# Concurrency: Multi-core Programming & Data Processing

#### Lab 6

-- Peterson's algorithm revisited --

## Peterson's mutual exclusion... again

- Generalized mutual exclusion algorithm for n threads
- What we already know...

Thread	1	2	3
Level	1	1	1

```
Legend: Variables:

n - total levels n = 3

L - current level L = 1
```

i - thread id i = 1,2,3

	Thread	1	2
L L-/ \	Level	1	1
lock() {     for (int L = 1; L < n; L++) {         level[i] = L;		level[1]=1	level[2]=1
<pre>victim[L] = i; while (( exists k != i with level[k] &gt;= L)           &amp;&amp; victim [L] == i ) {};</pre>			

3

1

level[3]=1

```
Legend: Variables:

n - total levels n = 3

L - current level L = 1

i - thread id i = 1,2,3
```

lock() {
for (int L = 1; L < n; L++) {
level[i] = L;
victim[L] = i;
while (( exists k != i with level[k] >= L)
&& victim [L] == i ) {};
}
}

```
        Thread
        1
        2
        3

        Level
        1
        1
        1

        level[1]=1
        level[2]=1 victim[1]=2
        level[3]=1
```

```
Legend: Variables:

n - total levels n = 3

L - current level L = 1

i - thread id i = 2
```

```
lock() {
  for (int L = 1; L < n; L++) {
    level[i] = L;
    victim[L] = i;
    while (( exists k != i with level[k] >= L)
         && victim [L] == i ) {};
  }
}
```

```
        Thread
        1
        2
        3

        Level
        1
        1
        1

        level[1]=1
        level[2]=1 victim[1]=2 Blocked!
        level[3]=1
```

```
Legend: Variables:

n - total levels n = 3

L - current level L = 1

i - thread id i = 2
```

	meau
lock() {	Level
for (int L = 1; L < n; L++) {	
level[i] = L;	
victim[L] = i;	
while (( exists k != i with level[k] >= L)	
&& victim [L] == i ) {};	
}	
}	

```
        Thread
        1
        2
        3

        Level
        1
        1
        1

        level[1]=1
        level[2]=1 victim[1]=2 Blocked!
        level[3]=1
```

```
Legend: Variables:

n - total levels n = 3

L - current level L = 1

i - thread id i = 1
```

	Thread	1	2	3
look() (	Level	1	1	1
lock() {    for (int L = 1; L < n; L++) {      level[i] = L;      victim[L] = i;		level[1]=1	level[2]=1 victim[1]=2 	level[3]=1
<pre>while (( exists k != i with level[k] &gt;= L)</pre>		victim[1]=1 Blocked!		
}				
	Level		2	

	Thread	1	2	3
	Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt; n; L++) {     level[i] = L;     victim[L] = i;</pre>		level[1]=1	level[2]=1 victim[1]=2 	level[3]=1
while (( exists k != i with level[k] >= L) && victim [L] == i ) {};		victim[1]=1 Blocked!		
<pre>} }</pre>				victim[1]=3
	Level		2	

```
2
                                            Thread
                                                                  1
                                                                                                             3
                                            Level
                                                                 1
                                                                                       1
                                                                                                             1
lock() {
                                                                 level[1]=1
                                                                                       level[2]=1
                                                                                                             level[3]=1
for (int L = 1; L < n; L++) {
                                                                                       victim[1]=2
  level[i] = L;
  victim[L] = i;
  while (( exists k != i with level[k] >= L)
                                                                 victim[1]=1
         && victim [L] == i ) {};
                                                                                                            victim[1]=3
                                                                                                              Blocked!
                                            Level
```

		Thread	1	2	3
lo als/) (		Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt;     level[i] = L;     victim[L] = i;</pre>	< n; L++) {		level[1]=1	level[2]=1 victim[1]=2 	level[3]=1
	k != i with level[k] >= L)		victim[1]=1		
&& vict } }	im [L] == i ) {};				victim[1]=3 Blocked!
		Level	2	2	
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 2 i = 1,2		level[1]=2	level[2]=2	

