Synchronization in Linux

- Atomic operations
  - Operatores on atomic_t, which should be 24 bits (because that is what you can do in the Sparc)
    - ATOMIC_INIT(int I)
    - atomic_read()
    - atomic_set()
    - atomic_add()
    - test_and_set_bit()

```
include/asm-i386/atomic.h:
static __inline__ void atomic_add(int i, atomic_t *v)
{
        __asm__ __volatile__(
                LOCK "addl %1,%0"
                :"=m" (v->counter)
                :"ir" (i), "m" (v->counter));
}
```

Itanium

include/asm-ia64/atomic.h

```
static __inline__ int
ia64_atomic64_add (__s64 i, atomic64_t *v)
{
        __s64 old, new;
        CMPXCHG_BUGCHECK_DECL

        do {
                CMPXCHG_BUGCHECK(v);
                old = atomic_read(v);
                new = old + i;
        } while (ia64_cmpxchg(acq, v, old, new, sizeof(atomic64_t))
     != old);
        return new;
}
```

Spin locks

- Check in include/asm-ia64/spinlock.h
  - Architecture dependent way to spinlock
- Spinlocks can be used in interrupt handlers
  - Disable other interrupts
    - spin_lock_irqsave()
    - spin_unlock_irqrestore()
- Reader writer spin locks
  - Gives preference to readers over writers

Semaphore

- Linux semaphores are sleeping locks
- Reader-write semaphores

- Condition variables or completion variables

- asm/semaphore.h

Kernel preemption

- Preempt_disable()
- Preempt_enable()
- Preempt_enable_no_resched()

Linux futex

- Fast user level mutex: does not have to go to kernel space in the normal execution path
- Not user friendly, expected to be used by libraries