Operating Systems
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: 320202
Date: 2010-02-24
Time: 10 min.

Quiz Sheet #2

Problem 2.1: *concurrency and synchronization* (1+1+1+1 = 4 points)

Please mark whether the following statements are true or false. Hint: read carefully.

a) A race condition exists if the result produced by concurrent processes (or threads) depends unexpectedly on the order of the execution of the processes (or threads).

b) A spinlock is a lock where a process (or thread) waits in a loop ("spins") repeatedly checking until the lock becomes available.

c) POSIX mutexes and condition variables can be used to implement semaphores in user space.

d) Some programming languages provide language support (e.g., the Java `synchronized` keyword) that automate allocation of mutexes and thus help to avoid some programming errors.

Solution:

a) True

b) True

c) True

d) True

Problem 2.2: *condition variables* (3+3 = 6 points)

The following solution has been developed for the unisex bathroom problem (just showing the `male()` function, the `female()` function is similar).

```
typedef struct bathroom {
    unsigned int    males;
    unsigned int    females;
    pthread_mutex_t mutex;
    pthread_cond_t  full;
} bathroom_t;

static bathroom_t shared_bathroom = {
    .males = 0,
    .females = 0,
    .mutex = PTHREAD_MUTEX_INITIALIZER,
    .full  = PTHREAD_COND_INITIALIZER
};

static void* male(void *data)
{
    bathroom_t *bathroom = (bathroom_t *) data;

    while (1) {
        enjoy_live(100);
        pthread_mutex_lock(&bathroom->mutex);
        while (bathroom->females > 0 || bathroom->males > 2) {
            pthread_cond_wait(&bathroom->full, &bathroom->mutex);
        }
        bathroom->males++;
        pthread_mutex_unlock(&bathroom->mutex);
        use_bathroom(bathroom, 10);
        pthread_mutex_lock(&bathroom->mutex);
        bathroom->males--;
        pthread_cond_signal(&bathroom->full);
        pthread_mutex_unlock(&bathroom->mutex);
    }
    return NULL;
}
```

a) Why is `&bathroom->mutex` passed as the second argument of `pthread_cond_wait()`?

b) Can the function calls `pthread_cond_signal()` and `pthread_mutex_unlock()` in `male()` be swapped without breaking the program? Explain why or why not.

Solution:

a) The mutex passed as the second argument to `pthread_mutex_unlock()` is released while waiting on the condition variable. It will be reacquired before the `pthread_mutex_unlock()` function returns.

b) Swapping the order of these function calls has no effect on the correctness of the program. A thread waiting on the condition variable will have to acquire the mutex lock and thus the signal is practically delayed until the mutex has been released even if the condition variable is signalled before releasing the mutex.