	Thread	1	2	3
	Level	1	1	1
< n; L++) {		level[1]=1	level[2]=1 victim[1]=2 	level[3]=1
k != i with level[k] >= L)		victim[1]=1		
im [L] == i ) {};				
				victim[1]=3
		$\downarrow$	<b>↓</b>	Blocked!
	Level	2	2	
Variables: n = 3 L = 2 i = 2		level[1]=2	level[2]=2 victim[2]=2	
	<pre>im [L] == i ) {};  Variables:  n = 3  L = 2</pre>	Level  k!= i with level[k] >= L) im [L] == i) {};  Level  Variables: n = 3 L = 2	Level 1   level[1]=1   k!= i with level[k] >= L)   victim[1]=1     im [L] == i) {};   Level 2   Variables:   n = 3     L = 2	Level 1 1 1   level[2]=1   victim[1]=2     level[2]=2   victim[2]=2     level[2]=2   victim[2]=2     level[2]=2   victim[2]=2   vic

		Thread	1	2	3
lock() (		Level	1	1	1
lock() {     for (int L = 1; L <         level[i] = L;     victim[L] = i;	< n; L++) {		level[1]=1	level[2]=1 victim[1]=2 	level[3]=1
while (( exists l	k!= i with level[k] >= L)		victim[1]=1		
&& victi } }	m [L] == i ) {};				victim[1]=3 Blocked!
		Level	2	2	
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 2 i = 2		level[1]=2	level[2]=2 victim[2]=2 Blocked!	

		Thread	1	2	3
l = = l./ \		Level	1	1	1
lock() {     for (int L = 1; L <         level[i] = L;     victim[L] = i;	< n; L++) {		level[1]=1	level[2]=1 victim[1]=2 	level[3]=1
• •	k != i with level[k] >= L	)	victim[1]=1		
&& vict } }	im [L] == i ) {};				victim[1]=3 Blocked!
		Level	2	2	
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 2 i = 1		level[1]=2 victim[2]=1	level[2]=2 victim[2]=2 Blocked!	

		Thread	1	2	3
lock() (		Level	1	1	1
lock() {     for (int L = 1; L <         level[i] = L;     victim[L] = i;	< n; L++) {		level[1]=1	level[2]=1 victim[1]=2 	level[3]=1
while (( exists	k != i with level[k] >= L) im [L] == i ) {};		victim[1]=1 		
}					victim[1]=3
			<b>↓</b>	$\downarrow$	Blocked!
		Level	2	2	
Legend: n – total levels	Variables: n = 3		level[1]=2	level[2]=2 victim[2]=2	
L – current level i – thread id	L = 2 i = 1		victim[2]=1 Blocked!		
		Level		CriticalSection	

After critical section, Thread 2 calls

```
unlock() { level[2] = 0; }
```

...which unlocks Thread 1 (exists k != 1 with level[k])
 >= 2 is false now) => critical section

• Let's assume that Thread 3 is a huge slowpoke...

		Thread	1	2	3
lock() (		Level	1	1	1
lock() {     for (int L = 1; L <         level[i] = L;     victim[L] = i;	< n; L++) {		level[1]=1		level[3]=1
while (( exists && vict	k != i with level[k] >= L) im [L] == i ) {};		victim[1]=1 		
} }					victim[1]=3 Zzzzzz
		Level	2		
Legend:	Variables:		level[1]=2		
n – total levels L – current level	n = 3 L = CS		victim[2]=1		
i – thread id	i = 1	Level	<b>CriticalSection</b>		

		Thread	1	2	3
lock() (		Level	1	1	1
<pre>lock() {   for (int L = 1; L +     level[i] = L;     victim[L] = i;</pre>	< n; L++) {		level[1]=1	level[2]=1 victim[1]=2	level[3]=1
while (( exists	k != i with level[k] >= L) tim [L] == i ) {};		victim[1]=1 		
}					victim[1]=3 Zzzzz
		Level	2		
Legend:	Variables:		level[1]=2		
n – total levels L – current level i – thread id	n = 3 L = 1 i = 2		victim[2]=1		
		Level	<b>V</b> CriticalSection		

		Thread	1	2	3
lock() (		Level	1	1	1
lock() {     for (int L = 1; L -         level[i] = L;         victim[L] = i;	< n; L++) {		level[1]=1	level[2]=1 victim[1]=2 Blocked!	level[3]=1
• •	k != i with level[k] >= L) im [L] == i ) {};		victim[1]=1 		
<pre>} We still have Thread 3 sle } same level</pre>		eeping at the			victim[1]=3 Zzzzzz
		Level	2		
Legend: n – total levels	Variables: n = 3		level[1]=2		
L – current level i – thread id	i = 3 L = 1 i = 2		victim[2]=1 		
i direda id		Level	<b>↓</b> CriticalSection		

