section
}
```

```
P1:
while (true) {
        while (lock) { ; }
        lock = true;
        critical section
        lock = false;
        non-critical section
}
```

Wait for the releasing of the lock
Set the lock before entering the CS.

P0 sets / clears lock
P1 sets / clears lock

```
P0:
while (true) {
        while (lock) { ; }
        lock = true;
        critical section
        lock = false;
        non-critical section
}
```

```
P1:
while (true) {
        while (lock) { ; }
        lock = true;
        critical section
        lock = false;
        non-critical section
}
```

Try 2 – working examples

want[2] = {false, false};

```
P0:
while (true) {
        while (want[1]) { ; }
        want[0] = true;
        critical section
        want[0] = false;
        non-critical section
}
```

```
P1:
while (true) {
        while (want[0]) { ; }
        want[1] = true;
        critical section
        want[1] = false;
        non-critical section
}
```

Wait for the releasing of the want[J]
Set the want[I] before entering the CS.

P0 sets / clears want[0]
P1 sets / clears want[1]

If you want, I'll wait

```
P0:
while (true) {
        while (want[1]) { ; }
        want[0] = true;
        critical section
        want[0] = false;
        non-critical section
}
```

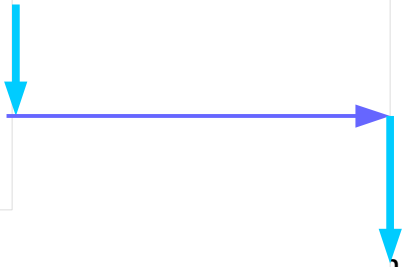
```
P1:
while (true) {
        while (want[0]) { ; }
        want[1] = true;
        critical section
        want[10] = false;
        non-critical section
}
```

Try 3 – working examples

```
turn = 1;
```

```
P0:
while (true) {
        while (turn != 0) { ; }
        critical section
        turn = 1;
        non-critical section
}
```

```
P1:
while (true) {
        while (turn != 1) { ; }
        critical section
        turn = 0;
        non-critical section
}
```



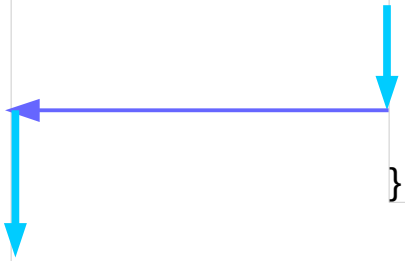
Wait for my turn
Switch the turn after exiting the CS.

P0 sets turn to 1
P1 sets turn to 0

I'll wait for my turn

```
P0:
while (true) {
        while (turn != 0) { ; }
        critical section
        turn = 1;
        non-critical section
}
```

```
P1:
while (true) {
        while (turn != 1) { ; }
        critical section
        turn = 0;
        non-critical section
}
```

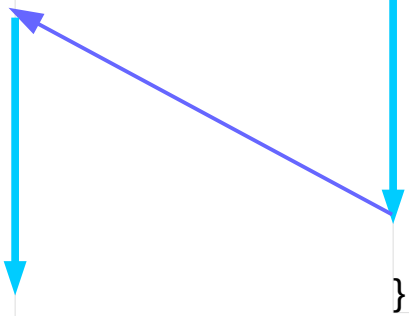


Try 4 – working examples

want[2] = {false, false};

```
P0:
while (true) { want[0] = true;
        while (want[1]) {
                want[0] = false;
                while (want[1]) {;}
                want[0] = true;
        }       critical section
        want[0] = false;
        non-critical section
}
```

```
P1:
while (true) { want[1] = true;
        while (want[0]) {
                want[1] = false;
                while (want[0]) {;}
                want[1] = true;
        }       critical section
        want[1] = false;
        non-critical section
}
```



```
P1:
while (true) { want[1] = true;
        while (want[0]) {
                want[1] = false;
                while (want[0]) {;}
                want[1] = true;
        }       critical section
        want[1] = false;
        non-critical section
}
```

