CS3281 / CS5281

Filesystems

CS3281 / CS5281

Spring 2024

ISIS
Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

VANDERBILT UNIVERSITY

Overview

- A filesystem is an organized collection of files and directories
- The Linux kernel maintains a single hierarchical directory structure to organize all files in the system
  - Not like Windows where each drive (C, D, E, etc) has its own hierarchy
- Root directory is named /
  - Pronounced “slash”

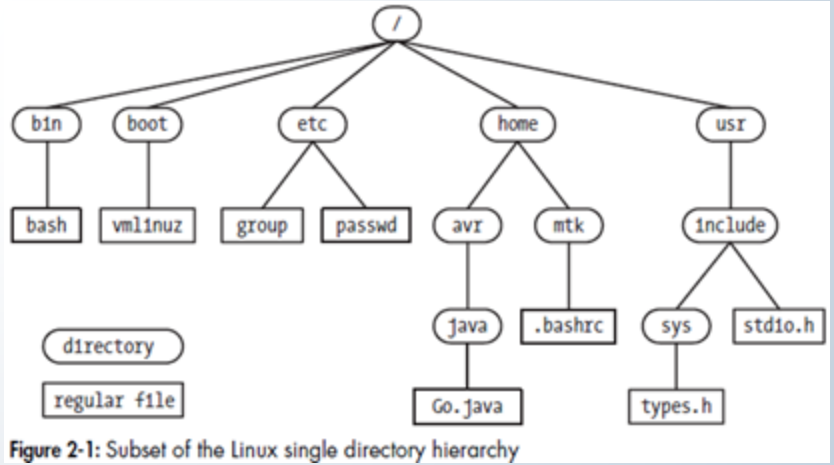


Figure 2-1: Subset of the Linux single directory hierarchy

*Figure from *The Linux Programming Interface* by Michael Kerrisk

File Types

- Every file has a type
  - This is the character in the first column when you do ls -l
- Regular files: ordinary data files, like text files, executables, libraries
- Special files: files other than ordinary data files
  - Devices: represents a device (virtual or physical)
    - Block device
    - Character device
  - Named pipes (also called fifos)
  - Directories
  - Symbolic links
- Example on right: block device files

```
daniel@ubuntu:/dev$ ls -l sda*
brw-rw---- 1 root disk 8, 0 Nov  5 09:21 sda
brw-rw---- 1 root disk 8, 1 Nov  5 09:21 sda1
daniel@ubuntu:/dev$
```

Filenames and Pathnames

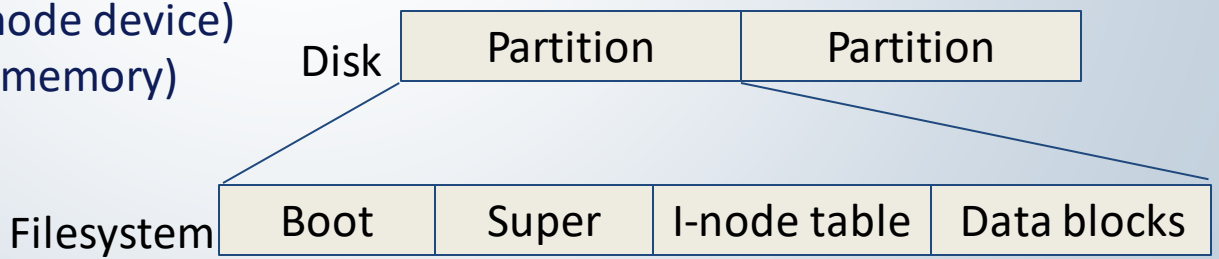
- Linux filesystems allow filenames:
  - Up to 255 characters
  - To contain any characters except slashes (/) and null characters (\0)
  - Recommended to only use letters, digits, . (period), _ (underscore), and - (hyphen)
- A pathname is a string with an optional / followed by a series of filenames separated by slashes
  - Absolute pathname: starts with a /
    - Examples: /home/student/ (student's home directory), /home/student/hw.txt (student's homework file)
  - Relative pathname: relative to the current directory
    - Examples: assignment-8/build (assignment-8 is a subdirectory of my current directory)

File Permissions

- Each file has a user ID and group ID that defines the owner and group
  - Ownership determines file access rights to users of the file
- Three categories of users:
  - Owner
  - Group: users who are members of the group
  - Other: everyone else
- There are three permission bits for each category: read, write, execute
  - Examples:
    - 111 110 000 (760): owner can read, write, and execute; group can read and write; everyone else cannot access the file
    - 110 000 000 (600): owner can read and write; everyone else cannot access
    - Look at the permission bits on ~/.ssh/id_rsa -- what is the reason for this?

