Seminarie 2

Operativsystem

Henrik Borg

January 5, 2020

Or...

No problem - On one condition
Do not share resources

2.4 A small test

How do we create the queue of threads?
The thread structure contains links to the next thread to run (struct green_t *next;). The first element in the list is the one currently running, there is a local static variable for this (static green_t *running;).

How to add a thread to the end of the queue?
We can either traverse the list//

```
while(NULL != object->next)
        object = object->next;
object->next = last_thread;
```

Or we kan keep track of the end of the list and directly add a thread to the end of the list. In this example we also take care of the corner case then the list is empty.

```
if(NULL == ready_queue_end) {
  ready_queue_end = new;
  ready_queue_end->next = new;
  running->next = new;
} else {
  ready_queue_end->next = new;
  ready_queue_end = new;
}
```

Do we need to keep track of the end of the queue?
The simple answer here is no, and yes. It depends on if we need to do it in a constant and very well defined length of time? If we are develing an application for a real time system with hard dead lines, then we have to perform calculation in a formal way to prove that the system will survive for ever, or at least till it is put out of use for other reasons then bad development. In this case we want to do it in a constant and very well define length of time, in clock cycles. Or if we are delevoping a system with a huge amount of threads so the time it takes to traverse huge queue of threads will have a bad enungh impact on the over all system, then we also want to do it in constant time. We can do it in constant time by keeping track of the end of the list.

3 Suspending on a condition

4 Adding a timer interrupt

5 A mutex lock

6 The final touch

7 Summary