```
P0:
while (true) { want[0] = true;
        while (want[1]) {
                want[0] = false;
                while (want[1]) {;}
                want[0] = true;
        }       critical section
        want[0] = false;
        non-critical section
}
```

Wait for the releasing of the want[J]
While waiting, temporarily clears want[I]
Set the want[I] before entering the CS.

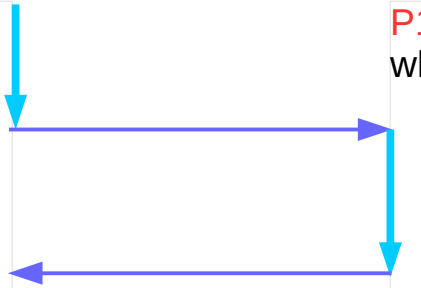
P0 sets / clears want[0]
P1 sets / clears want[1]

If you want, go ahead
I'll wait and do

Try 1 – Not working examples

lock = false;

```
P0:
while (true) {
        while (lock) { ; }
        Context Switching
        lock = true;
        critical section
        lock = false;
        non-critical section
}
```



```
P1:
while (true) {
        while (lock) { ; }
        lock = true;
        critical section
        lock = false;
        non-critical section
}
```

Wait for the releasing of the lock
Set the lock before entering the CS.

P0 sets / clears lock
P1 sets / clears lock

after seeing false lock,
a context switch occurs
plan to enter its critical section

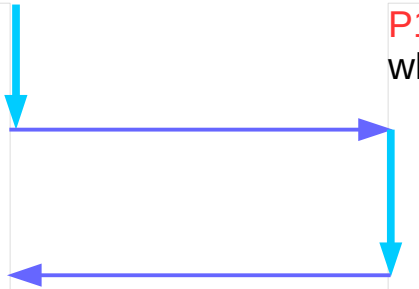
after seeing false lock,
entered its critical section
but before releasing the lock
a context switch occurs

Both P0 & P1 are in the critical section

Try 2 – Not working examples

want[2] = {false, false};

```
P0:
while (true) {
        while (want[1]) { ; }
        Context Switching
        want[0] = true;
        critical section
        want[0] = false;
        non-critical section
}
```



