Operativsystemer og multiprogrammering
Handin 3

Malte Stær Nissen and
Jacob Daniel Kirstejn Hansen

March 7, 2011

Contents

1 Overall assumptions

We've made the following assumption on which our implementation depends:

- The programs are small enough to be executed on the system.

2 Design decisions

In order to maintain atomic access to our locks, we chose to implement them by disabling interrupts and using spinlocks. This means that every time we acess the state variable of a given lock we first acquire the spinlock for the lock we wish to access. We then access the state of the lock and perform the desired changes followed by a release of the spinlock at the end. Furthermore we encapsulate the entire access to the spinlocks and the lock state with disabling of interrupts. We simply follow the guidelines given in the Buenos Roadmap, page 29.

We similarily disable interrupts in the implementation of the conditions. We don't use spinlocks in the implementation of the conditions though since it doesn't make any sense to do so.

We've chosen to give the syscalls for lock functions the hex values 0x301, 0x302 and 0x303. This means that the only syscalls starting with 0x3 are the lock functions. To follow this decision we chose to give the conditional locks hex values starting with 0x4, hence the values are 0x401, 0x402, 0x403 and 0x404.

3 Important observations

When we test out the conditions and locks using the test program given in tests/threads_ring.c, the program starts failing if we set it to run for too many rounds. If we eg. set the number of rounds to 10, the program stops at the 7th round and stalls. We assume that this is because of a lack of memory and hence our assumption about programs being small enough to run on the system isn't met.

4 List of changes

- kernel/lock_cond.h, kernel/lock_cond.c: Locks and conditions implemented.
- kernel/module.mk: lock_cond.c added to the list of files to be compiled.
- proc/process.c, proc/process.h: Fork function implemented.

- proc/syscall.c, proc/syscall.h: System calls for locks, conditions and fork implemented. Unique hex values for systemcalls defined (in the .h file).
- tests/lib.h, test/lib.c: System calls implemented here as well.
- tests/threads.c, tests/threads_locks.c, tests/threads_ring.c: Tests for fork, locks and conditions added.
- compile: Bash script for cleaning everything up, compiling, deleting hdd, creating new hdd and adding all tests programs.