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

```
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) {};
  }}
```

1. there exists a thread (k) and its not the current one (i) …

2. and that thread is at the same or bigger level(level[k]) than the current thread (L)

3. and the victim at the level where the current thread is (victim[L]) is the current thread (i)

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	level[3]=1

Legend:
n – total levels
L – current level
i – thread id

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

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

Variables:
n = 3
L = 1
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
	victim[1]=1		

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 1

```
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
	victim[1]=1 **Blocked!**		
Level		2	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 1

```
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
	victim[1]=1 **Blocked!**		
			victim[1]=3
Level		2	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 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
	victim[1]=1		
			victim[1]=3 **Blocked!**
Level	2	2	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 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
	victim[1]=1		
			victim[1]=3 **Blocked!**
Level	2	2	
	level[1]=2	level[2]=2	

Legend:	Variables:
n – total levels	n = 3
L – current level	L = 2
i – thread id	i = 1,2

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

Legend:	Variables:
n – total levels	n = 3
L – current level	L = 2
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	level[3]=1
	victim[1]=1		
			victim[1]=3 **Blocked!**
Level	2	2	
	level[1]=2	level[2]=2 victim[2]=2 **Blocked!**	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
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	level[3]=1
	victim[1]=1		
			victim[1]=3 **Blocked!**
Level	2	2	
	level[1]=2	level[2]=2 victim[2]=2 **Blocked!**	
	victim[2]=1		

Legend:	Variables:
n – total levels	n = 3
L – current level	L = 2
i – thread id	i = 1

```
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
	victim[1]=1		
			victim[1]=3 **Blocked!**
Level	2	2	
	level[1]=2	level[2]=2 victim[2]=2	
	victim[2]=1 **Blocked!**		
Level		**CriticalSection**	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
i = 1

- 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...

```
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[3]=1
	victim[1]=1		
	↓		victim[1]=3 **Zzzzzz...**
Level	2		
	level[1]=2		
	victim[2]=1		
	↓		
Level	**CriticalSection**		

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = CS
i = 1

```
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
	victim[1]=1		
	↓		victim[1]=3 **Zzzzzz...**
Level	2		
	level[1]=2		
	victim[2]=1		
	↓		
Level	**CriticalSection**		

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
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 ) {};
  }
}
```

We still have Thread 3 sleeping at the same level

Thread	1	2	3
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2 **Blocked!**	level[3]=1
	victim[1]=1		
	↓		victim[1]=3 **Zzzzzz...**
Level	2		
	level[1]=2		
	victim[2]=1		
	↓		
Level	**CriticalSection**		

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
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 making level[1]=0 doesn't change anything

Thread	1	2	3
Level	1	1	1
		level[2]=1 victim[1]=2 **Blocked!**	level[3]=1
			victim[1]=3 **Zzzzzz...**
Level			
Level			

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
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[2]=1 victim[1]=2 **Blocked!**	level[3]=1
	level[1]= 1		
			victim[1]=3 **Zzzzzz...**
Level			
Level			

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 1

```
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[2]=1 victim[1]=2	level[3]=1
	level[1]= 1 victim[1] = 1 **Blocked!**		
			victim[1]=3 **Zzzzzz...**
Level		2	
Level			

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 1

```
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[2]=1 victim[1]=2	level[3]=1
	level[1]= 1 victim[1] = 1 **Blocked!**		
			victim[1]=3 **Zzzzzz...**
Level		2	
		level[2]=2 victim[2]=2	
Level			

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
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[2]=1 victim[1]=2	level[3]=1
	level[1]= 1 victim[1] = 1 **Blocked!**		
			victim[1]=3 **Zzzzzz...**
Level		2	
		level[2]=2 victim[2]=2	
Level		**CriticalSection**	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
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 2 making level[2]=0 doesn't change anything

Thread	1	2	3
Level	1	1	1
	level[1]= 1 victim[1] = 1 **Blocked!**		level[3]=1
			victim[1]=3 **Zzzzzz...**
Level			
Level			

