INTRODUCTION TO MINIX3

Module 1.8

COP4600 – Operating Systems

Richard Newman

OUTLINE

- Structure of Minix3
  - 4 layers
  - Privileged instructions
- Memory layout
- Bootstrapping and startup
- Programming notes
- Message passing
  - Message types
  - Message queuing
  - Restrictions
- System calls

INTERNAL STRUCTURE OF MINIX

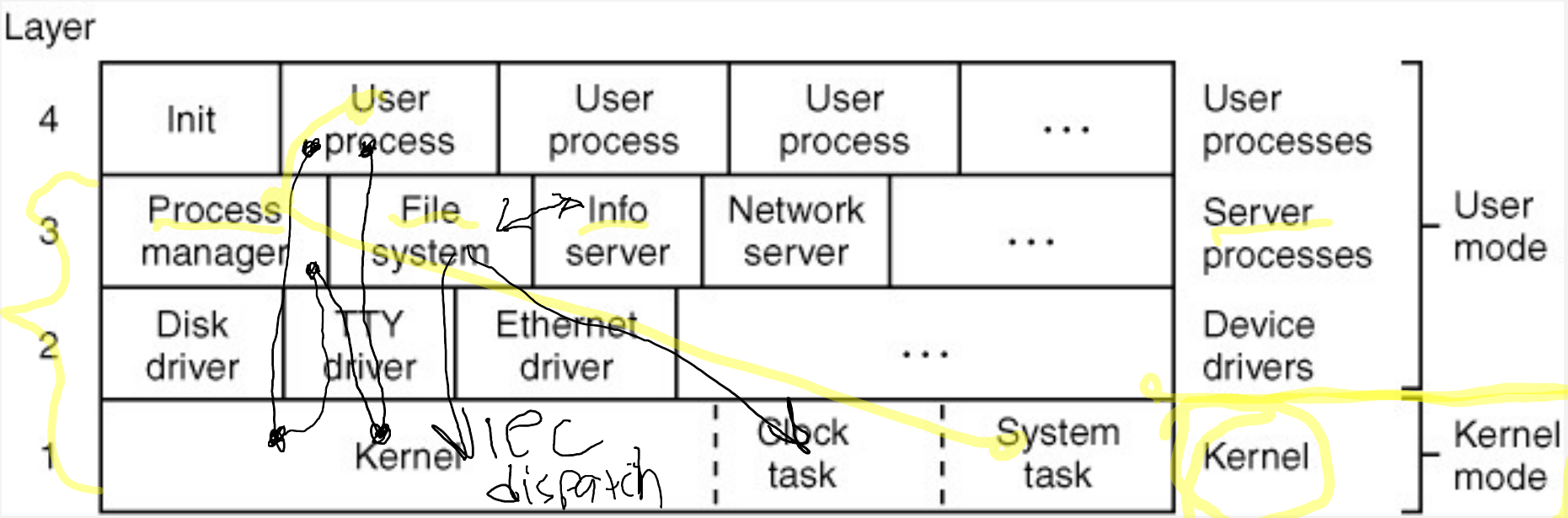


Figure 2-29. MINIX 3 is structured in four layers. Only processes in the bottom layer may use privileged (kernel mode) instructions.

WHO DOES WHAT

- Kernel
  - Message passing
  - Dispatcher on TRAP
  - System task – low level I/O
  - System task – low level memory tasks
  - Clock task – clock INTs
  - No msgs from user processes
- Device Drivers
  - Hardware-dependent code
  - Disk(s)
  - Terminal(s)
  - NIC
  - Only msgs from servers
- Services
  - PM
  - FS
  - RS
  - IS
  - Msg interface to user procs

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

parent

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)

