file management

file management

- that part of the OS that manage data on disk structured as files and directories
  - usually part of the os kernel, might be a process in microkernel architecture
  - usually based on block-oriented devices (i.e. disks), but also on file (e.g., in virtual machines)
- properties
  - structure
  - persistent (and “reliable” to crashes etc.)
  - shared among several processes

file operations

- create a new file
- open an existing file
  - needs specification of what operations are requested
- delete
- read (usually blocking)
- write (usually non-blocking)
- close
  - flushes buffers

directory operations

- create
- delete
- change
  - create a file or directory in it
  - delete a file or directory in it
  - move a file or directory from somewhere else into the directory
  - move a file or directory for the directory to somewhere else

objectives

- reliability: guarantee that the data in the file are always valid
  - upon regular operation
  - upon crashes
  - upon concurrent access
  - can be thought a huge data structure with consistency invariants
- optimize performance
- independency on storage device types
- security support for multiuser systems


not on
the book

“filesystem”

- an overloaded word
  - the rules of a filesystem (i.e. the rules describing the data structures stored on disk)
    - specification of the FAT filesystem
  - an instance of the filesystem
    - mount a filesystem
    - mount a partition containing a filesystem
  - the part of the kernel implementing the filesystem
    - fix a bug in the filesystem
- actual meaning should be clear from the context


not on the book

file systems need data structures...

- to keep track of which blocks are allocated to a given file
- to keep track of free blocks
- to keep names, directories, access rights, etc

- consistency and check
  - periodic, or after a crash
  - data can be lost but consistency is preserved
- recovery (journaled filesystems)

journaling

- when a change is requested perform it according to the following approach
  - write a journal of what have to be done
  - periodically
    - stop receiving further requests
    - perform the action in the journal
    - when all the actions are done empty the journal
- after a crash always perform the action in the journal
  - actions should be idempotent!
- improve reliability, degrade performance

UNIX File Management

- Types of files
  - Regular, or ordinary
    - a sequence of bytes (no records!)
  - Directory
  - Special (character of block devices)
  - Named pipes (FIFO)
  - Links (hard links)
  - Symbolic links (soft links)

Inodes

- Index node
- Control structure that contains key information for a particular file

File Mode	16-bit flag that stores access and execution permissions associated with the file. 12-14 File type (regular, directory, character or block special, FIFO pipe 9-11 Execution flags 8 Owner read permission 7 Owner write permission 6 Owner execute permission 5 Group read permission 4 Group write permission 3 Group execute permission 2 Other read permission 1 Other write permission 0 Other execute permission
Link Count	Number of directory references to this inode
Owner ID	Individual owner of file
Group ID	Group owner associated with this file
File Size	Number of bytes in file
File Addresses	39 bytes of address information
Last Accessed	Time of last file access
Last Modified	Time of last file modification
Inode Modified	Time of last inode modification