```
P1:
while (true) {
        while (want[0]) { ; }
        want[1] = true;
        critical section
        want[1] = false;
        non-critical section
}
```

Wait for the releasing of the want[J]
Set the want[I] before entering the CS.

P0 sets / clears want[0]
P1 sets / clears want[1]

If you want, I'll wait

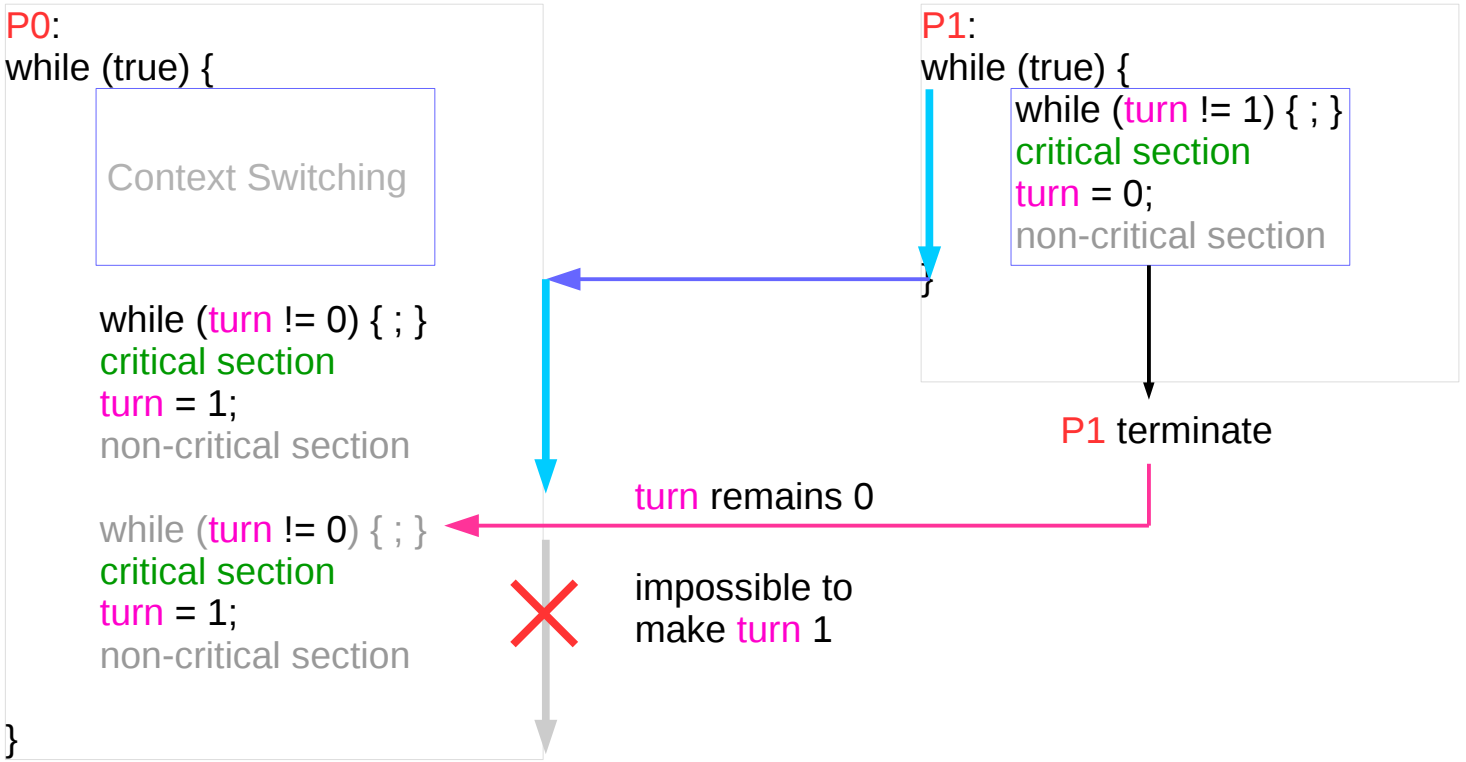
after seeing false want[1],
a context switch occurs
plan to enter its critical section

after seeing false want[0],
entered its critical section
but before releasing the lock
a context switch occurs