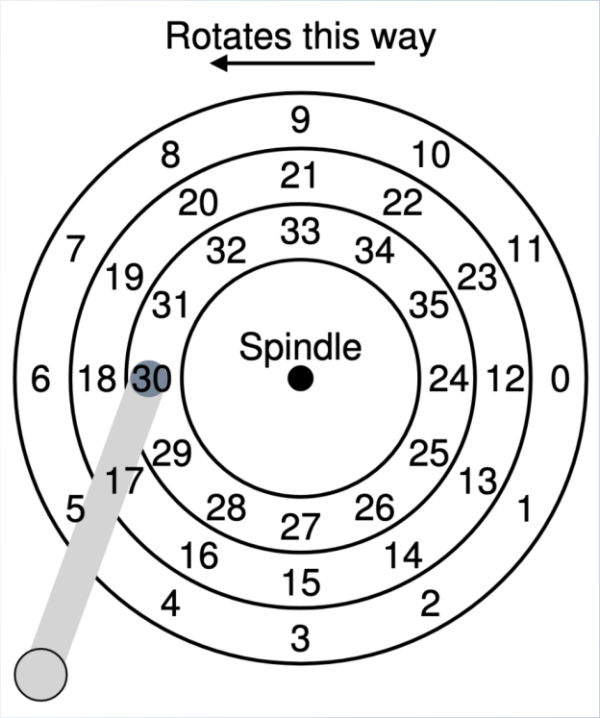
Back to Filesystems

- A disk drive is divided into circles called tracks
  - Tracks are divided into sectors
    - Sectors are a series of physical blocks
      - Physical block: the smallest unit a disk can read or write
        - Usually 512 bytes (older disks) or 4096 bytes (newer disks)
- Each disk is divided into partitions
  - Each is a separate device under /dev
  - A partition holds either
    - *Filesystem*
    - Data area (raw-mode device)
    - Swap (for virtual memory)



Spinning Disks

- Disks are composed of platters that store data
- Platters spin around a spindle
  - 7,200 RPM for consumer drives
  - 15,000 RPM for commercial drives
- Reading and writing is done by a disk head
- Head controlled by disk arm
- Disk geometry and access patterns affect performance