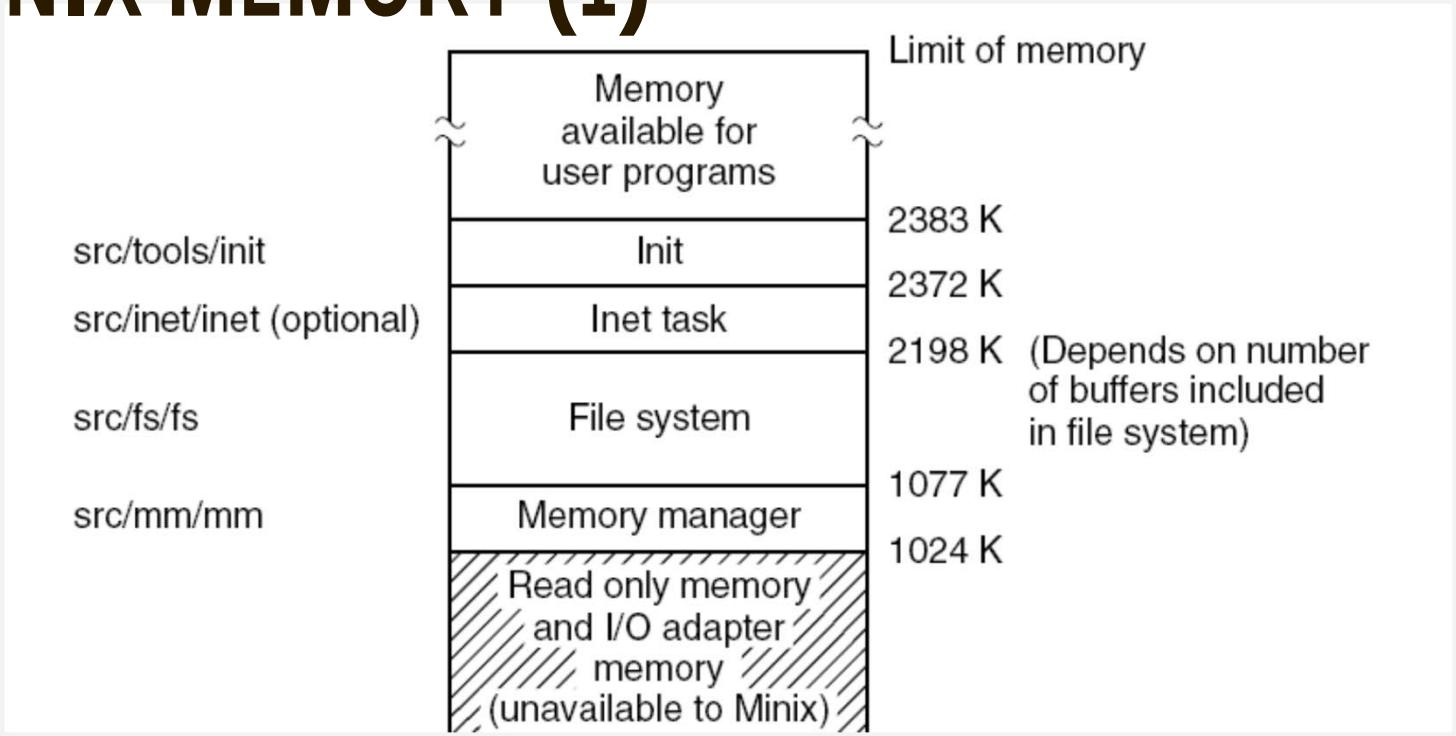


Figure 2-31. Memory layout after MINIX 3 has been loaded from the disk into memory. The kernel, servers, and drivers are independently compiled and linked programs, listed on the left. Sizes are not to scale

MINIX MEMORY (2)