Both P0 & P1 are in the critical section

Try 3 – Not working examples

turn = 1;



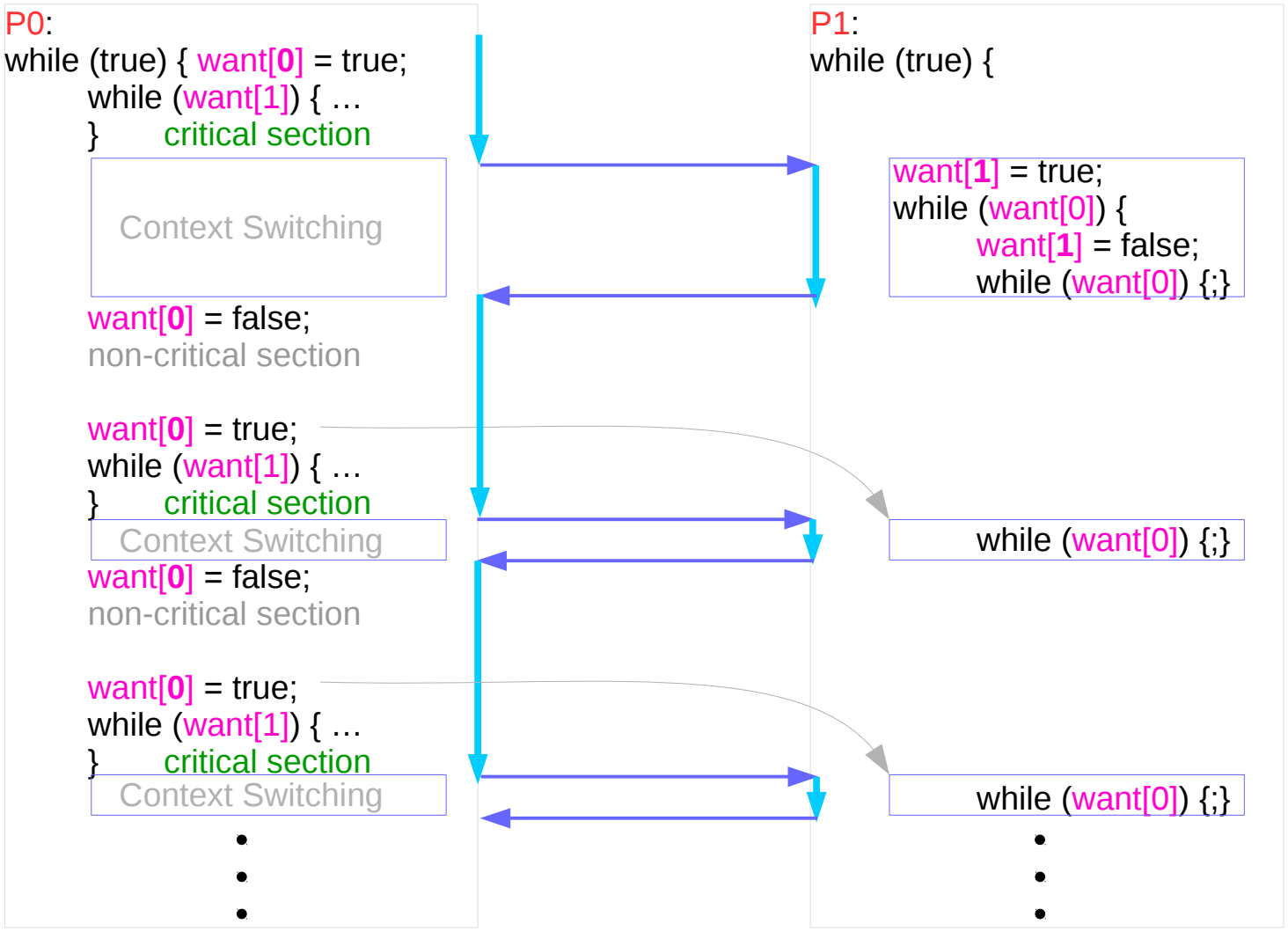
Wait for my turn
Switch the turn after exiting the CS.

P0 sets turn to 1
P1 sets turn to 0

I'll wait for my turn

Try 4 – Not working examples

want[2] = {false, false};



Wait for the releasing of the want[J]
While waiting, temporarily clears want[I]
Set the want[I] before entering the CS.

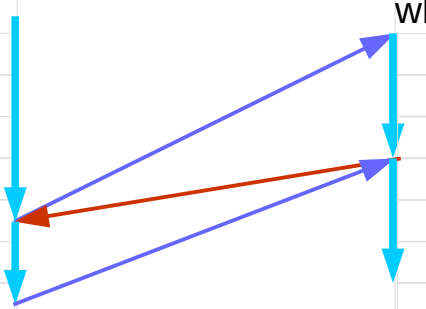
P0 sets / clears want[0]
P1 sets / clears want[1]

If you want, go ahead
I'll wait and do

Paterson's Algorithm

