Process Synchronization

@Wxy2003-xy

1 Producer-consumer problem

The process producing information and the process that consumes the information to be produced by the coorperating process are accessing shared memory. To run the producer and consumer processes concurrently, a buffer is used.

- Bounded buffer: a fixed size buffer, where the consumer must wait when the buffer is empty, and the producer must wait if the buffer is full.
- Unbounded buffer: a buffer without fixed size, the consumer may have to wait for new items, but the producer can always produce new items.

Protential Issue:

When using a counter to keep track the number of items in a bounded buffer (or the size of an unbounded buffer), concurrent `produce()` and `consume()` can lead to data inconsistency, and increment/decrement operations are not atomic on the assembly level.

Critical section:

A segment of a process that modifies shared resources e.g. global flags, shared memory. No more than 1 process can be in the critical section, i.e. no concurrent modification/access of shared resources.

- Each process must request permission to enter the CS
- Entry section: handling request, start of a CS
- Exit section: the end of a CS

Requirements

1. Mutual exclusion: No concurrent execution of the critical section.
2. Progress: No ready process wanting to enter the CS should not be indefinitely deferred when no other processes are in the CS.
3. Bounded wait:

Naive software level implementation

Process 0

```
1    bool* lock = (bool*) shmat(sid, NULL, 0);
2    *lock = false;
3
4    while (!lock);
5    *lock = true;
6    // critical section
7    *lock = false;
```

Process 1

```
1
2
3
4    while (!lock);
5    *lock = true;
6    // critical section
7    *lock = false;
```

Problem with this implementation is that the locking and unlocking operations are not atomic.

Process 0

```
1  # initialization...
2  loop:
3      la $t0, LOCK_ADDR
4      lw $t1, $t0, 0
5      bne $t1, $0, loop
6      sw $t0, 1
7  # Critical Section
8      la $t0, LOCK_ADDR
9      sw $t0, 1
```

Process 1

```
1
2  loop:
3      la $t0, LOCK_ADDR
4      lw $t1, $t0, 0
5      bne $t1, $0, loop
6      sw $t0, 1
7  # Critical Section
8      la $t0, LOCK_ADDR
9      sw $t0, 1
```

Peterson's Solution

Test and Set lock
