Betriebssysteme und Systemnahe Programmierung

Kapitel 6 • MINIX Internals

Winter 2016/17

Marcel Waldvogel

The Internal Structure of MINIX

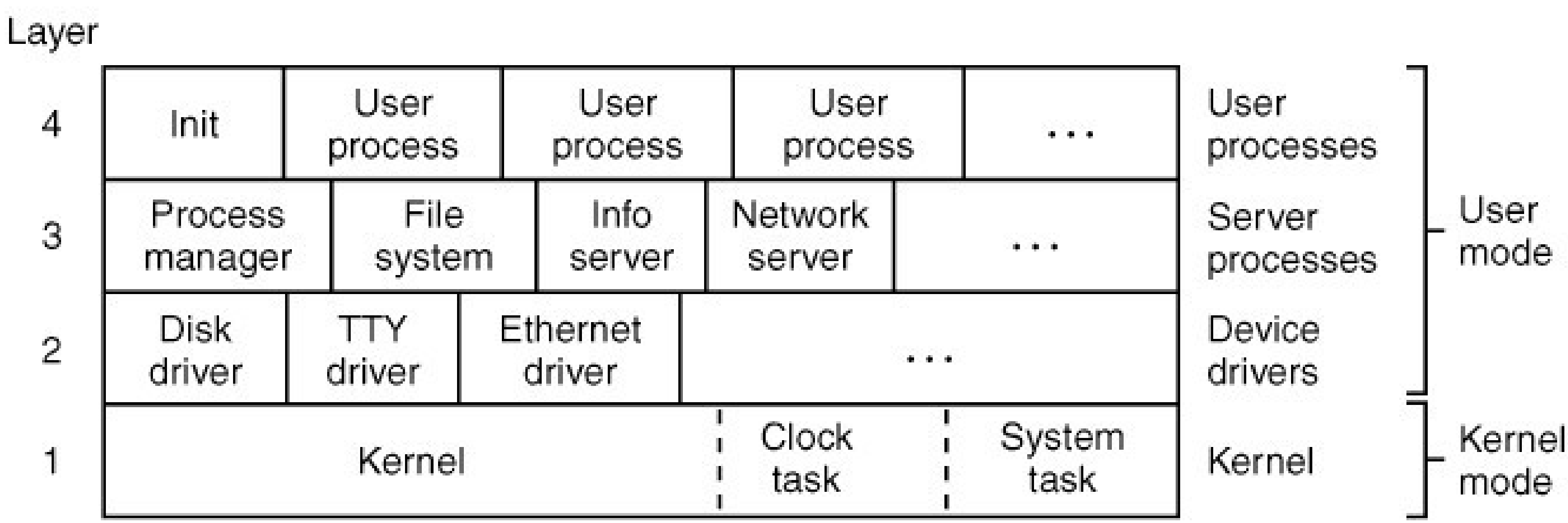


Figure 2-29. MINIX 3 is structured in four layers. Only processes in the bottom layer may use privileged (kernel mode) instructions.

MINIX 3 Startup

Component	Description	Loaded by
kernel	Kernel + clock and system tasks	(in boot image)
pm	Process manager	(in boot image)
fs	File system	(in boot image)
rs	(Re)starts servers and drivers	(in boot image)
memory	RAM disk driver	(in boot image)
log	Buffers log output	(in boot image)
tty	Console and keyboard driver	(in boot image)
driver	Disk (at, bios, or floppy) driver	(in boot image)
init	parent of all user processes	(in boot image)
floppy	Floppy driver (if booted from hard disk)	/etc/rc
is	Information server (for debug dumps)	/etc/rc
cmos	Reads CMOS clock to set time	/etc/rc
random	Random number generator	/etc/rc
printer	Printer driver	/etc/rc

Figure 2-30. Some important MINIX 3 system components.

Others such as an Ethernet driver and the inet server may also be present.

MINIX Memory (1)

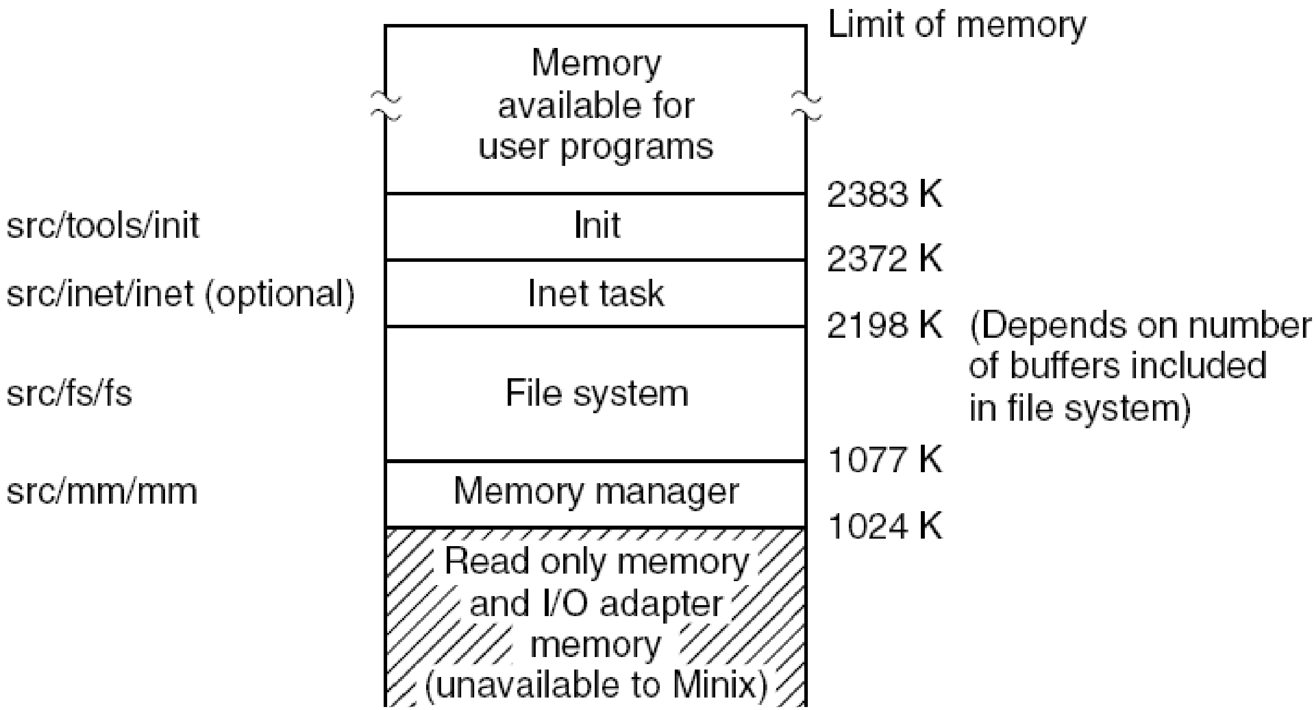


Figure 2-31. Memory layout after MINIX 3 has been loaded from the disk into memory. The kernel, servers, and drivers are independently compiled and linked programs, listed on the left. Sizes are approximate and not to scale.