	Thread	1	2	3
lock() {	Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt; n; L++) {     level[i] = L;     victim[L] = i;     while (( exists k != i with level[k] &gt;= L</pre>	>= L)		level[2]=1 victim[1]=2 Blocked!	level[3]=1
Thread 1 making le change a				victim[1]=3 Zzzzzz
	Level			

Variables: Legend: n – total levels n = 3L – current level L = 1 i – thread id

i = 2

```
2
                                              Thread
                                                                                                               3
                                              Level
                                                                                         1
  lock() {
                                                                                                               level[3]=1
                                                                                         level[2]=1
  for (int L = 1; L < n; L++) {
                                                                                         victim[1]=2
    level[i] = L;
                                                                                           Blocked!
    victim[L] = i;
    while (( exists k != i with level[k] >= L)
                                                                    level[1]= 1
           && victim [L] == i ) {};
                                                                                                               victim[1]=3
                                                                                                                Zzzzzz...
                                              Level
Legend:
                     Variables:
n – total levels
                     n = 3
L – current level
                     L = 1
i – thread id
                     i = 1
```

	Thread	1	2	3
lock() (	Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt; n; L++) {     level[i] = L;     victim[L] = i;</pre>			level[2]=1 victim[1]=2 	level[3]=1
<pre>while (( exists k != i with level[k] &gt;= L)</pre>		level[1]= 1 victim[1] = 1 Blocked!		victim[1]=3
}				Zzzzzz
	Level		2	

		Thread	1	2	3
lock() (		Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt;     level[i] = L;     victim[L] = i;</pre>	< n; L++) {			level[2]=1 victim[1]=2 	level[3]=1
while (( exists	k != i with level[k] >= L)		level[1]= 1		
&& victi	im [L] == i ) {};		victim[1] = 1 Blocked!		
}			biocked!		victim[1]=3 Zzzzzz
		Level		2	
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 2 i = 2	Level		level[2]=2 victim[2]=2	

		Thread	1	2	3
lock() {		Level	1	1	1
lock() { for (int L = 1; L < level[i] = L; victim[L] = i;	< n; L++) {			level[2]=1 victim[1]=2 	level[3]=1
	<u>k != i with level[k] &gt;= L)</u> im [L] == i ) {};		level[1]= 1 victim[1] = 1 Blocked!		
}					victim[1]=3 Zzzzzz
		Level		2	
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 2 i = 2			level[2]=2 victim[2]=2 ↓	
		Level		CriticalSection	

```
Thread
                                                                                      2
                                                                                                            3
                                            Level
                                                                                      1
lock() {
                                                                 level[1]= 1
                                                                                                            level[3]=1
 for (int L = 1; L < n; L++) {
                                                                 victim[1] = 1
  level[i] = L;
                                                                   Blocked!
  victim[L] = i;
  while (( exists k != i with level[k] >= L)
         && victim [L] == i ) {};
                Thread 2 making level[2]=0 doesn't
                                                                                                            victim[1]=3
                          change anything
                                                                                                             Zzzzzz...
                                            Level
```

```
Thread
                                                                                         2
                                                                                                               3
                                              Level
                                                                                         1
  lock() {
                                                                                                               level[3]=1
                                                                    level[1]= 1
  for (int L = 1; L < n; L++) {
                                                                    victim[1] = 1
    level[i] = L;
                                                                      Blocked!
    victim[L] = i;
    while (( exists k != i with level[k] >= L)
                                                                                         level[2]=1
           && victim [L] == i ) {};
                                                                                                               victim[1]=3
                                                                                                                 Zzzzzz...
                                              Level
Legend:
                     Variables:
n – total levels
                     n = 3
L – current level
                     L = 1
```

Level

i – thread id

i = 2

	Thread	1	2	3
lock() {	Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt; n; L++) {     level[i] = L;     victim[L] = i;     while (( exists k != i with level[k] &gt;= L)</pre>		level[1]= 1 victim[1] = 1	level[2]=1 victim[1]=2 Blocked!	level[3]=1  victim[1]=3 Zzzzzz
	Level	2		

And so on and so forth...

• Slowpoke starves.

• End of story.

And so on and so forth...

• Slowpoke starves.

• End of story.

• Unless...

