HOGESCHOOL ROTTERDAM

Knowledge Clip

Embedded Systems

Pthread
Problem with Shared Memory

brojz@hr.nl

Problem with Shared Memory

Source: pthread_shared.c

```
volatile int aantal = 0;

void *teller(void *par) {
    for (int i = 0; i < 10000000; i++) {
        aantal++;
    }
    return NULL;
}

//…
    pthread_create(&t1, &pta, &teller, NULL);
    pthread_create(&t2, &pta, &teller, NULL);
    pthread_create(&t3, &pta, &teller, NULL);
```

What is the final value of aantal?

Problem with Shared Memory

- The operation `aantal++` is **not atomic** (in machine code).
  - For example, `X10` contains the address of `aantal`:

```
LDUR X9, [X10, #0]
ADDI X9, X9, #1
STUR X9, [X10, #0]
```

What happens when a task switch occurs at this moment?

- What is the minimal and the maximal final value of aantal?
  - Minimum = 10000000
  - Maximum = 30000000

Solution?

- There are solutions which use shared variables (2 flags and 1 turn variable) and **busy waiting**.
  - Dekker's algorithm: http://en.wikipedia.org/wiki/Dekker's_algorithm
  - Peterson's algorithm: http://en.wikipedia.org/wiki/Peterson's_algorithm
- Busy waiting **costs** clock cycles!
- OSes offer solutions **without** busy waiting.

IPC Inter Process (Task) Communication

- Shared variable based
  - Busy waiting
    - Inefficient
    - Mutual exclusion is hard (Dekker’s or Peterson’s algorithm)
  - Spinlock
    - Busy waiting
  - Mutex
  - Semaphore
  - Monitor
    - Mutex combined with Conditional variables
  - Barrier
  - Read Write Lock
  - Event Groups

- Message based
  - Message Queue