MINIX Memory (2)

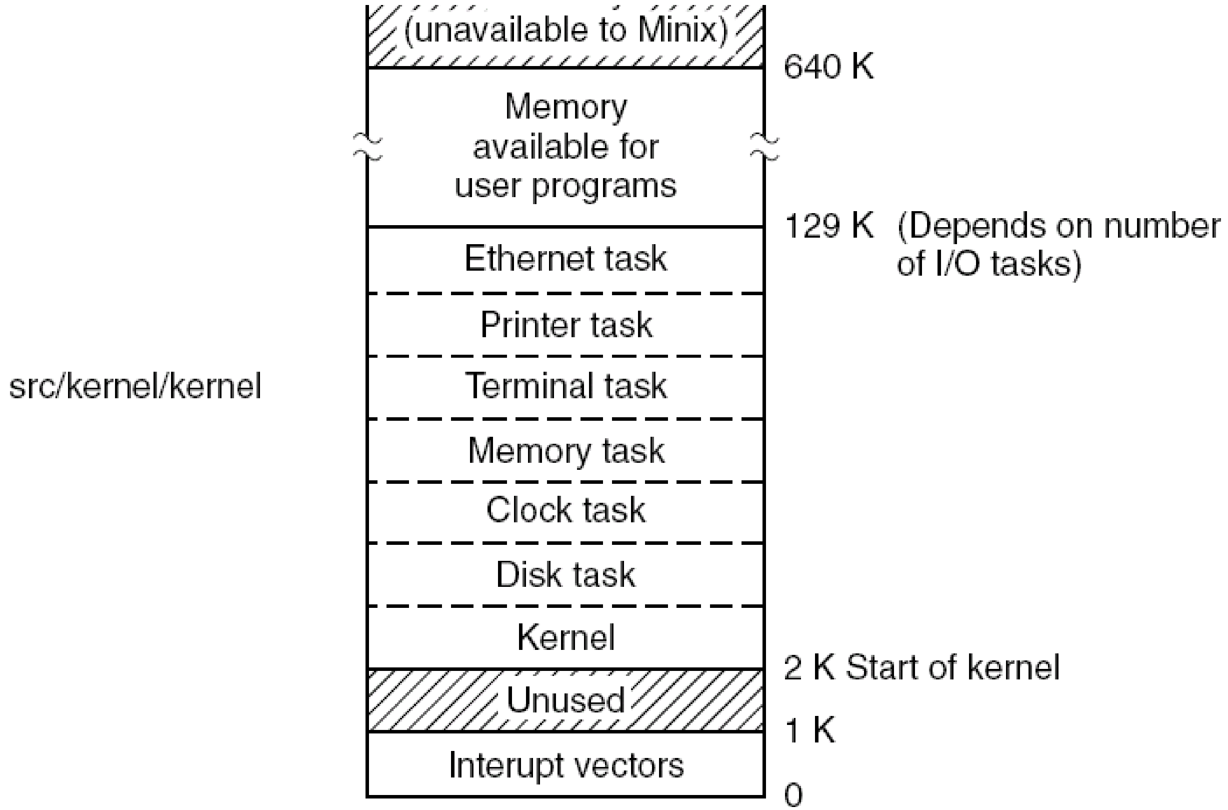


Figure 2-31. Memory layout after MINIX 3 has been loaded from the disk into memory. The kernel, servers, and drivers are independently compiled and linked programs, listed on the left. Sizes are approximate and not to scale.

MINIX Header File

```
#include <minix/config.h>        /* MUST be first */
#include <ansi.h>                /* MUST be second */
#include <sys/types.h>
#include <minix/const.h>
#include <minix/type.h>
#include <limits.h>
#include <errno.h>
#include <minix/syslib.h>
#include "const.h"
```

Figure 2-32. Part of a master header which ensures inclusion of header files needed by all C source files. Note that two *const.h* files, one from the *include/* tree and one from the local directory, are referenced.

