CS495 OS Lab
Lowlevel Synchronization in the Linux Kernel
A two minute Tour de Force

Abhineet Sawa, Stephan Seyboth

13.11.2003

Linux Kernels $\leq$ 2.2

UP Systems:

- no process running in kernel mode will be pre-empted by another process
- interrupts may pre-empt a kernel running process, but will always return to that process
- an interrupt can only be pre-empted by another interrupt
- $\Rightarrow$ disable interrupts to guarantee atomicity

SMP Systems:

- single spinlock across all processors protects all kernel entry points: syscall, interrupts...
- spinlock needs to be reentrant!
- only one processor executing kernel code at any given time
- $\Rightarrow$ effectively same guarantees as on UP systems

Linux Kernels $\geq$ 2.4

Fine Grained SMP Locking:

- no more global kernel lock
- spinlock primitives: spin_lock()/spin_unlock()
- avoid deadlocks with interrupt handlers:
  spin_lock_irqsave()/spin_unlock_irqrestore()

UP Systems:

- no spinlocks necessary
- spin_lock()/spin_unlock() do nothing
- irq versions compile to local_irq_save()/local_irq_restore()

Implementing Spinlocks:

- lots of macros, inlines and assembly in
  include/linux/spinlock.h (architecture independent)
  include/asm-$ARCH/spinlock.h (architecture dependent)

- architecture dependant implementation of atomic primitives in
  include/asm-$ARCH/atomic.h:
  atomic_add(), atomic_sub(), atomic_dec_and_test(),...