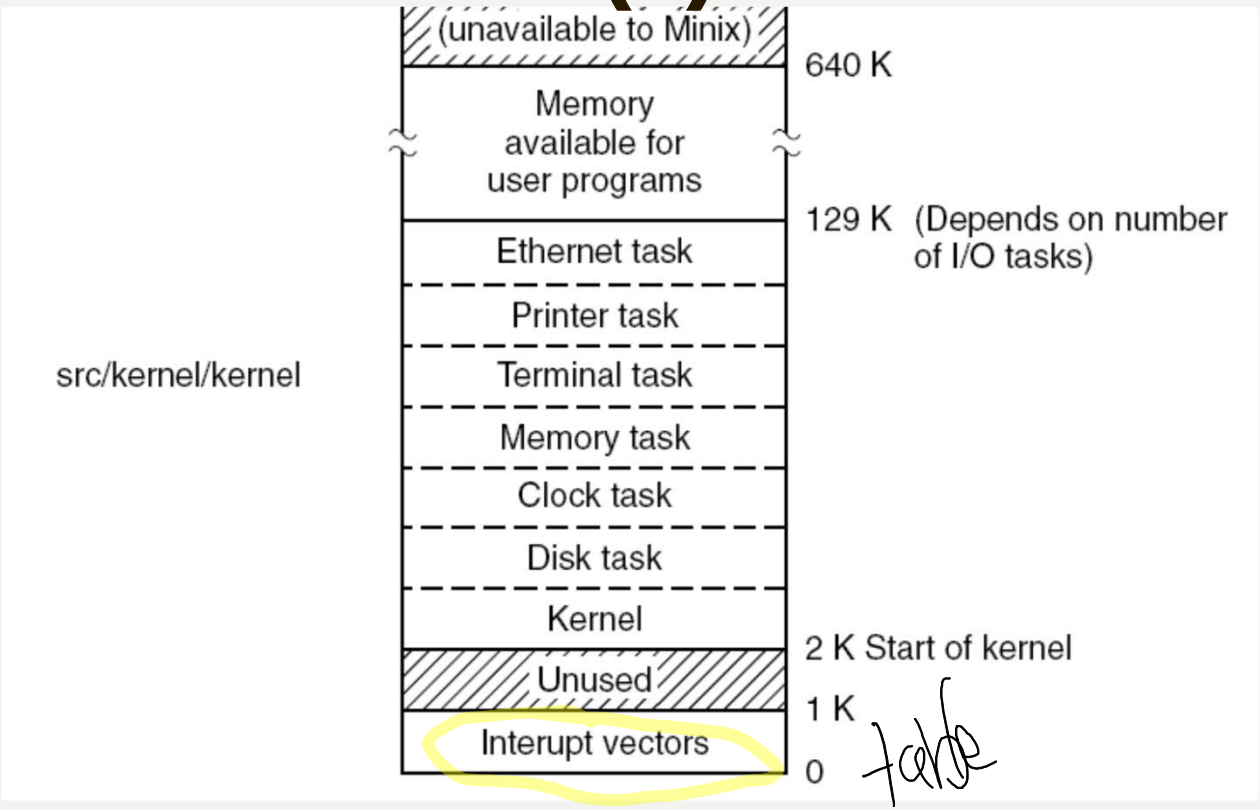


Figure 2-31. Memory layout after MINIX 3 has been loaded from the disk into memory.