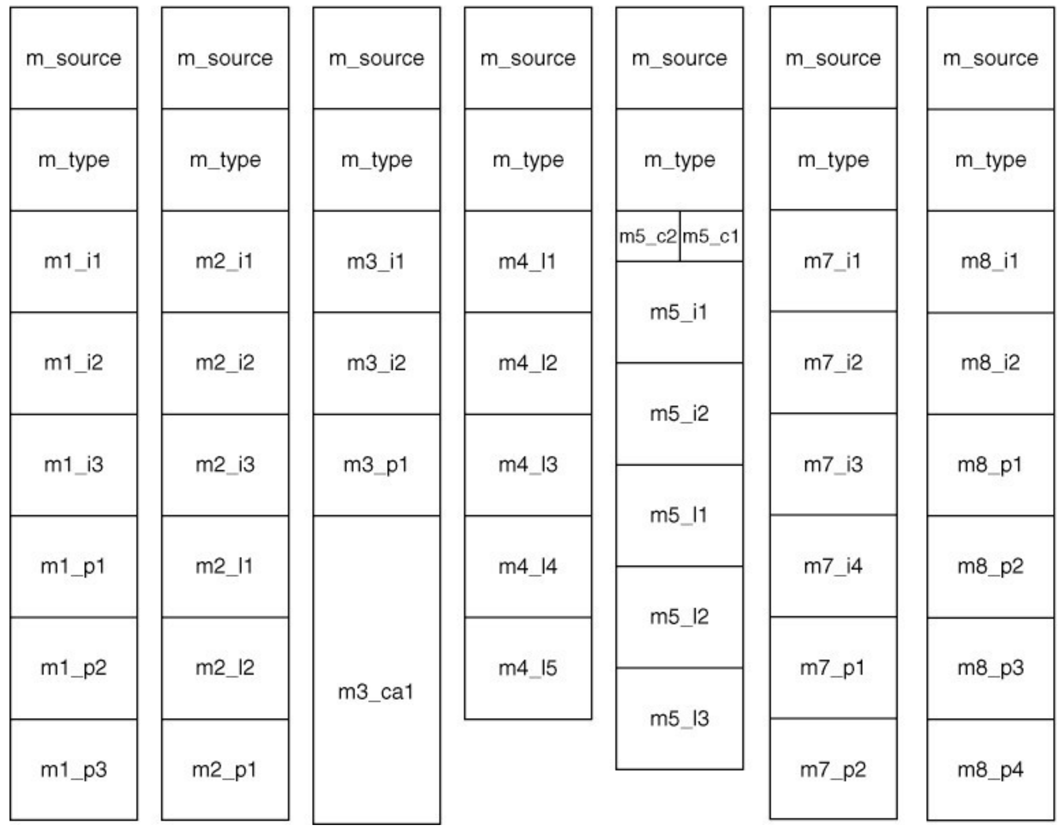
Sizes of Types in MINIX

Type	16-Bit MINIX	32-Bit MINIX
gid_t	8	8
dev_t	16	16
pid_t	16	32
ino_t	16	32

Figure 2-33. The size, in bits, of some types on 16-bit and 32-bit systems.

MINIX Message Types

Figure 2-34. The seven message types used in MINIX 3. The sizes of message elements will vary, depending upon the architecture of the machine; this diagram illustrates sizes on CPUs with 32-bit pointers, such as those of Pentium family members.



Debug Dump

--nr-	-id-	-name-	-flags-	-traps-	-ipc_to mask ------
(-4)	(01)	IDLE	P-BS-	-----	00000000 00001111
[-3]	(02)	CLOCK	---S-	--R--	00000000 00001111
[-2]	(03)	SYSTEM	---S-	--R--	00000000 00001111
[-1]	(04)	KERNEL	---S-	-----	00000000 00001111
0	(05)	pm	P--S-	ESRBN	11111111 11111111
1	(06)	fs	P--S-	ESRBN	11111111 11111111
2	(07)	rs	P--S-	ESRBN	11111111 11111111
3	(09)	memory	P--S-	ESRBN	00110111 01101111
4	(10)	log	P--S-	ESRBN	11111111 11111111
5	(08)	tty	P--S-	ESRBN	11111111 11111111
6	(11)	driver	P--S-	ESRBN	11111111 11111111
7	(00)	init	P-B--	E--B-	00000111 00000000

Figure 2-35. Part of a debug dump of the privilege table. The clock task, file server, tty, and init processes privileges are typical of tasks, servers, device drivers, and user processes, respectively. The bitmap is truncated to 16 bits.

Bootstrapping MINIX (1)

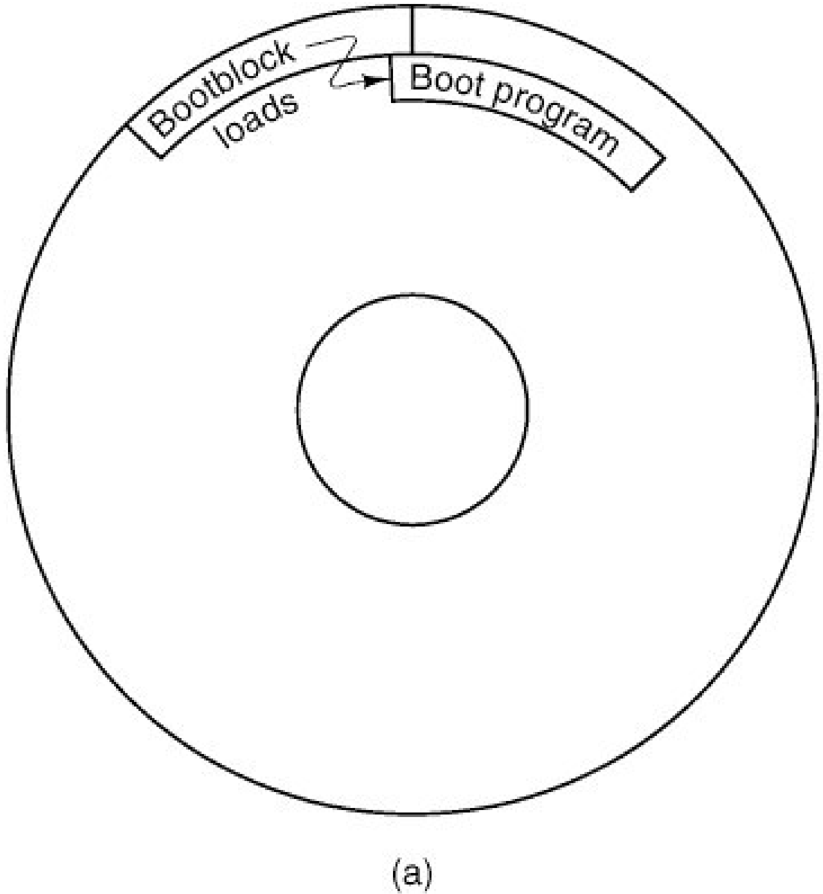
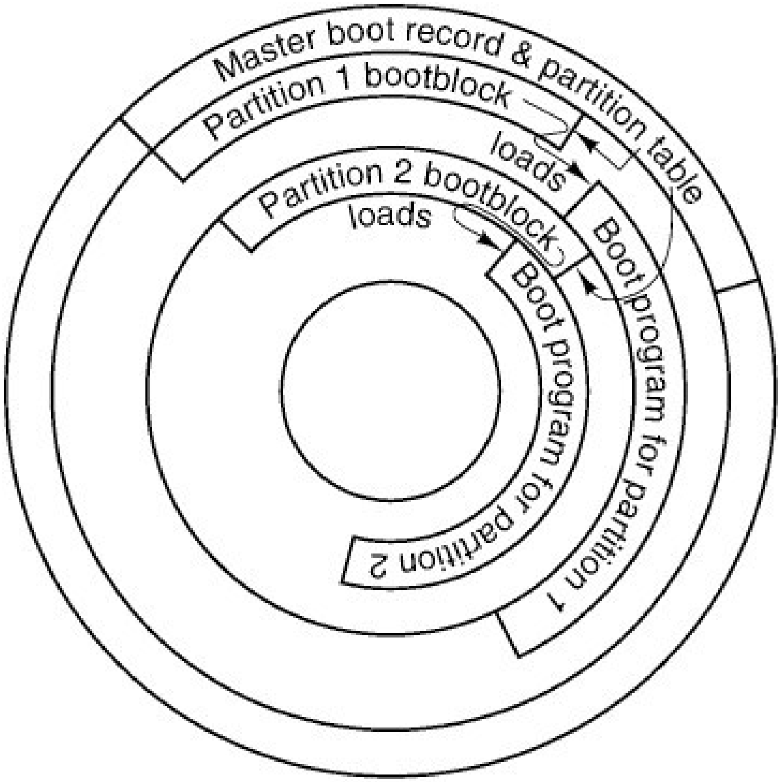


Figure 2-36. Disk structures used for bootstrapping.
(a) Unpartitioned disk. The first sector is the bootblock.