```
want[2] = {false, false};        turn;
```

```
P0:
while (true) {
        want[0] = true;
        turn = 1;
        while
         (want[1] && turn==1) { ; }
        critical section
        want[0] = false;
        non-critical section
}
```



```
P1:
while (true) {
        want[1] = true;
        turn = 0;
        while
         (want[0] && turn==0) { ; }
        critical section
        want[1] = false;
        non-critical section
}
```

busy waiting until context sw

Wait for the releasing of either want[J] or J turn. Before waiting, first yield to J turn

P0 sets / clears want[0]
P1 sets / clears want[1]

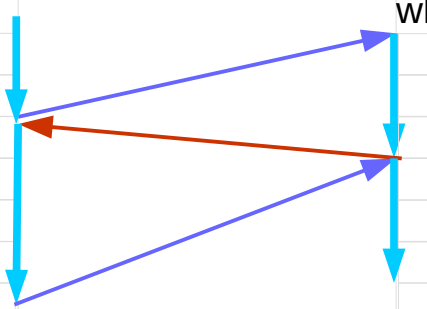
P0 sets turn to 1
P1 sets turn to 0

http://www2.cs.uregina.ca/~hamilton/courses/330/notes/synchro/node3.html

Paterson's Algorithm

```
want[2] = {false, false};        turn;
```

```
P0:
while (true) {
        want[0] = true;
        turn = 1;
        while
         (want[1] && turn==1) { ; }
        critical section
        want[0] = false;
        non-critical section
}
```



```
P1:
while (true) {
        want[1] = true;
        turn = 0;
        while
         (want[0] && turn==0) { ; }
        critical section
        want[1] = false;
        non-critical section
}
```

busy waiting until context sw

Wait for the releasing of either want[J] or J turn. Before waiting, first yield to J turn

P0 sets / clears want[0]
P1 sets / clears want[1]

P0 sets turn to 1
P1 sets turn to 0

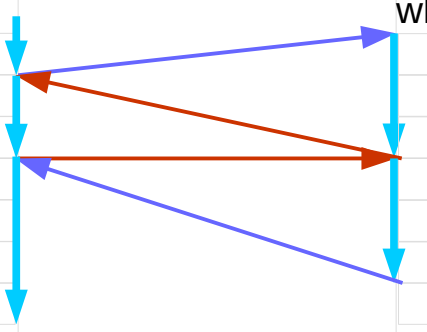
http://www2.cs.uregina.ca/~hamilton/courses/330/notes/synchro/node3.html

Paterson's Algorithm

```
want[2] = {false, false};        turn;
```

```
P0:
while (true) {
        want[0] = true;
        turn = 1;
        while
         (want[1] && turn==1) { ; }
        critical section
        want[0] = false;
        non-critical section
}
```

busy waiting until context sw