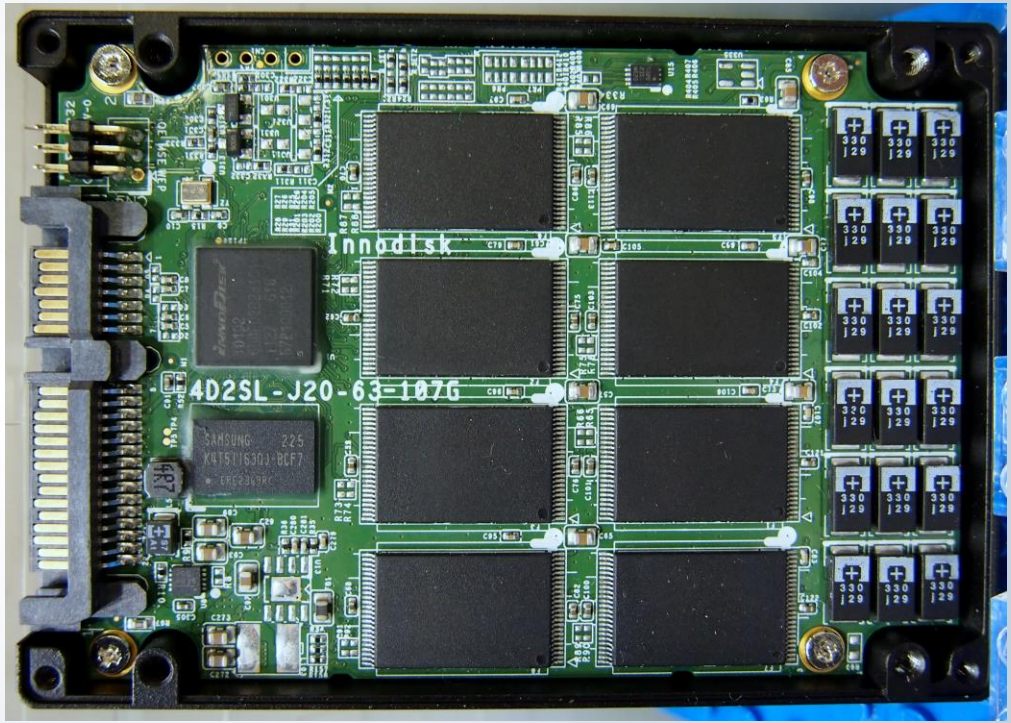
OSTEP Fig. 37.3

Types of Filesystems

- Many types of filesystems -- too many to list!
  - https://en.wikipedia.org/wiki/List_of_file_systems
- A few common ones:
  - ext4: the most recent ext filesystem; common on Linux (ext3, ext2 are older versions)
  - ZFS: supports very large files, journaling, snapshots, cloning, RAID, etc.
  - ReiserFS, Reiser4: journaling filesystem
  - NTFS: Windows filesystem (Linux driver has read and write support)
  - NFS/AFS: network filesystem; allows you to access files over a network
  - Ceph, GFS: high-performance distributed filesystems
- How do (disk-based) filesystems differ?
  - Different filesystems may be better/optimized for different things, like file sizes, speed of file access, integrity, etc.

Solid-State Drives

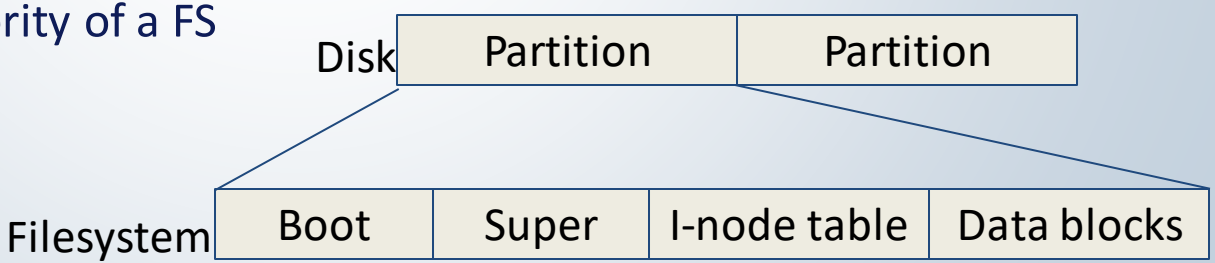
- Many devices today do not have spinning disks and instead have solid-state drives (SSDs)
- No arm to seek or platter to spin
- Other filesystem considerations for SSDs
  - Reduce writes to preserve device lifespan



Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

Filesystem Structure

- Boot block: always the first block in a FS
  - Not used by FS; contains info to boot the OS
  - Only one needed by OS
- Super block: contains parameter info about the filesystem
  - Size of the i-node table, size of logical blocks, size of the filesystem (in logical blocks)
- I-node table: contains one (unique) entry for every file in file system
  - Contains most of the “metadata” about individual file
- Data blocks: the (logical) blocks that contain the data for files and directories
  - This is the vast majority of a FS



I-Nodes

- Index nodes (i-nodes) contains the following metadata about a file
  - File type (for example, regular, char device, block device, directory, symbolic link)
  - Owner of the file
  - Group of the file
  - File access permissions (user, group, other)
  - Three timestamps:
    - Time of last access (ls -lu)
    - Time of last modification (default timestamp in ls -l)
    - Time of last status change (change to i-node info) (ls -lc)
  - Number of hard links to file
  - Size of the file (in bytes)
  - Number of blocks allocated to file
  - Pointers to the data blocks

Directories

- A directory is stored in a filesystem in a similar way as a regular file, but
  - It is marked as a directory in its i-node
  - It's a file with a special organization: it's a table consisting of filenames and i-node numbers
- Example is on the right
- Note: the i-node doesn't have a filename!
  - Implication: you can have multiple links to the same file!