Bootstrapping MINIX (2)



(b)

Figure 2-36. Disk structures used for bootstrapping. (b) Partitioned disk. The first sector is the master boot record, also called masterboot.

Boot Time in MINIX

```
rootdev=256
ramimagedev=916
ramsize=4096
processor=586
bus=at
video=vga
chrome=color
memory=800:92880,100000:2F00000
c0=at
image=/minix/2.0.3r5
```

Figure 2-37. Boot parameters passed to the kernel at boot time in a typical MINIX 3 system.

System Initialization in MINIX

```
#include <minix/config.h>
#if _WORD_SIZE == 2
#include "mpx88.s"
#else
#include "mpx386.s"
#endif
```

Figure 2-38. How alternative assembly language source files are selected.

Interrupt Handling in MINIX (1)

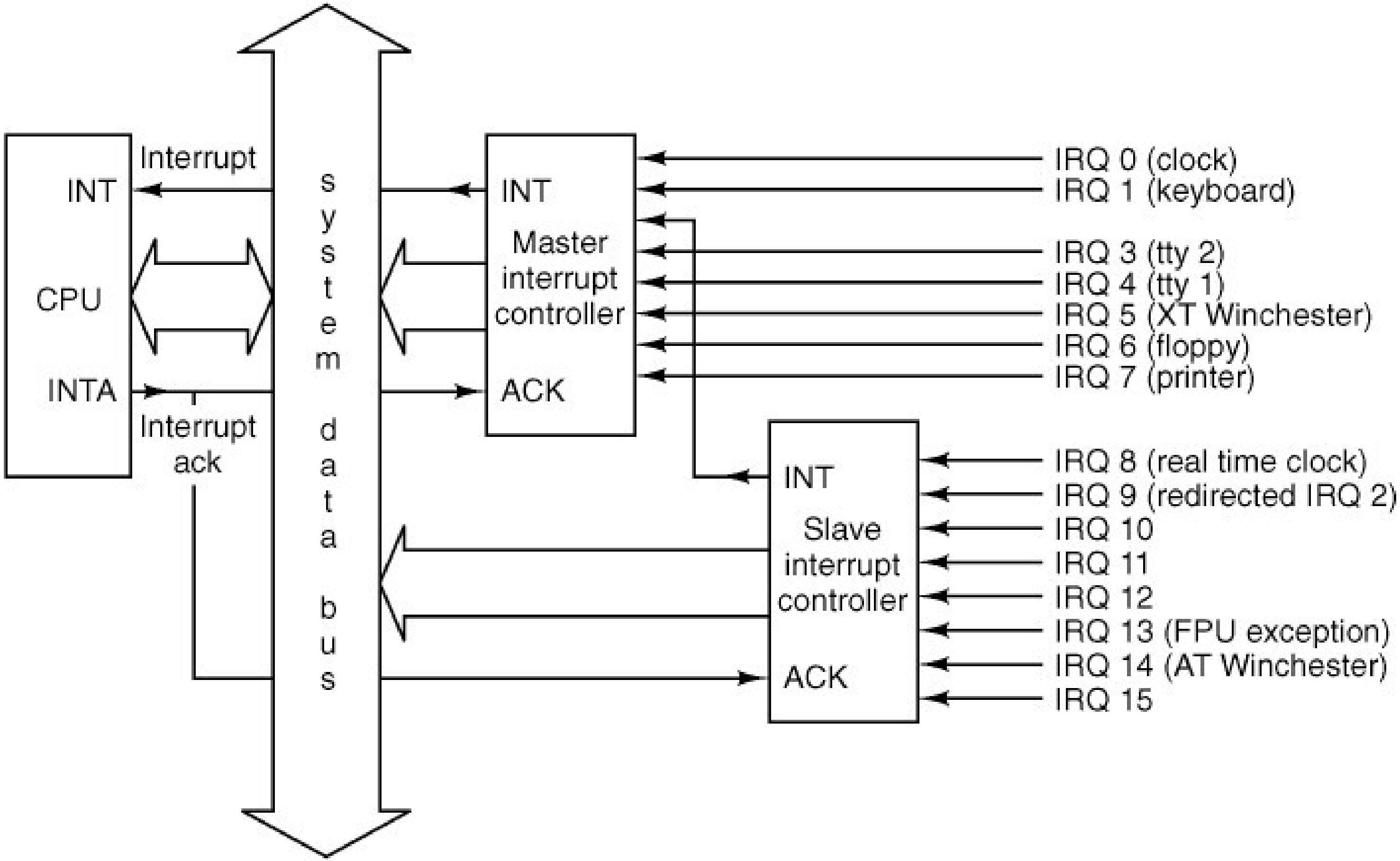


Figure 2-39. Interrupt processing hardware on a 32-bit Intel PC.

Interrupt Handling in MINIX (2)