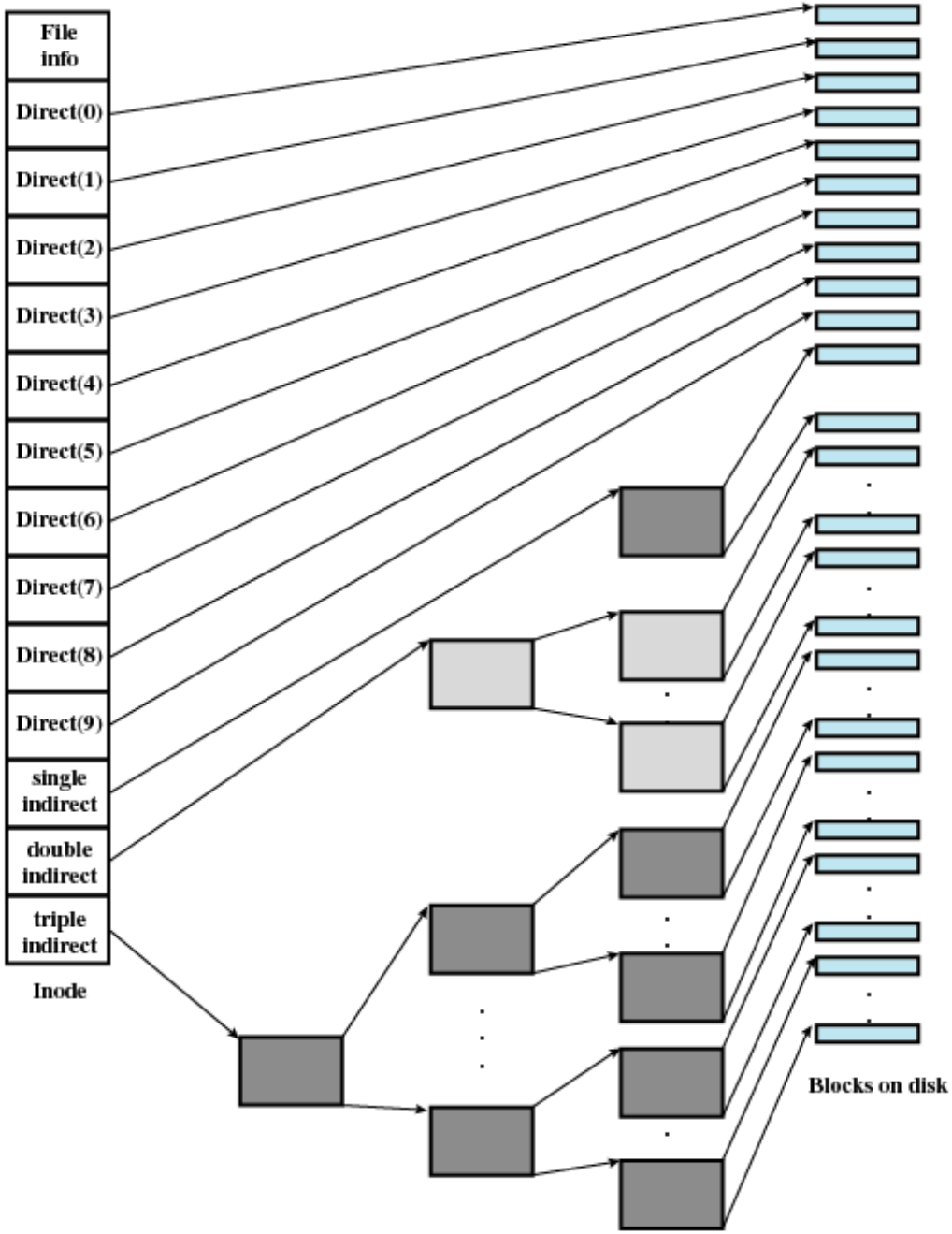


Figure 12.13 Layout of a UNIX File on Disk

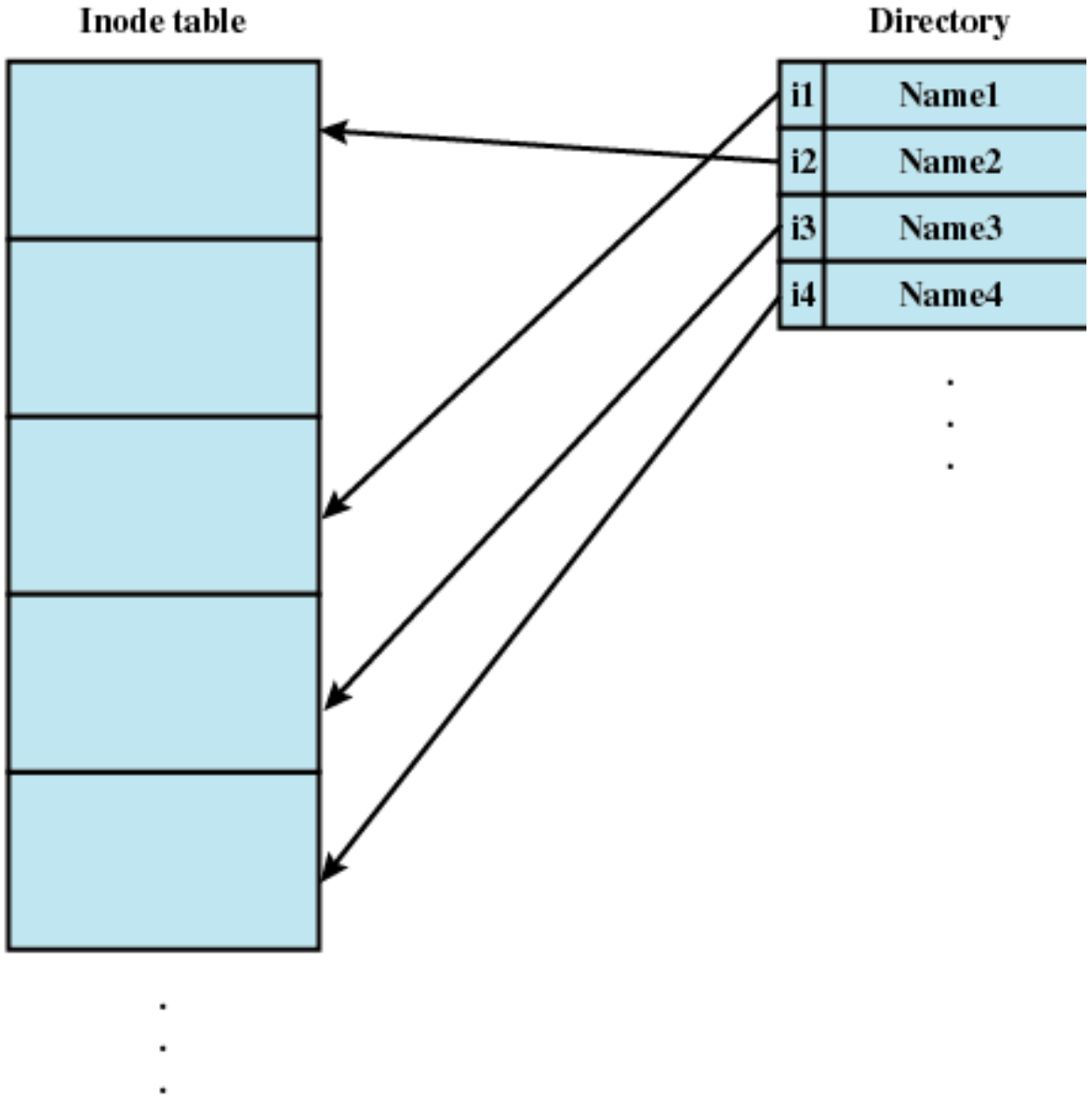


Figure 12.14 UNIX Directories and Inodes

not on
the book

hard and soft links

- hard links file
  - more filenames can point to the same inode
  - each name is said **hard link**
  - an hard link is always valid
  - an inode is deleted if and only if it has no names
    - reference counting
- soft links
  - a soft link is a file that contains a pathname
  - they can be relative or absolute
  - they can be invalid

Linux Virtual File System

- Uniform file system interface to user processes
- Represents any conceivable file system's general feature and behavior
- Assumes files are objects that share basic properties regardless of the target file system