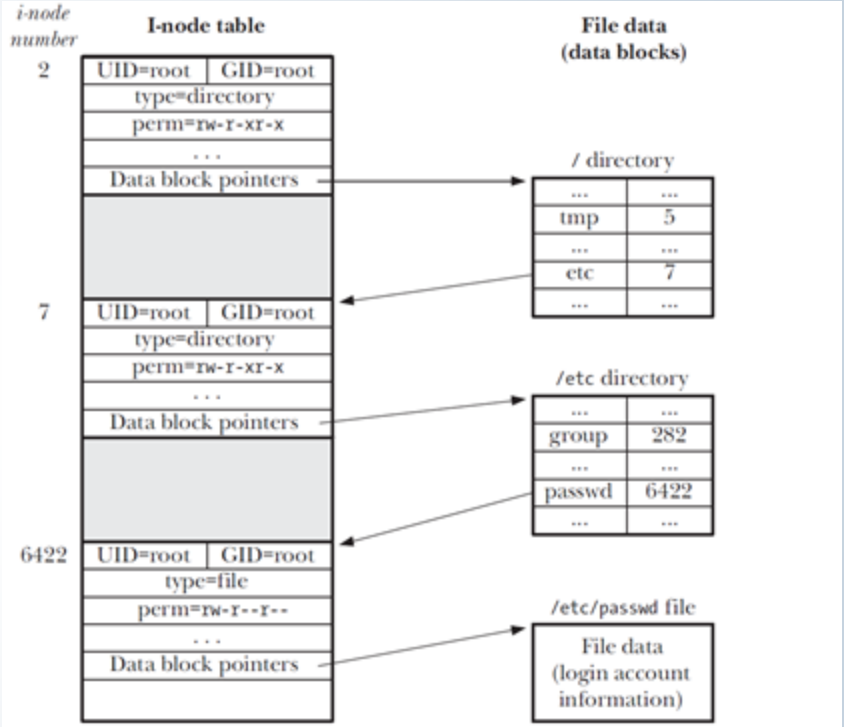


Figure 18-1: Relationship between i-node and directory structures for the file `/etc/passwd`

*Figure from *The Linux Programming Interface* by Michael Kerrisk

Data Blocks

- How can files of very different sizes be supported?
  - One method: store pointers to the data blocks!
- Figure on the right shows how ext2 does this
  - Small files might fit entirely in direct pointers
- Bigger files use:
  - Indirect pointers
  - Double-indirect pointers
  - Triple-indirect pointers
- Advantages of pointers
  - Fixed-size i-node
    - But arbitrary size files
  - Store blocks non-contiguously

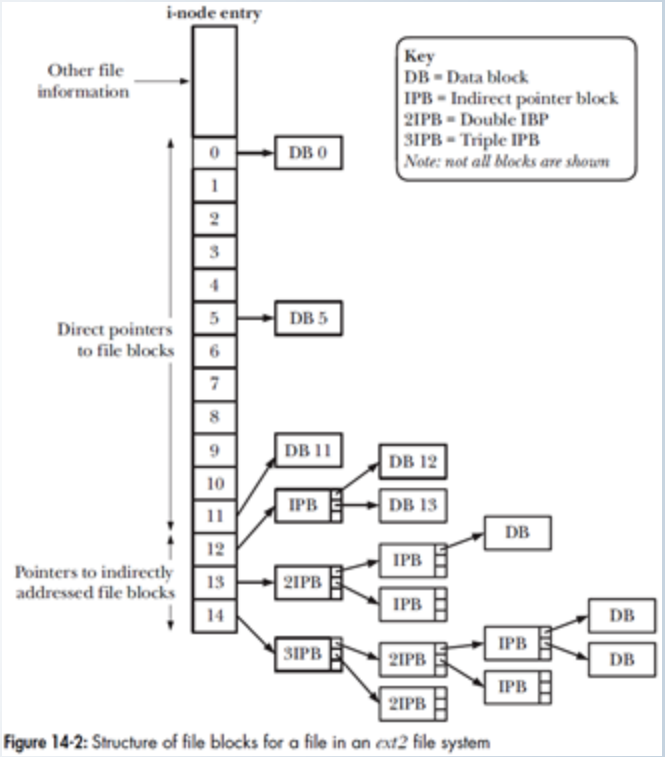


Figure 14-2: Structure of file blocks for a file in an *ext2* file system

*Figure from *The Linux Programming Interface* by Michael Kerrisk