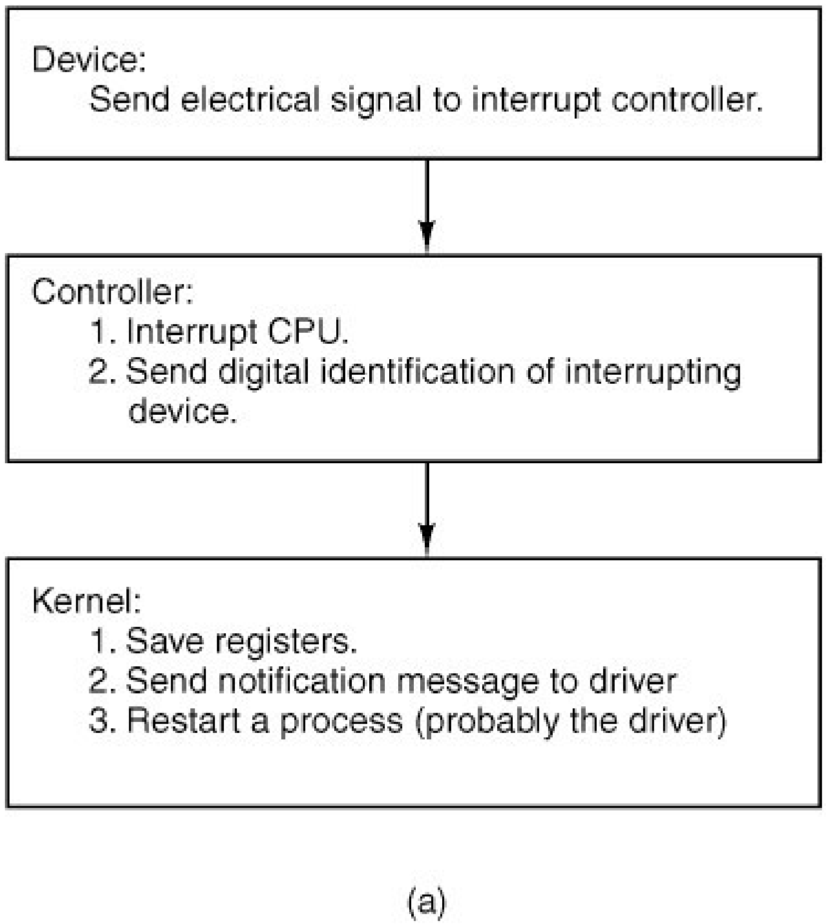
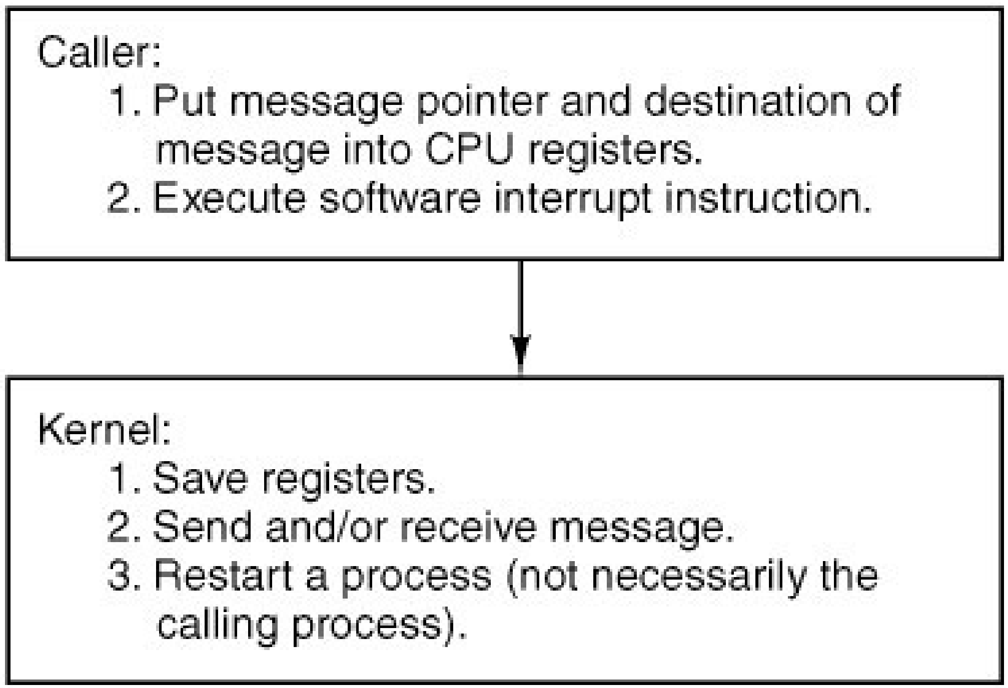


Figure 2-40. (a) How a hardware interrupt is processed.

Interrupt Handling in MINIX (3)



(b)

Figure 2-40. (b) How a system call is made.

Restart

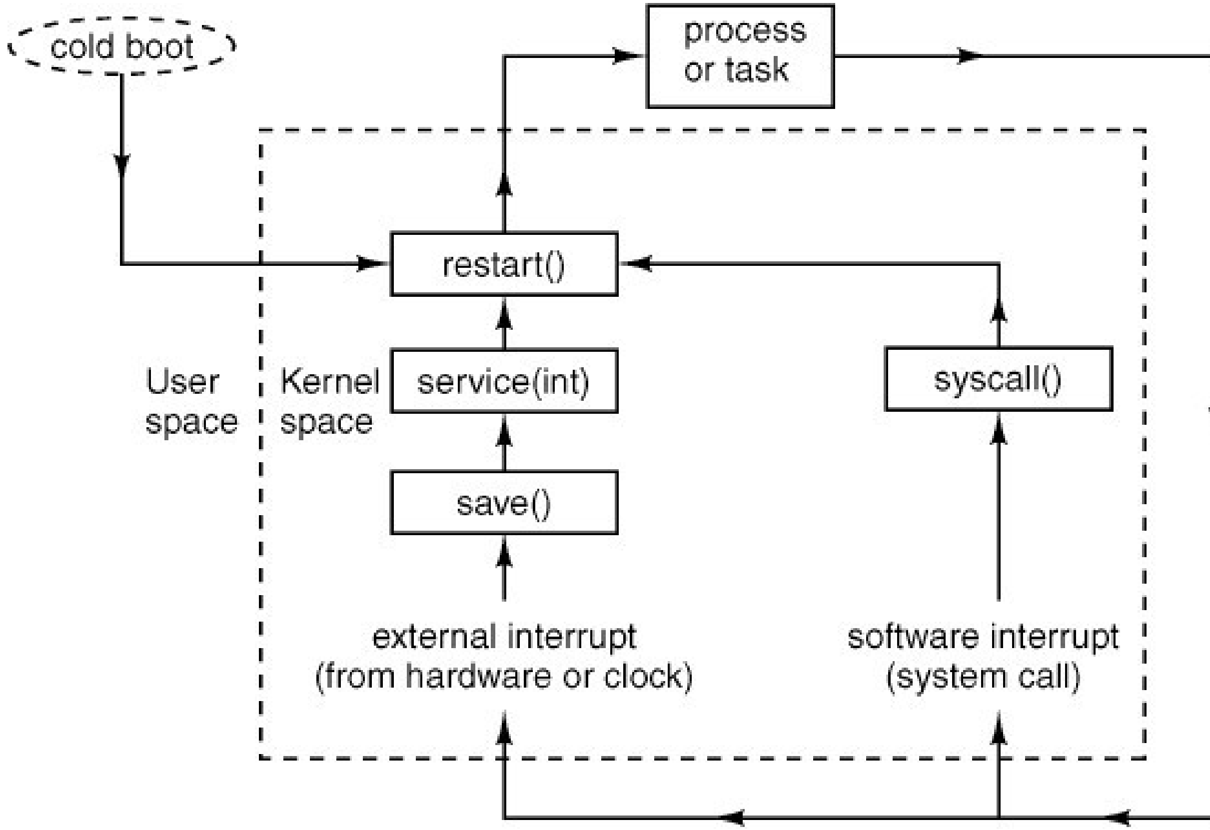


Figure 2-41. Restart is the common point reached after system startup, interrupts, or system calls. The most deserving process (which may be and often is a different process from the last one interrupted) runs next. Not shown in this diagram are interrupts that occur while the kernel itself is running.