BOOTSTRAPPING MINIX

Boot from CD-ROM or DVD-ROM copies into RAM disk, then boots from RAM disk image

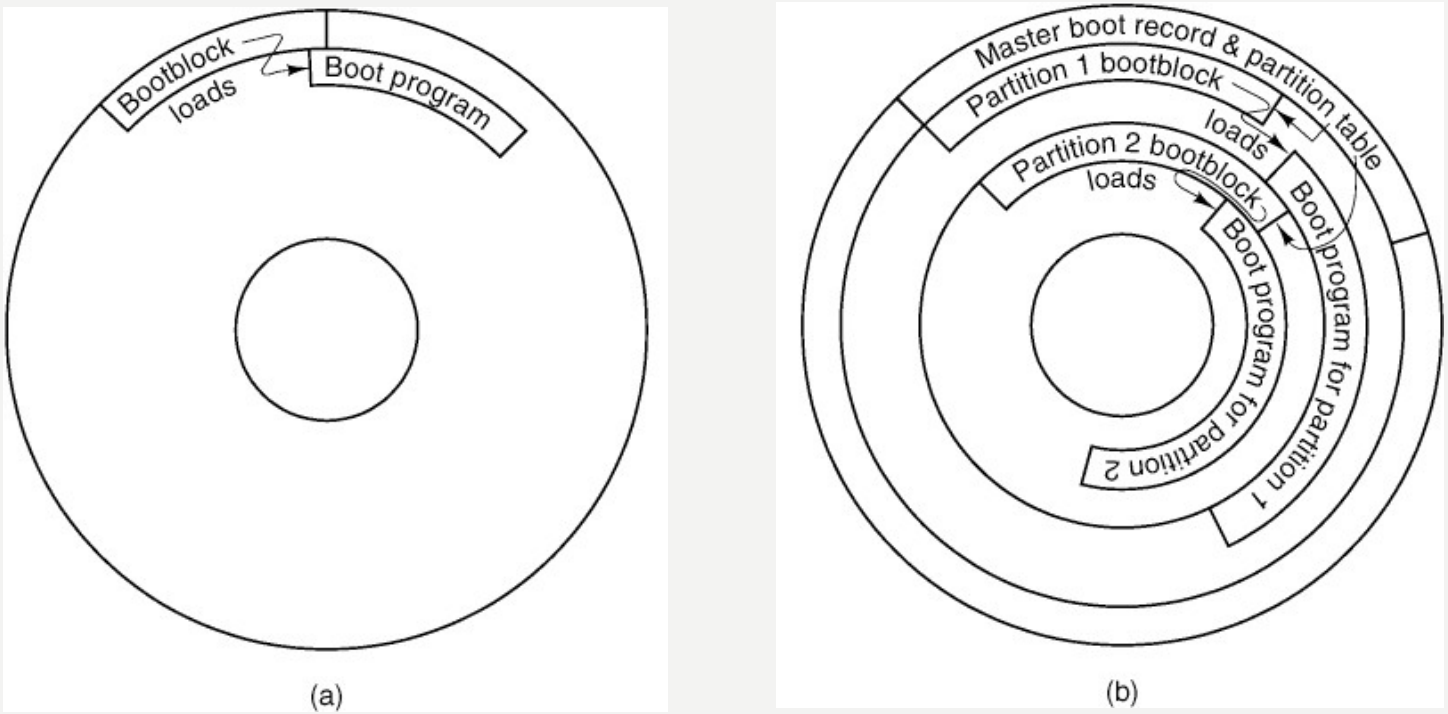


Figure 2-36. Disk structures used for bootstrapping. (a) Unpartitioned disk. The first sector is the bootblock. (b) Partitioned disk. The first sector is the master boot record, also called masterboot.

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.

CODE BASE NOTES

- Full path is /usr/src/…
  - So src/include is really /usr/src/include/
- May make /src/ a copy on root directory if root is a RAM disk
- Makefile in every directory
  - Make facility manages efficient compilation and linking of file in its directory subtree
  - More on make in exercises

changes

- /usr/src/include/ - master include directory – overwrites /usr/include
- /usr/include/ - standard search path for include files
  - E.g., #include <filename>
- Local include files
  - E.g., #include "filename"
  - Same filename may be reused
- include/ subdirectories:
  - sys/ - more POSIX headers
  - minix/ - Minix3 headers
  - ibm/ - IBM PC-specific headers
  - arpa/, net/ - network extensions

CODE BASE NOTES

- Full path is /usr/src/…
  - So src/include is really /usr/src/include/
  - kernel/ is really /usr/src/kernel/
- kernel/ - layer 1 source code
  - Scheduling, messages
  - Clock task
  - System task
- drivers/ - layer 2 source code
  - Device drivers
- servers/ - layer 3 source code
  - Process Manager
  - File System
- src/lib/ - API
  - library procedures
  - E.g., open, read, fork, exec, etc.
- src/tools/ -
  - makefile and scripts for building Minix3 system
- src/boot/ -
  - Code for booting and installing Minix3 system
- src/test/ -
  - Programs to test newly compiled Minix3 system
- src/commands/ -
  - Utility programs (> 200 of them!)
  - E.g., cat, cp, date, ls, pwd, etc.