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 1

```
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 victim[1] = 1 **Blocked!**		level[3]=1
		level[2]=1	
			victim[1]=3 **Zzzzzz...**
Level			
Level			

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
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 victim[1] = 1		level[3]=1
		level[2]=1 victim[1]=2 **Blocked!**	
			victim[1]=3 **Zzzzzz...**
Level	2		
Level			

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 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 ) {};
  }
  while ( exists k != i, level[k] != 0 &&
        victim[level[k]] != k ) {};
}
```

Aka "Please wait for the less gifted as well..."

```
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 ) {};
}
```

Thread	**1**	**2**	**3**
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=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 = 1
i = 1

```
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 ) {};
}
```

Thread	1	2	3
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
	victim[1]=1 **Blocked!**		victim[1]=3 **Zzzzzz...**
Level		2	
		level[2]=2 victim[2]=2	
Level			

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
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 ) {};
  }
  while ( exists k != i, level[k] != 0 &&
      victim[level[k]] != k ) {};
}
```

Thread	1	2	3
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
	victim[1]=1 **Blocked!**		victim[1]=3 **Zzzzzz...**
Level		2	
		level[2]=2 victim[2]=2	
Level		**CriticalSection**	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
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 ) {};
  }
  while ( exists k != i, level[k] != 0 &&
      victim[level[k]] != k ) {};
}
```

We have level[3] = 1 and victim[1] = 1

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
i = 2

Thread	**1**	**2**	**3**
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
	victim[1]=1 **Blocked!**		victim[1]=3 **Zzzzzz...**
Level		2	
		level[2]=2 victim[2]=2 **Blocked!**	
Level		**CriticalSection**	

```
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 ) {};
}
```

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 3

Thread	**1**	**2**	**3**
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
	victim[1]=1 **Blocked!**		victim[1]=3 **What time is it? Oh. O.o**
Level		2	
		level[2]=2 victim[2]=2 **Blocked!**	
Level		**CriticalSection**	

```
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 ) {};
}
```

Thread	1	2	3
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
	victim[1]=1 **Blocked!**		victim[1]=3
Level		2	2
		level[2]=2 victim[2]=2 **Blocked!**	
Level		**CriticalSection**	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 1
i = 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 ) {};
  }
  while ( exists k != i, level[k] != 0 &&
      victim[level[k]] != k ) {};
}
```

Thread	1	2	3
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
	victim[1]=1		victim[1]=3
Level	2	2	2
		level[2]=2 victim[2]=2	level[3]=2
		Blocked!	
Level		**CriticalSection**	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
i = 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 ) {};
  }
  while ( exists k != i, level[k] != 0 &&
      victim[level[k]] != k ) {};
}
```

Thread	1	2	3
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
	victim[1]=1		victim[1]=3
Level	2	2	2
		level[2]=2 victim[2]=2	level[3]=2 victim[2]=3
		Blocked!	
Level		**CriticalSection**	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
i = 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 ) {};
  }
  while ( exists k != i, level[k] != 0 &&
      victim[level[k]] != k ) {};
}
```

victim[level[1]] = 1 and victim[level[3] = 3

Legend:	Variables:
n – total levels	n = 3
L – current level	L = CS
i – thread id	i = 2

Thread	**1**	**2**	**3**
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
	victim[1]=1		victim[1]=3
Level	2	2	2
		level[2]=2 victim[2]=2	level[3]=2 victim[2]=3
Level		**CriticalSection**	

```
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 ) {};
}
```

Thread	1	2	3
Level	1	1	1
	level[1]=1	level[2]=1 victim[1]=2	level[3]=1
	victim[1]=1		victim[1]=3
Level	2	2	2
		level[2]=2 victim[2]=2	level[3]=2 victim[2]=3
Level		**CriticalSection**	

Legend:
n – total levels
L – current level
i – thread id

Variables:
n = 3
L = 2
i = 3

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.