```
P1:
while (true) {
        want[1] = true;
        turn = 0;
        while
         (want[0] && turn==0) { ; }
        critical section
        want[1] = false;
        non-critical section
}
```

busy waiting until context sw

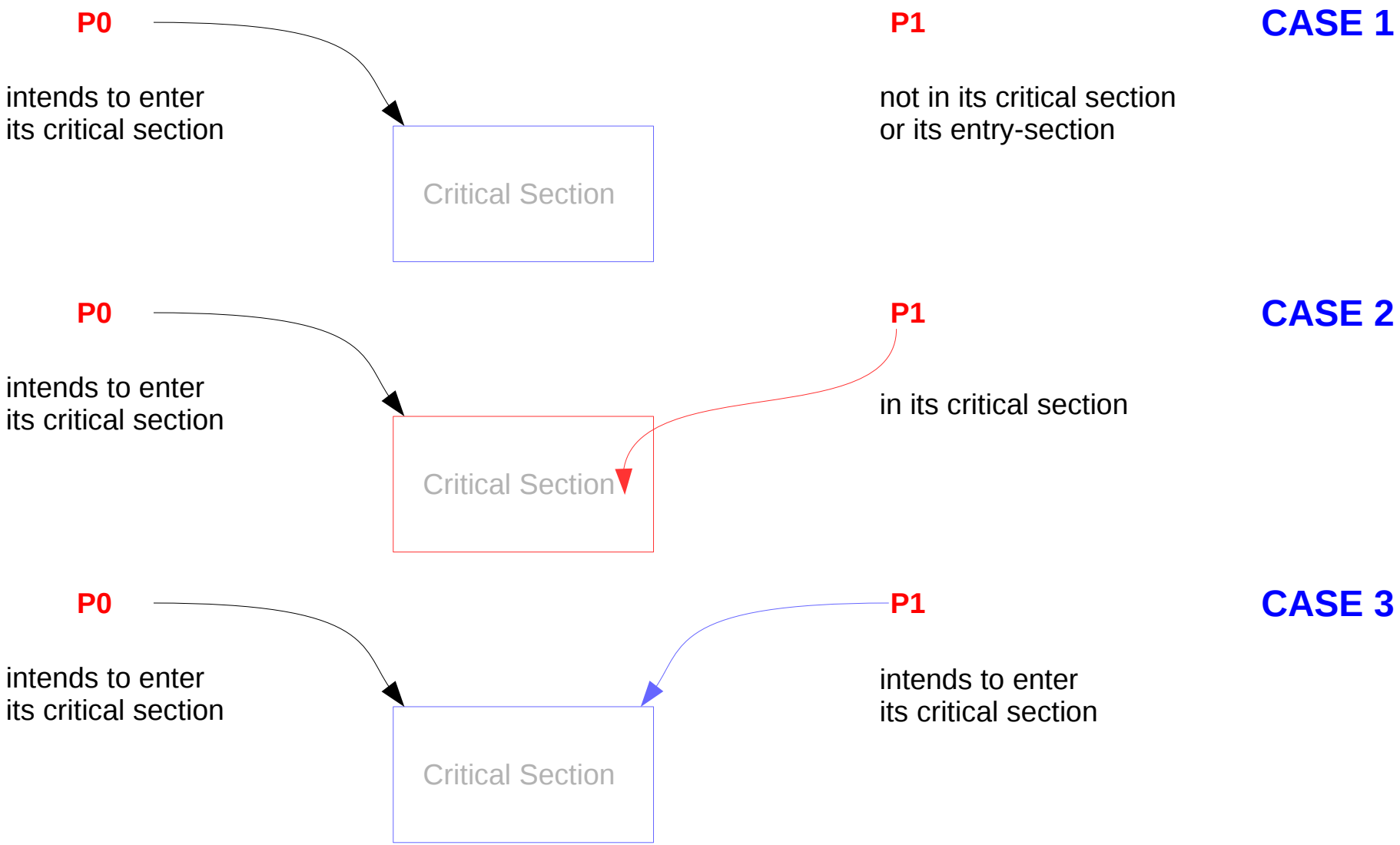
Wait for the releasing of either want[J] or J turn. Before waiting, first yield to J turn

P0 sets / clears want[0]
P1 sets / clears want[1]

P0 sets turn to 1
P1 sets turn to 0

http://www2.cs.uregina.ca/~hamilton/courses/330/notes/synchro/node3.html

Test Cases


P0
intends to enter
its critical section
Critical Section
P1
not in its critical section
or its entry-section
CASE 1
P0
intends to enter
its critical section
Critical Section
P1
in its critical section
CASE 2
P0
intends to enter
its critical section
Critical Section
P1
intends to enter
its critical section
CASE 3

Case 1

want[2] = {false, false}; turn;

```
P0:
while (true) {
        want[0] = true;
        turn = 1;
        while
         (want[1] && turn==1) { ; }
        critical section
        want[0] = false;
        non-critical section
}
```