Queueing

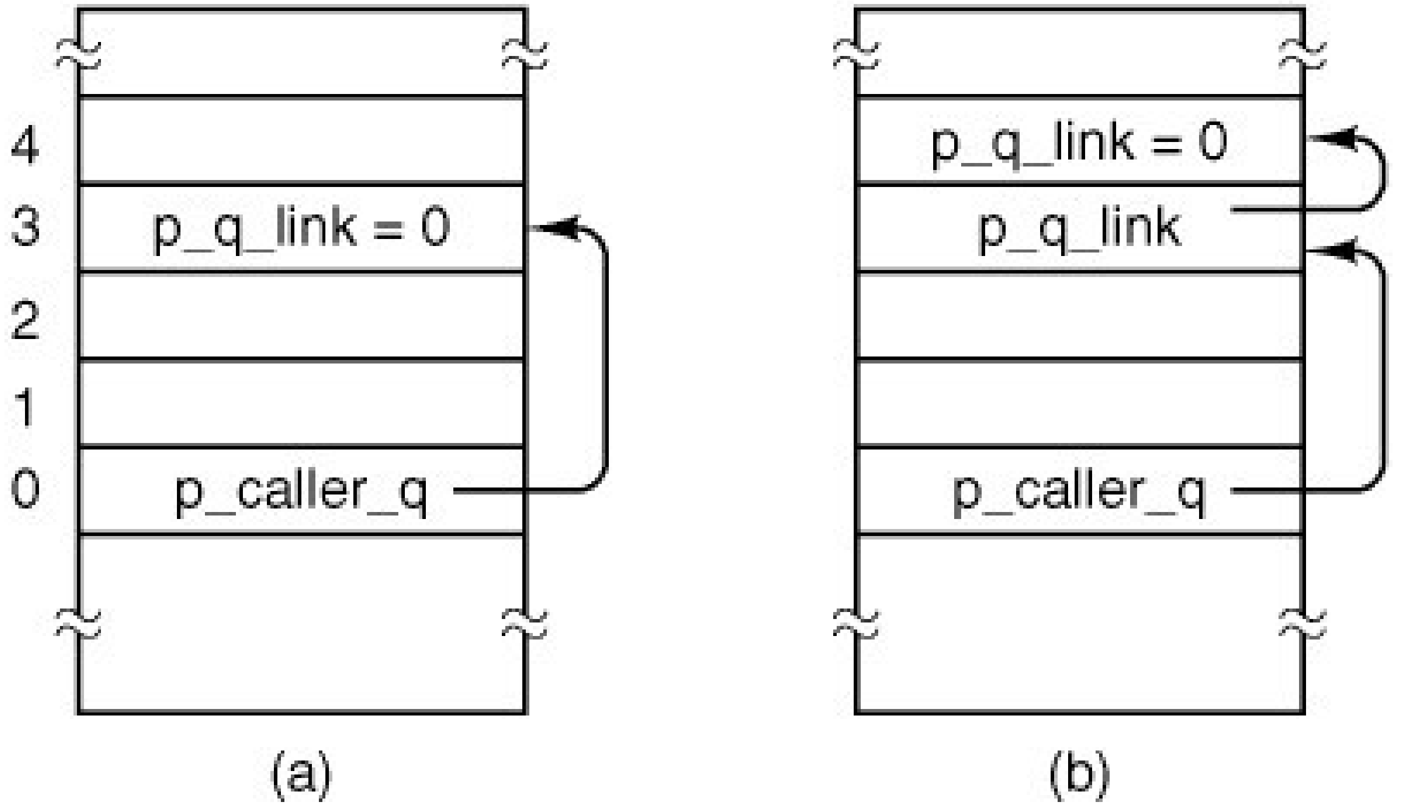


Figure 2-42. Queueing of processes trying to send to process 0.