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"
```

<const.h> in standard location

"const.h" in local directory

not here

ACHTUNG! /usr/include/ is deleted on recompile – copied from /usr/src/include

changes here

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.

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.

SIZES OF TYPES IN MINIX

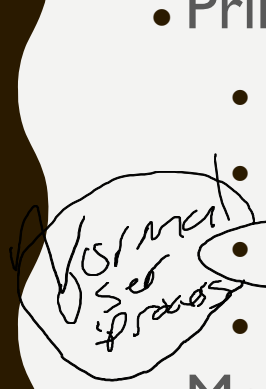
Type	16-Bit MINIX	32-Bit MINIX
gid_t	8	8
dev_t	16	16
pid_t	16	32
ino_t	16	32

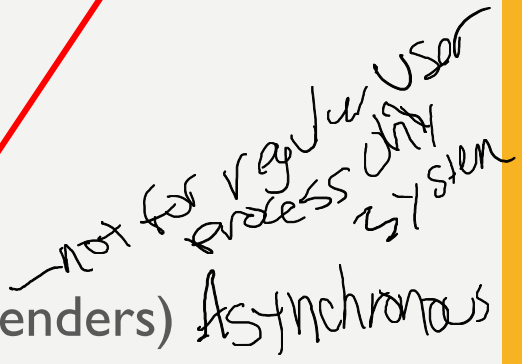
Figure 2-33. The size, in bits, of some types on 16-bit and 32-bit systems.

INTERPROCESS COMMUNICATION

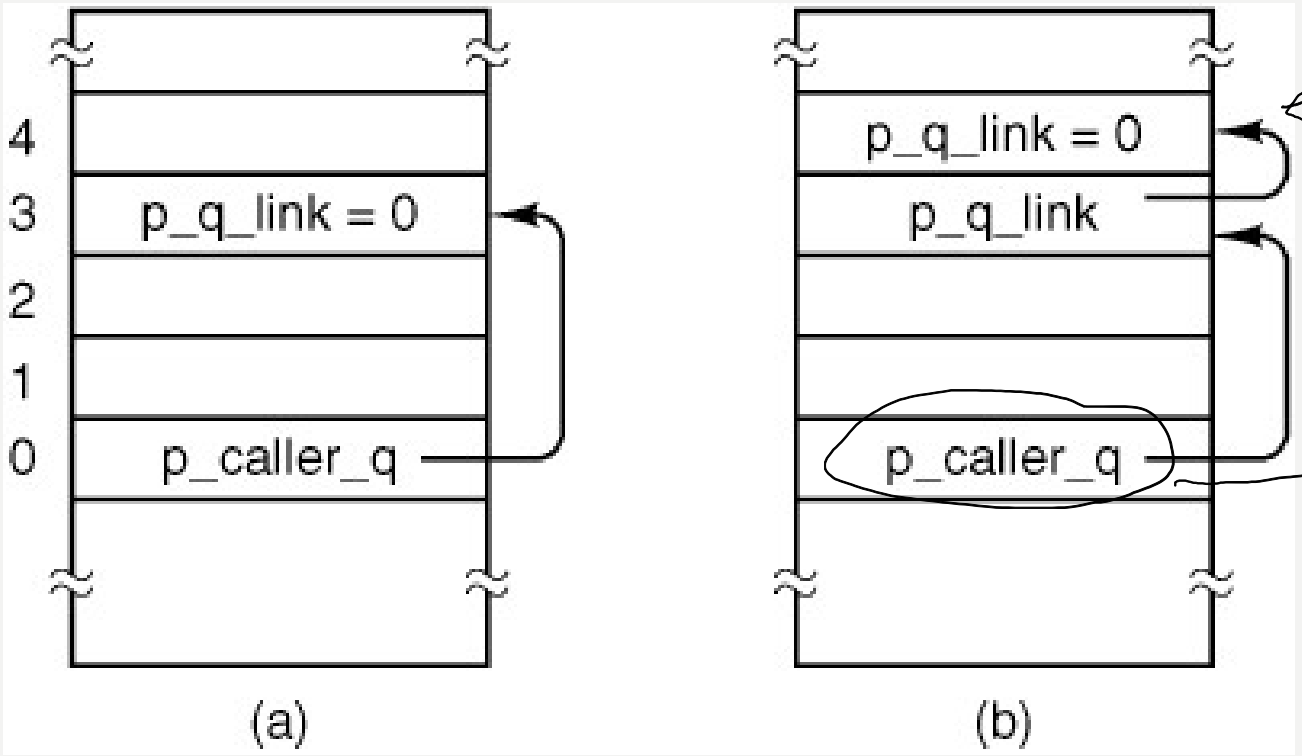
- Everything in Minix3 is done through message-passing
- Message mechanism in Minix 3 is rendezvous
  - Blocking receive
  - Blocking send
  - Allows unreceived message to be stored by sender
