Karlsruhe Institute of Technology

Operating Systems 2016/17
Tutorial-Assignment 7

Prof. Dr. Frank Bellosa
Dipl.-Inform. Marc Rittinghaus

Question 7.1: Race Conditions

a. Explain the term *race condition* with this scenario: Two people try to access a bank account simultaneously. One person tries to deposit 100 Euros, while another wants to withdraw 50 Euros. These actions trigger two update operations in a central bank system. Both operations run in "parallel" on the same computer, each represented by a single thread executing the following code:

```
current = get_balance();
current += delta;
set_balance(current);
```

where `delta` is either 100 or -50 in our example.

b. Determine the lower and upper bounds of the final value of the shared variable `tally` as printed in the following program:

```
const int N = 50;
int tally;


void total ()
{
    for( int i = 0; i < N; ++i )
        tally += 1;
}
```

```
int main ()
{
    tally = 0;

    #pragma omp parallel for
    for( int i = 0; i < 2; ++i )
        total();

    printf( "%d\n", tally );
    return 0;
}
```

Assume that threads can execute at any relative speed and that a value can only be incremented after it has been loaded into a register by a separate machine instruction.

c. Suppose that an arbitrary number $t > 2$ of parallel threads are performing the above procedure `total`. What (if any) influence does the value of t have on the range of the final values of `tally`?

d. Now suppose userlevel threads (i.e., the many-to-one model) were used. Would this change make a difference to the output?

e. Finally consider a modified `total` routine:

```
void total ()
{
    for( int i = 0; i < N; ++i )
    {
        tally += 1;
        sched_yield();
    }
}
```

What will be printed in the one-to-one model, when a voluntary yield is added?

Question 7.2: Critical Sections

a. Explain the terms *critical section*, *entry section*, *exit section*, and *remainder section*.

b. Enumerate and explain the requirements for a valid synchronization solution.

c. Recap the banking example from the previous question. How could the race condition be avoided?

Question 7.3: Synchronization Primitives

a. Distinguish the various types of synchronization objects and summarize their respective operations' semantics: spinlocks, counting semaphores, binary semaphores, and mutex objects.

Question 7.4: Producer-Consumer Problem

a. Solve the producer-consumer problem for the following buffer using a single pthread mutex and two semaphores:

```
#define BUFFER_SIZE 10
int buffer[BUFFER_SIZE];
int index = 0; // Current element in buffer
```