Scheduling in MINIX

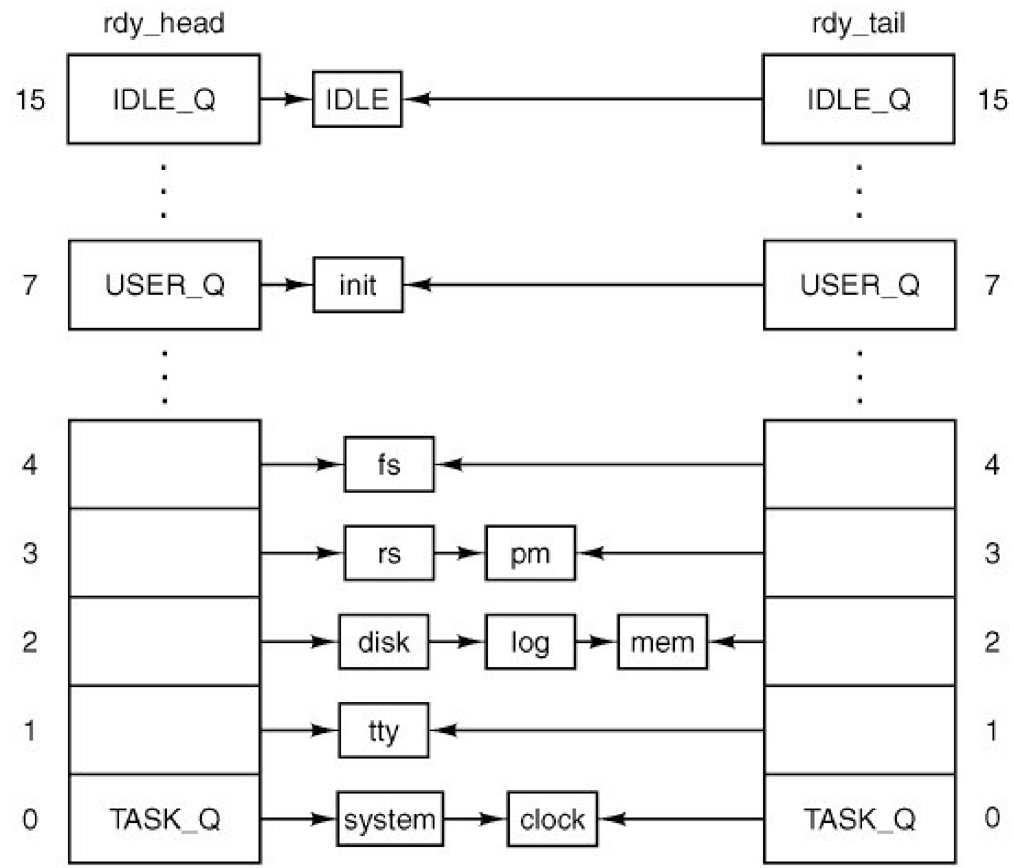


Figure 2-43. The scheduler maintains sixteen queues, one per priority level. Shown here is the initial queuing process as MINIX 3 starts up.

Hardware-Dependent Kernel Support

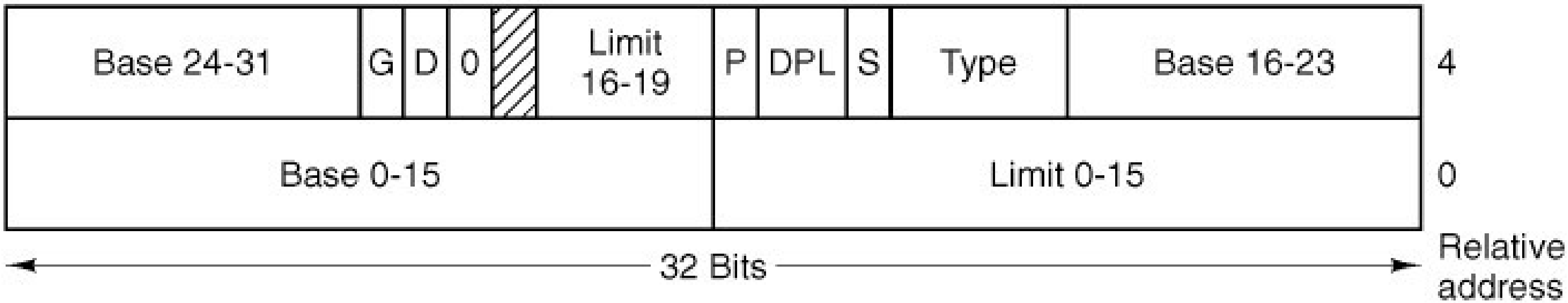


Figure 2-44. The format of an Intel segment descriptor.

Overview of System Task (1)

Message type	From	Meaning
sys_fork	PM	A process has forked
sys_exec	PM	Set stack pointer after EXEC call
sys_exit	PM	A process has exited
sys_nice	PM	Set scheduling priority
sys_privctl	RS	Set or change privileges
sys_trace	PM	Carry out an operation of the PTRACE call
sys_kill	PM,FS, TTY	Send signal to a process after KILL call
sys_getksig	PM	PM is checking for pending signals
sys_endksig	PM	PM has finished processing signal
sys_sigsend	PM	Send a signal to a process
sys_sigreturn	PM	Cleanup after completion of a signal
sys_irqctl	Drivers	Enable, disable, or configure interrupt

Figure 2-45. The message types accepted by the system task. "Any" means any system process; user processes cannot call the system task directly

Overview of System Task (2)

Message type	From	Meaning
sys_devio	Drivers	Read from or write to an I/O port
sys_sdevio	Drivers	Read or write string from/to I/O port
sys_vdevio	Drivers	Carry out a vector of I/O requests
sys_int86	Drivers	Do a real-mode BIOS call
sys_newmap	PM	Set up a process memory map
sys_segctl	Drivers	Add segment and get selector (far data access)
sys_memset	PM	Write char to memory area
sys_umap	Drivers	Convert virtual address to physical address
sys_vircopy	FS, Drivers	Copy using pure virtual addressing
sys_physcopy	Drivers	Copy using physical addressing
sys_virvcopy	Any	Vector of VCOPY requests
sys_physvcopy	Any	Vector of PHYSCOPY requests
sys_times	PM	Get uptime and process times
sys_setalarm	PM, FS, Drivers	Schedule a synchronous alarm
sys_abort	PM, TTY	Panic: MINIX is unable to continue
sys_getinfo	Any	Request system information

Figure 2-45. The message types accepted by the system task. “Any” means any system process; user processes cannot call the system task directly

Clock Hardware

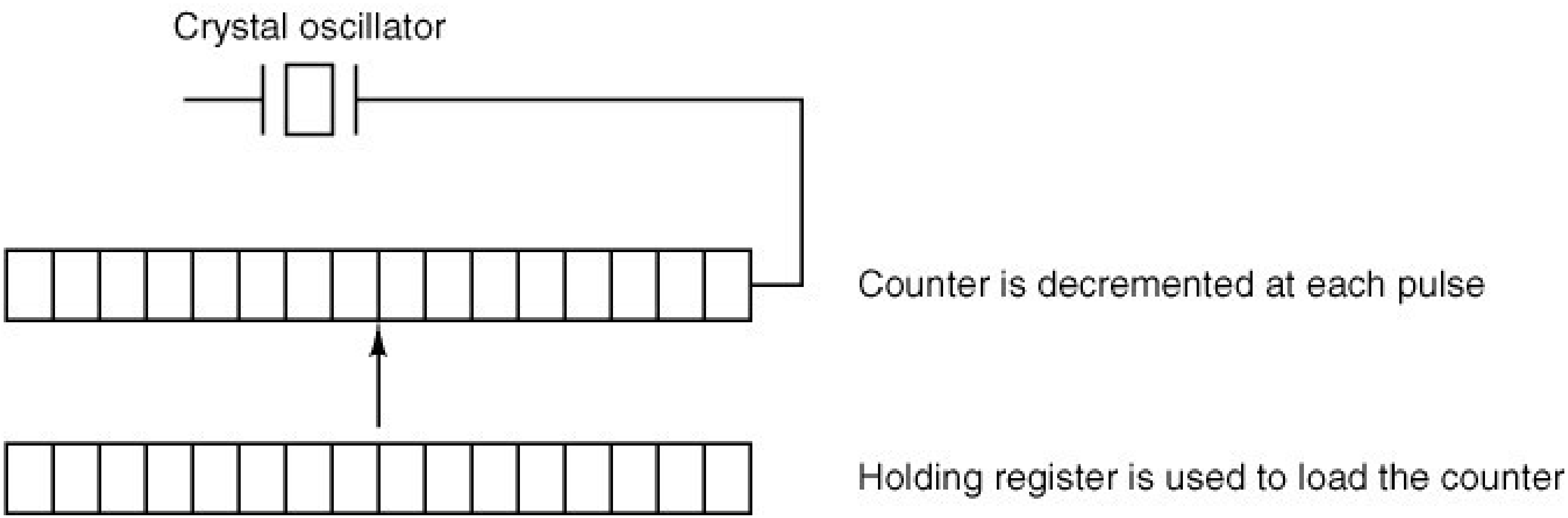


Figure 4-47. A programmable clock.