### Peterson's mutual exclusion – Fair Version

```
lock() {
 for (int L = 1; L < n; L++) {
   level[i] = L;
   victim[L] = i;
   while (( exists k != i with level[k] >= L)
                && victim [L] == i ) {};
 while ( exists k != i, level[k] != 0 &&
      victim[level[k]] != k ) {};
```

## Peterson's mutual exclusion – Fair Version

```
lock() {
  for (int L = 1; L < n; L++) {
    level[i] = L;
    victim[L] = i;
    while (( exists k != i with level[k] >= L)
                 && victim [L] == i ) {};
                                Aka "please wait for the less well..."
 while ( exists k != i, level[k] != 0 &&
       victim[level[k]] != k ) {};
```

	Thread	1	2	3
lock() (	Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt; n; L++) {     level[i] = L;     victim[L] = i;     while (( exists k != i with level[k] &gt;= L)         &amp;&amp; victim [L] == i ) {};</pre>		level[1]=1	level[2]=1 victim[1]=2	level[3]=1
<pre>} while ( exists k != i, level[k] != 0 &amp;&amp;   victim[level[k]] != k ) {};</pre>		victim[1]=1 Blocked!		victim[1]=3 Zzzzzz
}	Level		2	

```
Legend: Variables:

n - total levels n = 3

L - current level L = 1

i - thread id i = 1
```

		Thread	1	2	3
lock() (		Level	1	1	1
&& victi	< n; L++) { k != i with level[k] >= L) im [L] == i ) {};		level[1]=1	level[2]=1 victim[1]=2	level[3]=1
} while ( exists k ! victim[level[l	= i, level[k] != 0 && <]] != k ) {};	Level	victim[1]=1 Blocked!	2	victim[1]=3 Zzzzzz
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 2 i = 2	Level		level[2]=2 victim[2]=2	

		Thread	1	2	3
look() (		Level	1	1	1
	< n; L++) { <u>k != i with level[k] &gt;= L</u> tim [L] == i ) {};	)	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
victim[level[	!= i, level[k] != 0 && [k]] != k ) {};		victim[1]=1 Blocked!		victim[1]=3 Zzzzzz
}		Level		2	
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 2 i = 2			level[2]=2 victim[2]=2  <b>v</b>	
		Level		CriticalSection	

		Thread	1	2	3
la als/) (		Level	1	1	1
lock() {			level[1]=1	level[2]=1	level[3]=1
for (int L = 1; L	< n; L++) {		ievei[1]-1		16/6[[3]-1
level[i] = L;				victim[1]=2 I	
victim[L] = i;					
while (( exists	k = i  with level[k] >= L)				
&& vict	im [L] == i ) {};				
}					
while (exists k	!= i, level[k] != 0 &&		victim[1]=1		victim[1]=3
victim[level[	k]] != k ) {};		Blocked!	<b>↓</b>	Zzzzzz
}		Level		2	
We	have level[3] = 1 and victim[1] = 1	Level		2	
	victini[1] - 1				
Logondi	Variables			level[2]=2	
Legend:	Variables:			victim[2]=2	
n – total levels	n = 3			:	
L – current level	L = 2				
i – thread id	i = 2			Blocked!	
		Level		CriticalSection	

