Looking at the Sources

Kernel boot starts with the execution of real mode assembly code living in the *arch/x86/boot/* directory. Look at *arch/x86/kernel/setup_32.c* to see how the protected mode kernel obtains information gleaned by the real mode kernel.

The first boot message is printed by code residing in *init/main.c.* Dig inside *init/calibrate.c* to learn more about BogoMIPS calibration and *include/asm-your-arch/bugs.h* for an insight into architecture-dependent checks.

Timer services in the kernel consist of architecture-dependent portions that live in *arch/your-arch/kernel/* and generic portions implemented in *kernel/timer.c.* For related definitions, look at the header files, *include/linux/time*.h.*

`jiffies` is defined in *linux/jiffies.h.* The value for `HZ` is processor-dependent and can be found in *include/asm-your-arch/param.h.*

Memory management sources reside in the top-level *mm/* directory.

Table 2.1 contains a summary of the main data structures used in this chapter and the location of their definitions in the source tree. Table 2.2 lists the main kernel programming interfaces that you used in this chapter along with the location of their definitions.

TABLE 2.1 Summary of Data Structures

Data Structure	Location	Description
`HZ`	*include/asm-your-arch/param.h*	Number of times the system timer ticks in 1 second
`loops_per_jiffy`	*init/main.c*	Number of times the processor executes an internal delay-loop in 1 jiffy
`timer_list`	*include/linux/timer.h*	Used to hold the address of a routine that you want to execute at some point in the future
`timeval`	*include/linux/time.h*	Timestamp
`spinlock_t`	*include/linux/spinlock_types.h*	A busy-locking mechanism to ensure that only a single thread enters a critical section
`semaphore`	*include/asm-your-arch/semaphore.h*	A sleep-locking mechanism that allows a predetermined number of users to enter a critical section
`mutex`	*include/linux/mutex.h*	The new interface that replaces `semaphore`
`rwlock_t`	*include/linux/spinlock_types.h*	Reader-writer spinlock
`page`	*include/linux/mm_types.h*	Kernel's representation of a physical memory page

TABLE 2.2 Summary of Kernel Programming Interfaces

Kernel Interface	Location	Description

`time_after()`
`time_after_eq()`
`time_before()`
`time_before_eq()`
	include/linux/jiffies.h	Compares the current value of `jiffies` with a specified future value

`schedule_timeout()`	*kernel/timer.c*	Schedules a process to run after a specified timeout has elapsed
`wait_event_timeout()`	*include/linux/wait.h*	Resumes execution if a specified condition becomes true or if a timeout occurs
`DEFINE_TIMER()`	*include/linux/timer.h*	Statically defines a timer
`init_timer()`	*kernel/timer.c*	Dynamically defines a timer
`add_timer()`	*include/linux/timer.h*	Schedules the timer for execution after the timeout has elapsed
`mod_timer()`	*kernel/timer.c*	Changes timer expiration
`timer_pending()`	*include/linux/timer.h*	Checks if a timer is pending at the moment
`udelay()`	*include/asm-your-arch/delay.h* *arch/your-arch/lib/delay.c*	Busy-waits for the specified number of microseconds
`rdtsc()`	*include/asm-x86/msr.h*	Gets the value of the TSC on Pentium-compatible processors
`do_gettimeofday()`	*kernel/time.c*	Obtains wall time
`local_irq_disable()`	*include/asm-your-arch/system.h*	Disables interrupts on the local CPU
`local_irq_enable()`	*include/asm-your-arch/system.h*	Enables interrupts on the local CPU
`local_irq_save()`	*include/asm-your-arch/system.h*	Saves interrupt state and disables interrupts
`local_irq_restore()`	*include/asm-your-arch/system.h*	Restores interrupt state to what it was when the matching `local_irq_save()` was called
`spin_lock()`	*include/linux/spinlock.h* *kernel/spinlock.c*	Acquires a spinlock.
`spin_unlock()`	*include/linux/spinlock.h*	Releases a spinlock
`spin_lock_irqsave()`	*include/linux/spinlock.h* *kernel/spinlock.c*	Saves interrupt state, disables interrupts and preemption on local CPU, and locks their critical section to regulate access by other CPUs
`spin_unlock_irqrestore()`	*include/linux/spinlock.h* *kernel/spinlock.c*	Restores interrupt state and preemption and releases the lock
`DEFINE_MUTEX()`	*include/linux/mutex.h*	Statically declares a mutex

Continues

TABLE 2.2 Continued

Kernel Interface	Location	Description
`mutex_init()`	*include/linux/mutex.h*	Dynamically declares a mutex
`mutex_lock()`	*kernel/mutex.c*	Acquires a mutex
`mutex_unlock()`	*kernel/mutex.c*	Releases a mutex
`DECLARE_MUTEX()`	*include/asm-your-arch/semaphore.h*	Statically declares a semaphore
`init_MUTEX()`	*include/asm-your-arch/semaphore.h*	Dynamically declares a semaphore
`up()`	*arch/your-arch/kernel/semaphore.c*	Acquires a semaphore
`down()`	*arch/your-arch/kernel/semaphore.c*	Releases a semaphore

`atomic_inc()`
`atomic_inc_and_test()`
`atomic_dec()`
`atomic_dec_and_test()`
`clear_bit()`
`set_bit()`
`test_bit()`
`test_and_set_bit()`
	include/asm-your-arch/atomic.h	Atomic operators to perform light-weight operations

`read_lock()`
`read_unlock()`
`read_lock_irqsave()`
`read_lock_irqrestore()`
`write_lock()`
`write_unlock()`
`write_lock_irqsave()`
`write_lock_irqrestore()`
	include/linux/spinlock.h
	kernel/spinlock.c
		Reader-writer variant of spinlocks

`down_read()`
`up_read()`
`down_write()`
`up_write()`
	kernel/rwsem.c	Reader-writer variant of semaphores

`read_seqbegin()`
`read_seqretry()`
`write_seqlock()`
`write_sequnlock()`
	include/linux/seqlock.h	Seqlock operations

`kmalloc()`	*include/linux/slab.h* *mm/slab.c*	Allocates physically contiguous memory from `ZONE_NORMAL`
`kzalloc()`	*include/linux/slab.h* *mm/util.c*	Obtains zeroed kmalloced memory
`kfree()`	*mm/slab.c*	Releases kmalloced memory
`vmalloc()`	*mm/vmalloc.c*	Allocates virtually contiguous memory that is not guaranteed to be physically contiguous.