Clock Software (1)

Typical duties of a clock driver.

1. Maintain time of day
2. Prevent processes from running longer than allowed
3. Accounting for CPU usage
4. Handling alarm system call by user processes
5. Providing watchdog timers for parts of system itself
6. Doing profiling, monitoring, and statistics gathering

Clock Software (2)

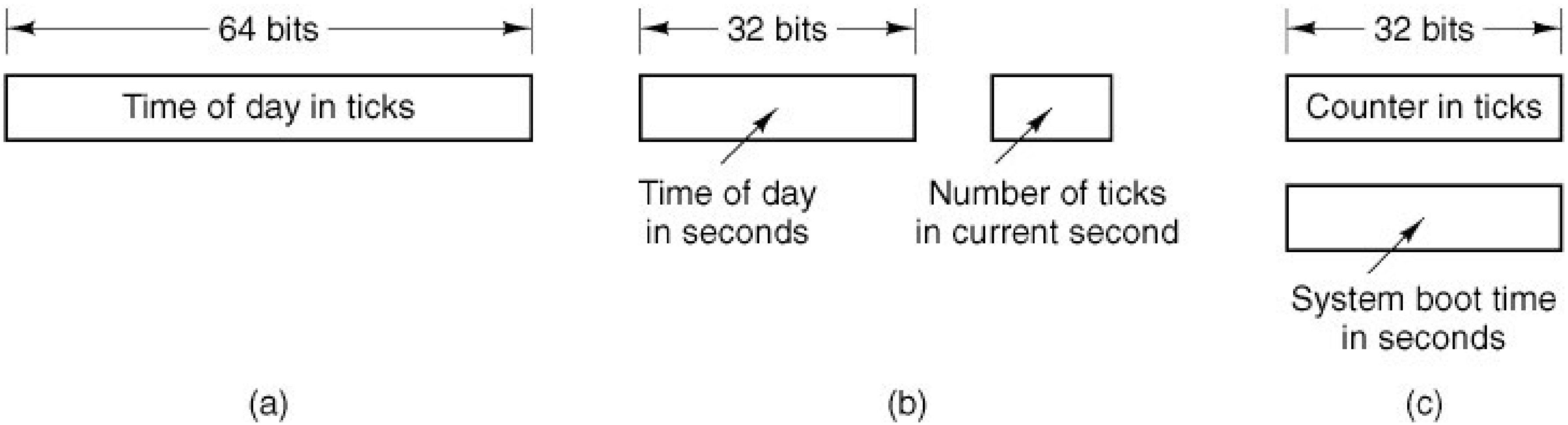


Figure 2-48. Three ways to maintain the time of day.

Clock Software (3)

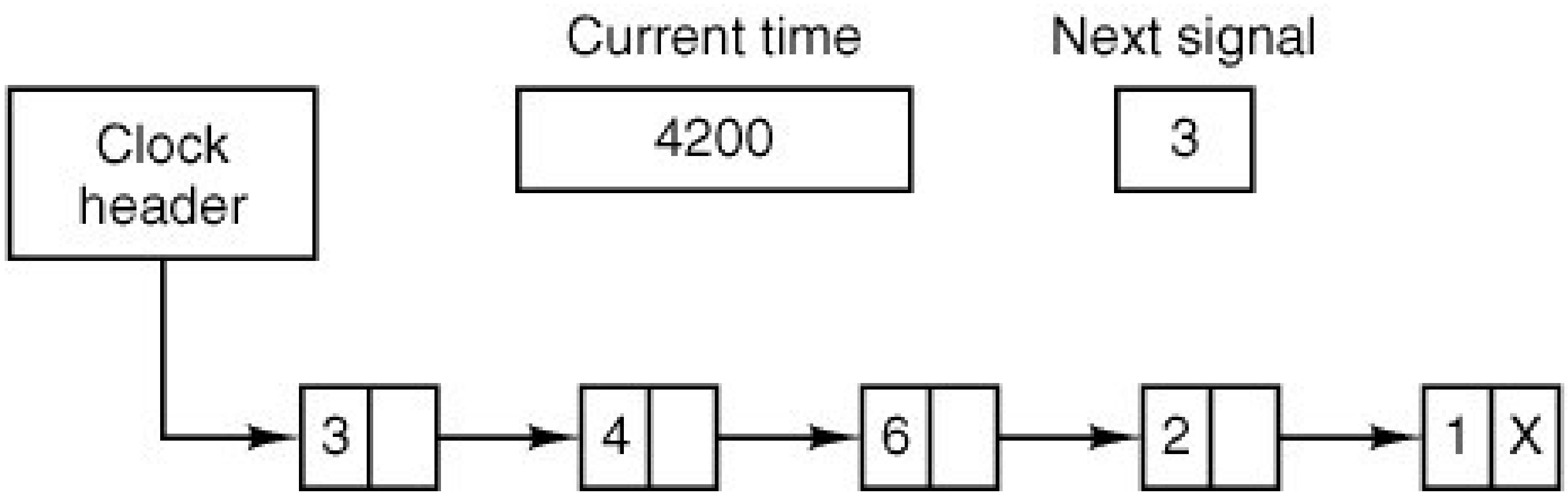


Figure 2-49. Simulating multiple timers with a single clock.

Summary of Clock Services

Service	Access	Response	Clients
get_uptime	Function call	Ticks	Kernel or system task
set_timer	Function call	None	Kernel or system task
reset_timer	Function call	None	Kernel or system task
read_clock	Function call	Count	Kernel or system task
clock_stop	Function call	None	Kernel or system task
Synchronous alarm	System call	Notification	Server or driver, via system task
POSIX alarm	System call	Signal	User process, via PM
Time	System call	Message	Any process, via PM

Figure 2-50. The time-related services supported by the clock driver.