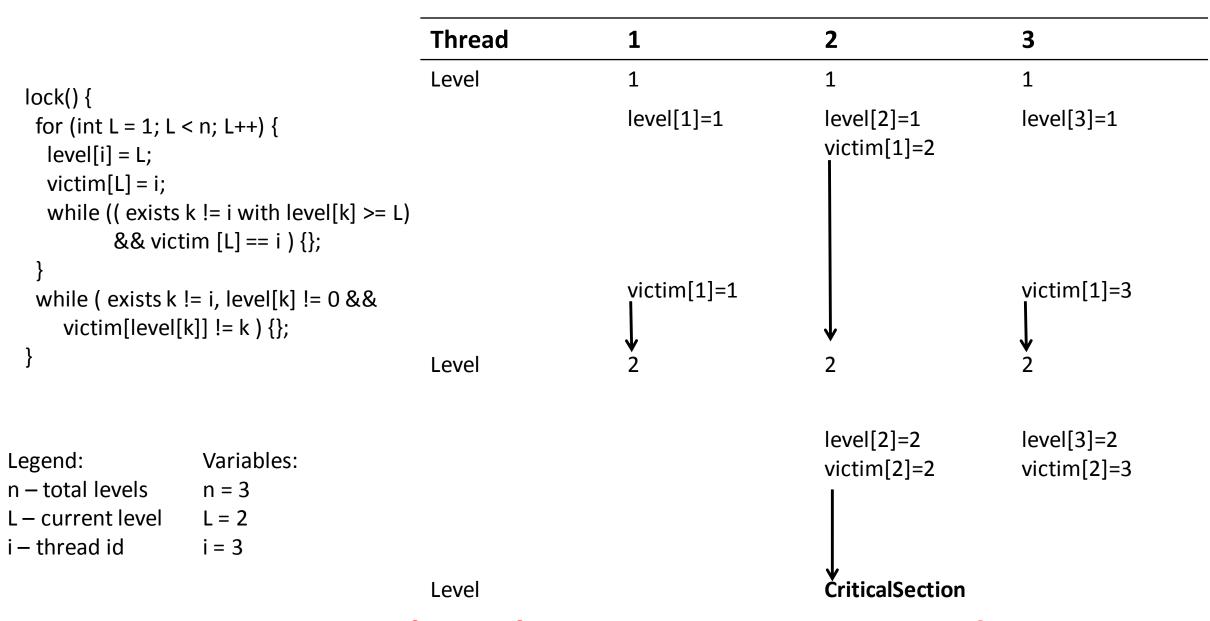
		Thread	1	2	3
look() (		Level	1	1	1
• •	<pre>c n; L++) { c != i with level[k] &gt;= L) m [L] == i ) {};</pre>		level[1]=1	level[2]=1 victim[1]=2	level[3]=1
while ( exists k ! victim[level[k }	= i, level[k] != 0 && <]] != k ) {};	Level	victim[1]=1 Blocked!	2	victim[1]=3 What time is it? Oh. O.o
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 1 i = 3	Level		level[2]=2 victim[2]=2 Blocked! CriticalSection	

		Thread	1	2	3
look() (		Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt; n; L++) {     level[i] = L;     victim[L] = i;     while (( exists k != i with level[k] &gt;= L)         &amp;&amp; victim [L] == i ) {}; }</pre>			level[1]=1 victim[1]=1	level[2]=1 victim[1]=2	level[3]=1 victim[1]=3
while ( exists k ! victim[level[k }	= i, level[k] != 0 && <]] != k ) {};	Level	Blocked!	2	• 1
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 1 i = 3			level[2]=2 victim[2]=2 v Blocked!	
		Level		CriticalSection	

		Thread	1	2	3
look/) (		Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt; n; L++) {     level[i] = L;     victim[L] = i;     while (( exists k != i with level[k] &gt;= L)         &amp;&amp; victim [L] == i ) {}; }</pre>			level[1]=1	level[2]=1 victim[1]=2	level[3]=1
while ( exists k ! victim[level[l }	= i, level[k] != 0 && <]] != k ) {};	Level	victim[1]=1 <b>V</b> 2	2	victim[1]=3 <b>↓</b> 2
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 2 i = 3			level[2]=2 victim[2]=2 v Blocked!	level[3]=2
		Level		CriticalSection	

		Thread	1	2	3
look() (		Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt; n; L++) {     level[i] = L;     victim[L] = i;     while (( exists k != i with level[k] &gt;= L)         &amp;&amp; victim [L] == i ) {}; }</pre>			level[1]=1	level[2]=1 victim[1]=2	level[3]=1
while (exists k! victim[level[k]	= i, level[k] != 0 && k]] != k ) {};	Level	victim[1]=1	2	victim[1]=3   
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = 2 i = 3			level[2]=2 victim[2]=2 v Blocked!	level[3]=2 victim[2]=3
		Level		CriticalSection	

		Thread	1	2	3
lock() (		Level	1	1	1
<pre>lock() {   for (int L = 1; L &lt; n; L++) {     level[i] = L;     victim[L] = i;     while (( exists k != i with level[k] &gt;= L)         &amp;&amp; victim [L] == i ) {}; }</pre>			level[1]=1	level[2]=1 victim[1]=2	level[3]=1
<pre>while ( exists k != i, level[k] != 0 &amp;&amp;     victim[level[k]] != k ) {};</pre>			victim[1]=1		victim[1]=3
} •	victim[level[1]] = 1 and victim[level[3] = 3	Level	2	2	2
Legend: n – total levels L – current level i – thread id	Variables: n = 3 L = CS i = 2			level[2]=2 victim[2]=2	level[3]=2 victim[2]=3
		Level		CriticalSection	



From here they start competing again!

#### Exercise

- More practice on linearizability and sequential consistency.
- Instructions and tasks in exercise\_lab6.pdf file on ILIAS in the lab6/exercises/ folder.