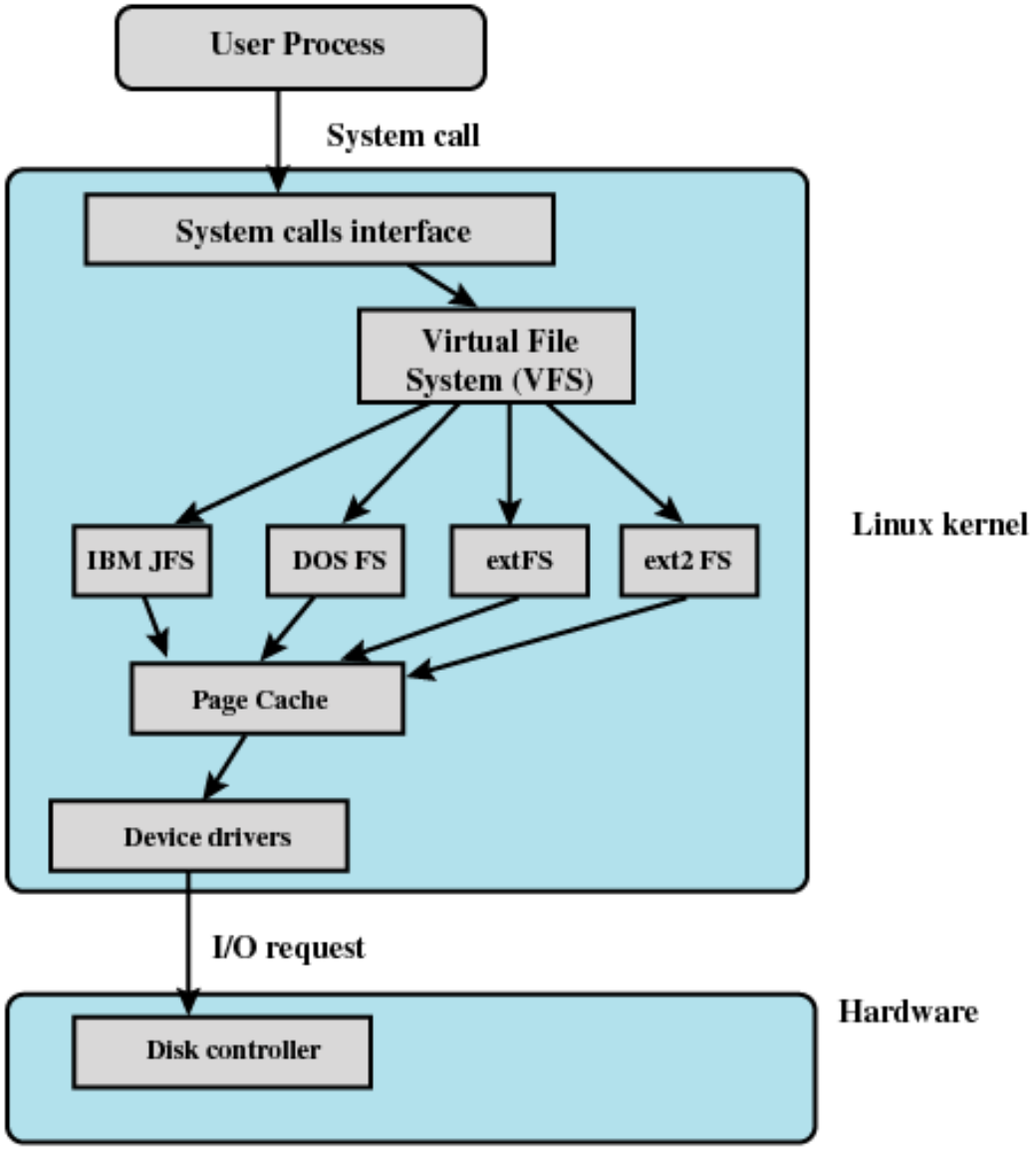


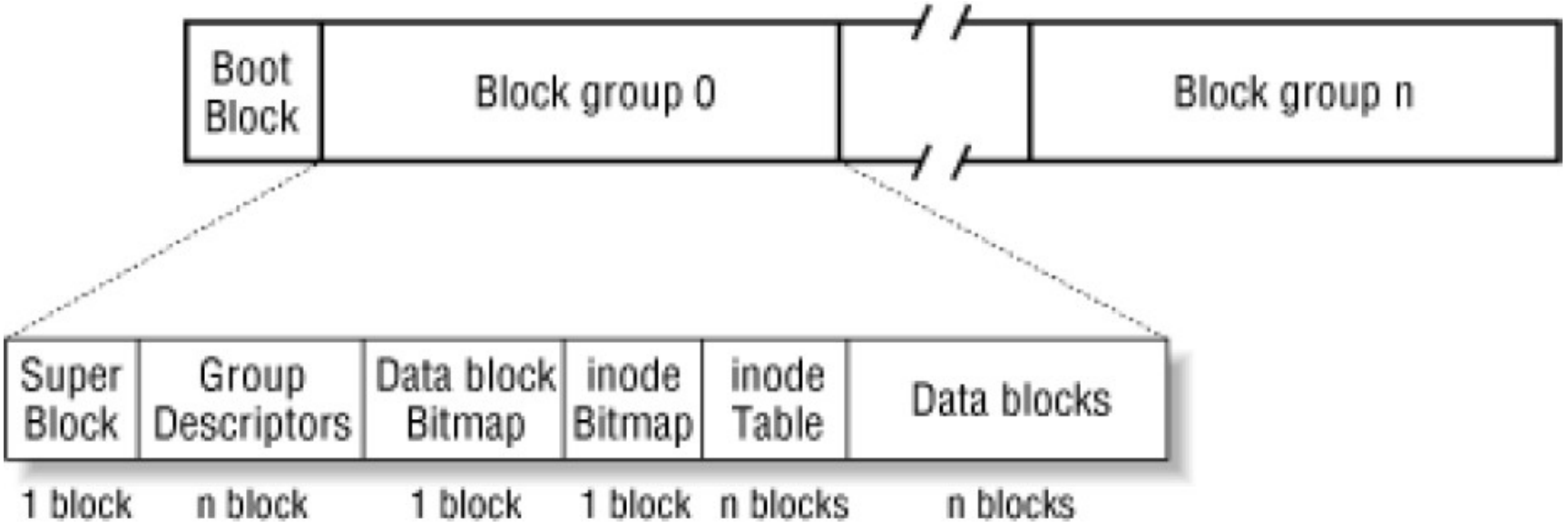
Figure 12.15 Linux Virtual File System Context

Primary Objects in VFS

- Superblock object
  - Represents a specific mounted file system
- Inode object
  - Represents a specific file
- Dentry object
  - Represents a specific directory entry
- File object
  - Represents an open file associated with a process

not on the book

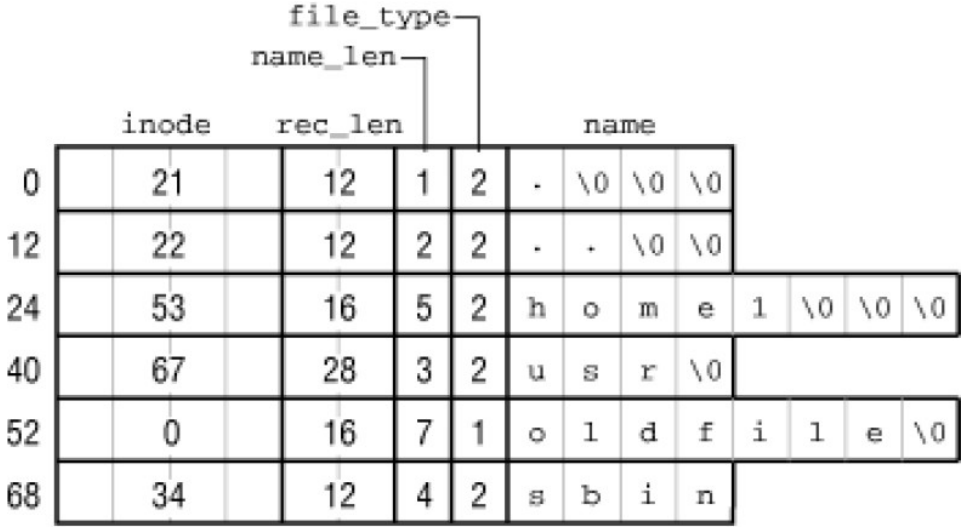
ext2 file system



- for each group
  - all groups are of the same length
  - each group says where all other groups are (redundancy)
  - 2 bitmaps for free/allocated spaces
    - both data blocks and inodes of this group

not on
the book

ext2 file system



- directory stored in variable record length format
- stored as a file
- given the inode *i*, where is it stored on the disk?

  group: *i* / inodes_in_one_group (/: the integer quotient)

  inode: i % inodes_in_one_group (%: is the reminder)

ext3

- ext2 + journal