Process synchronization: a few advanced topics

not for exam ☺

Indranil Sen Gupta (odd section)
and Mainack Mondal (even section)
CS39002

Spring 2019-20

Uses of CAS

(source: http://www.cs.cornell.edu/courses/cs4410/2015su/lectures/lec06-spin.html)

Compare and swap

```
int compare _and_swap(int *value, int expected, int
 new_value) {

        int temp = *value;

        if (*value == expected)

           *value = new_value;

     return temp;

    }
```

How is CAS used?

- Optimistic transactional data structures
  - all updates are performed on a copy of the data structure
  - when the operations are finished, a compare and swap replace the data structure in one fell swoop

Example: concurrent balanced binary search tree

```
Shared state:
    root = pointer to the root of the tree
Insert code:
    do
        old_root = root
        new_root = new Tree
        # copy old_root into new_root
        # do insertion into new_root
    while compare_and_swap (root, old_root, new_root) == old_root
Balance code:
    do
        old_root = root
        new_root = balanced_copy_of (old_root)
    while compare_and_swap (root, old_root, new_root) == old_root
```

How does it work?

- If an insertion is performed while a balance is in progress
  - update the root to point to its new root
  - When the balancing thread completes, the CAS will fail
- If balance finishes before the insertion
  - CAS in the insertion code will fail
  - insertion will be retried on the new balanced root

Locking in Linux kernel : pre-emptive

(source: http://www.informit.com/articles/article.aspx?p=101760&seqNum=3)

What does it mean?

- In a non-preemptive kernel
  - Code runs till completion
  - Scheduler is not capable of rescheduling a process while it is in the kern
  - Kernel code is scheduled cooperatively, not preemptively
  - version 2.6 onwards this is NOT the case with linux

- Good : Linux kernel is Symmetric Multi Processing (SMP) safe
  - In other words "thread safe"

How does pre-emption happen in implementation?

- Addition of a preemption counter, preempt_count, to each process's task_struct
  - Counter begins at zero
  - Increments for each lock that is acquired
  - Decrements for each lock that is released
  - When the counter is zero, the kernel is preemptible

How does pre-emption happen in implementation? (contd.)

- The kernel checks the values of **`need_resched`** and **`preempt_count`**
  - If **`need_resched`** is set and **`preempt_count`** is zero

  => more important task is runnable and it is safe to preempt
  - Then the scheduler is onvoked
  - If **`preempt_count`** is nonzero, a lock is held and it is unsafe to reschedule

How is mutex implemented in linux?

Resources

- https://en.wikipedia.org/wiki/Futex
- https://linux.die.net/man/2/futex
- https://stackoverflow.com/a/5870415
- https://eli.thegreenplace.net/2018/basics-of-futexes/