- Primitives
  - send(dest, &msg) – msg is local buffer for message
  - receive(source, &msg) – source can be ANY
  - sendrec(src_dst, &mgs) – reply overwrites msg
  - notify(dest) – non-blocking alert (uses bitmap for senders)
- Messages are small, fixed size
  - Can keep in process table with process info

Only primitive available to normal user processes





MESSAGE QUEUING

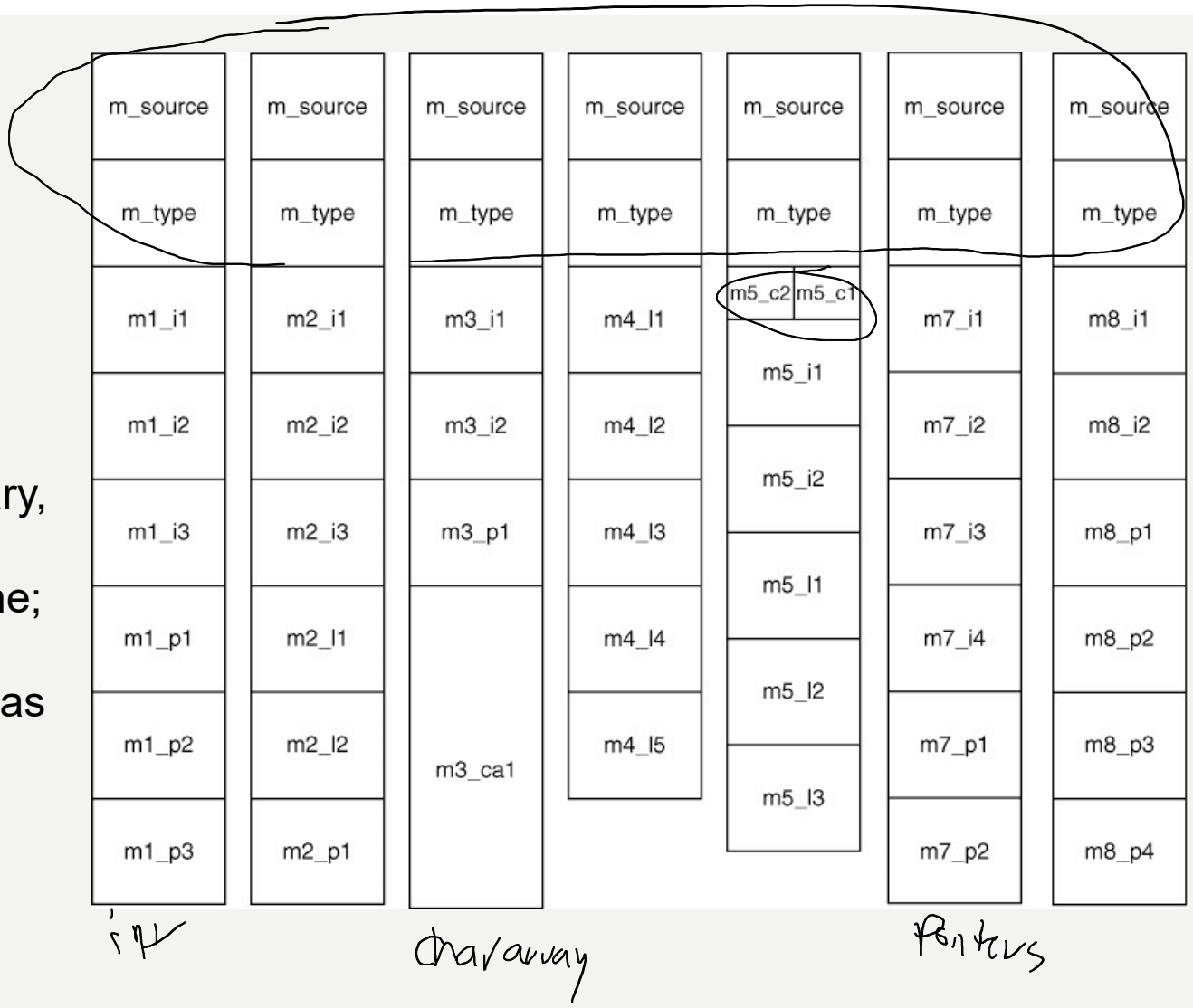


Process 3 tries to send to process 0, must wait. Process 4 also must wait.

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

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; these are sizes on CPUs with 32-bit pointers, such as those of Pentium family members.



MESSAGE QUEUING

--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 bitmap is truncated to 16 bits.

MESSAGE QUEUING