```
P1:
```

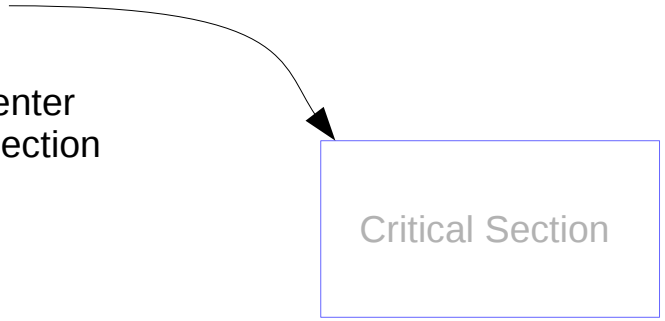
Wait for the releasing of either want[J] or J turn. Before waiting, first yield to J turn

P0 sets / clears want[0]
P1 sets / clears want[1]

P0 sets turn to 1
P1 sets turn to 0

P0

intends to enter
its critical section



P1

not in its critical section
or its entry-section

CASE 1

Case 2

```
want[2] = {false, false};        turn;
```

```
P0:
while (true) {
        want[0] = true;
        turn = 1;


        while
         (want[1] && turn==1) { ; }
        critical section
        want[0] = false;
        non-critical section
}
```

```
P1:
while (true) {
        want[1] = true;
        turn = 0;
        while
         (want[0] && turn==0) { ; }
        critical section
        want[1] = false;
        non-critical section
}
```

Wait for the releasing of either want[J] or J turn. Before waiting, first yield to J turn

P0 sets / clears want[0]
P1 sets / clears want[1]

P0 sets turn to 1
P1 sets turn to 0

CASE 2

Case 3

want[2] = {false, false}; turn;

```
P0:
while (true) {
        want[0] = true;
        turn = 1;
        while
         (want[1] && turn==1) { ; }
        critical section
        want[0] = false;
        non-critical section
}
```

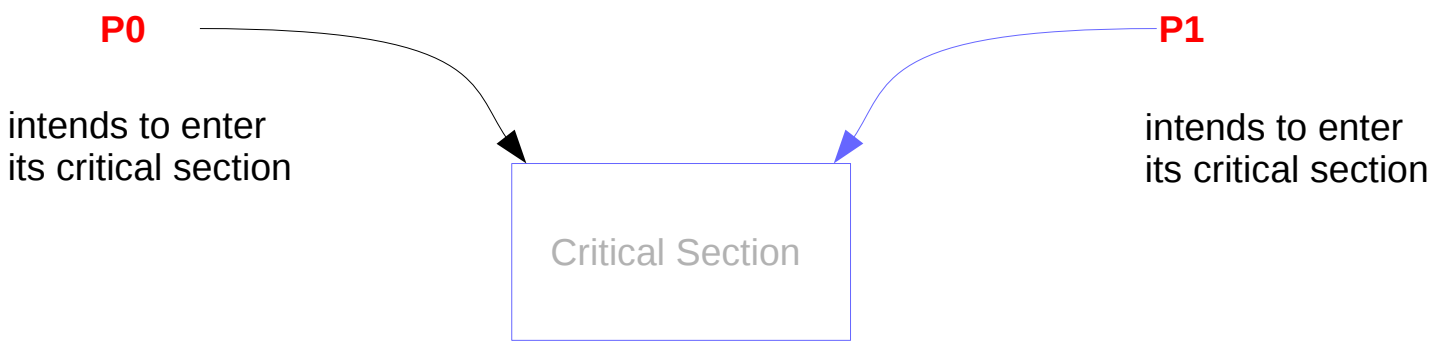
```
P1:
while (true) {
        want[1] = true;

        turn = 0;
        while
         (want[0] && turn==0) { ; }
        critical section
        want[1] = false;
        non-critical section
}
```

Wait for the releasing of either want[J] or J turn. Before waiting, first yield to J turn

P0 sets / clears want[0]
P1 sets / clears want[1]

P0 sets turn to 1
P1 sets turn to 0

PI first executes turn = **J;** waits until **PJ** executes turn = **I** and then enters its critical section.
The **PJ** will be the next



CASE 3

References

[1] http://crystal.uta.edu/~ylei/cse6324/data/critical-section.pdf