ID 0 is for user procs

P flag –preemptable?

SRB traps – can proc use send/receive?

ipc_to mask – who can process send a message?

--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. Clock task, file server, tty, and init processes privileges are typical of tasks, servers, device drivers, and user processes, resp.

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. Message types accepted by the system task.

SYSTEM CALLS IN MINIX 3

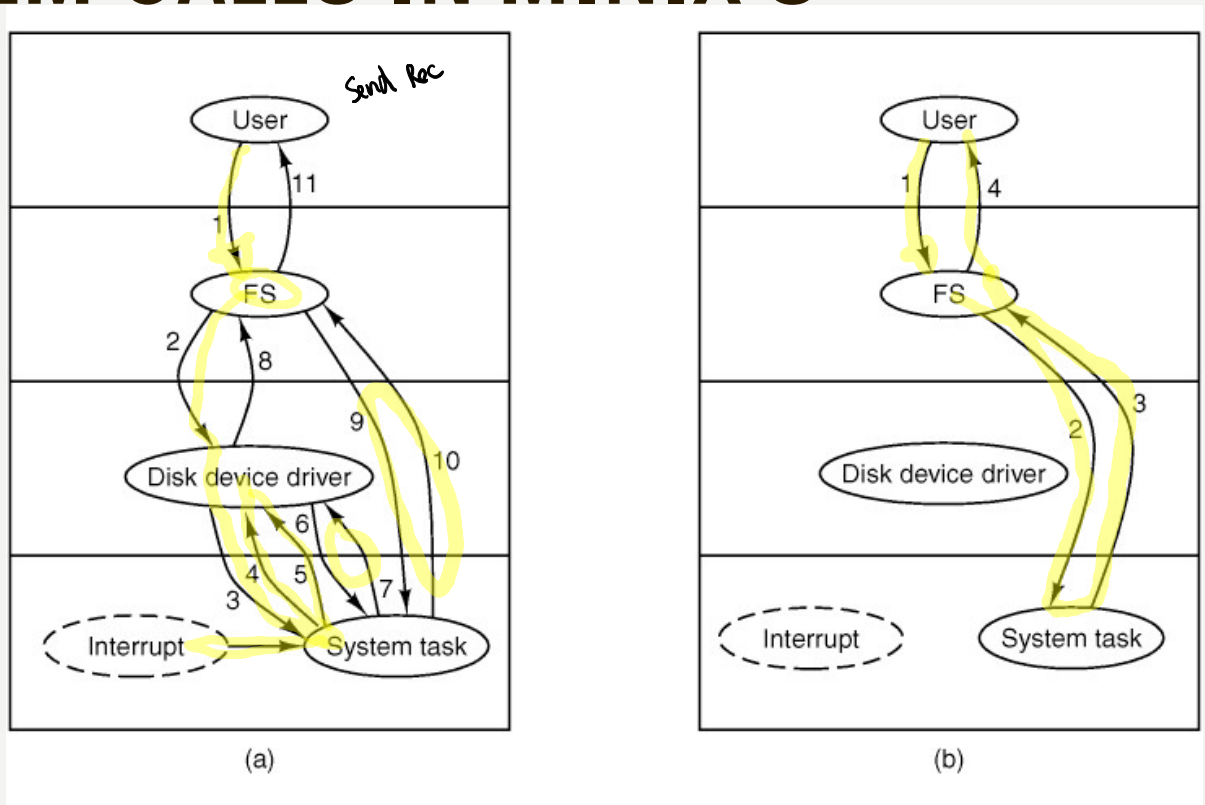


Figure 2-46. (a) Worst case for reading a block requires seven messages. (b) Best case for reading a block requires four messages.

SUMMARY

- Structure of Minix3
  - 4 layers
  - Privileged instructions
- Memory layout
- Bootstrapping and startup
- Programming notes
- Message passing
  - Message types
  - Message queuing
  - Restrictions
- System calls