Linux Filesystems

Advanced Embedded Linux Development

with **Dan Walkes**

University of Colorado **Boulder**

Learning objectives:

Understand use of files in Linux

Introduction to Inodes

Introduction to Links

Linux filesystems

“Everything Is A File”

- Much of the interaction with the kernel occurs via reading/writing files.
- Devices are accessed similar to files
  - Example /dev/ttyUSB0 for USB serial port.
- Common open, manipulate, close paradigm shared with files and devices.

Files in Linux

- Represented by paths.
  - Absolute “/path/to/file” - from root of filesystem.
  - Relative “path/to/file” - from working directory.
    - No leading “/” = relative path.

Linux Regular File Properties

- Bytes of data in a byte stream.
  - No structure enforced at the system level.
- File position/offset tracks location.
- Can be truncated to size smaller or larger than original.

Linux Regular File Properties

- Can be opened more than once, by different or same processes.
- Referenced and accessed by inode in the filesystem.

Linux Inode (Index Node)

- Metadata used to track files on disk.
- Includes timestamp, owner, size, mode (access permission), location.
- Fixed small (128 byte) size.
  - Filename is not included.

Inode					
Accessed Time	Size	UID	GID	Permission	Location On Disk

https://premaseem.wordpress.com/2016/02/14/what-is-inode-in-linux-unit/

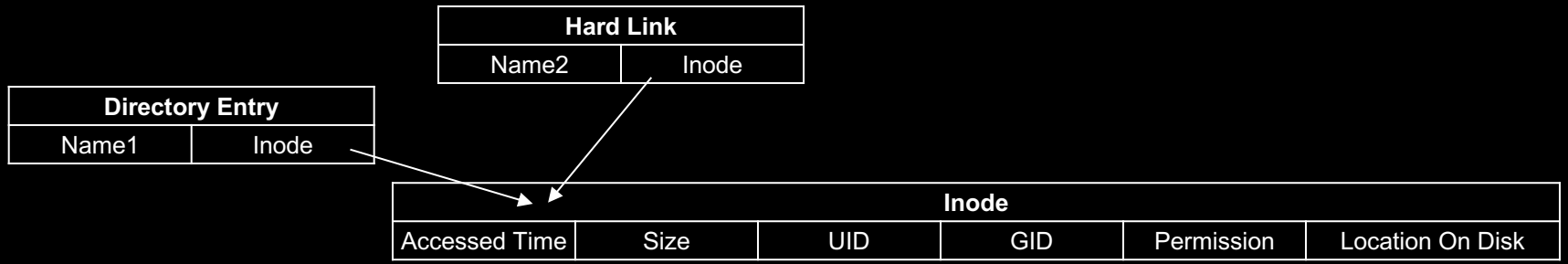
Linux Directories

- Map human readable names to Inode numbers.
- Actually just files (with their own inodes) containing mapping of names to Inodes.
- Start at the root directory (“/”)

Linux System Programming, Robert Love Chapter 1
https://premaseem.wordpress.com/2016/02/14/what-is-inode-in-linux-unit/

Linux Directories

- Why not include filenames in Inodes?
  - Allows different file names to share the same content without duplicating Inode content.



Linux System Programming, Robert Love Chapter 1
https://premaseem.wordpress.com/2016/02/14/what-is-inode-in-linux-unit/

Linux Links

- Flexibility of Inode design means multiple names can resolve to the same Inode.
- Links redirect two file/directory paths to the same Inode

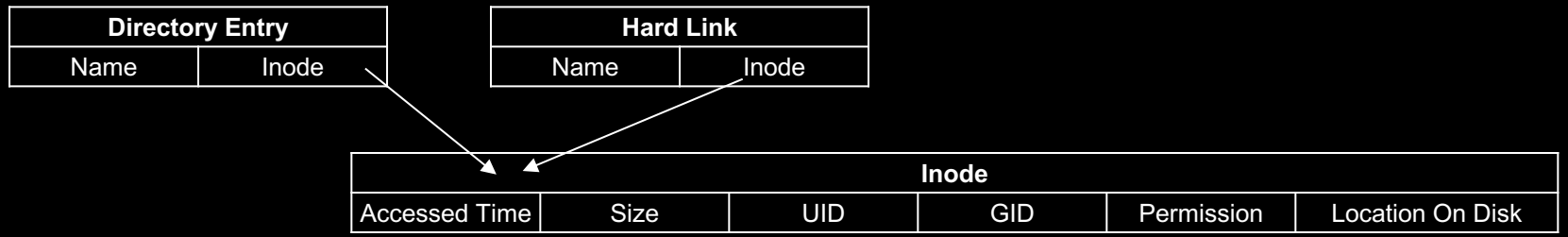
```
user@myhost:~/myfolder$ ls -l
total 4
-rw-r--r-- 1 user user 29 May  8 17:09 myfile.txt
lrwxrwxrwx 1 user user 10 May  9 05:55 mylink -> myfile.txt
```

Linux Links

- Two types of Links, Hard and Symbolic (symlinks)
  - Hard Links map directly to inodes, only allowed on the same filesystem.
  - Soft links map to filenames, work across filesystems, can be broken.

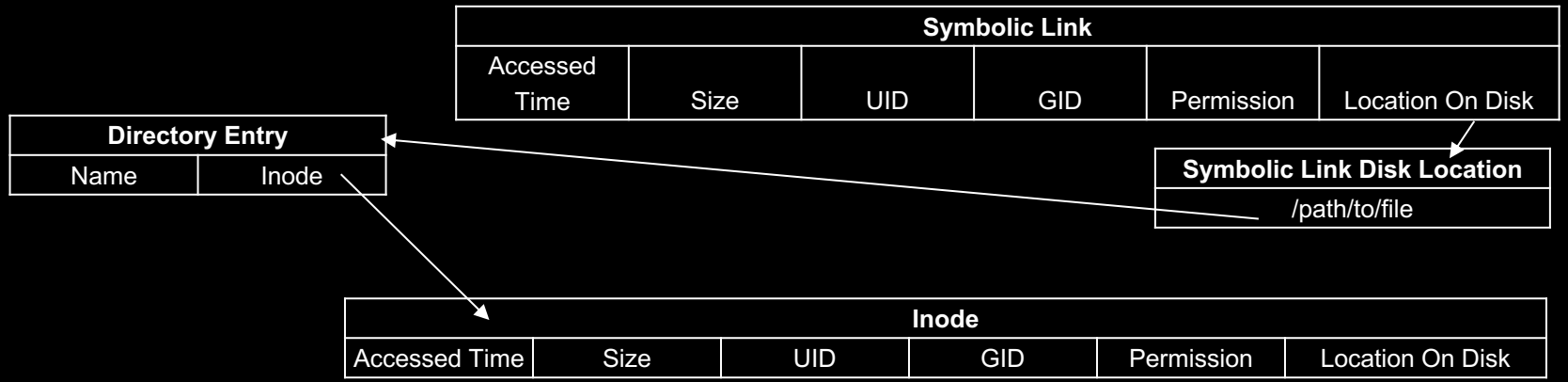
Hard Links

- Can’t span filesystems (Inode references a specific filesystem.)
- Conceptually the same as a Directory Entry.
- File deletes aren’t allowed until all references are deleted (preventing broken Hard Links).



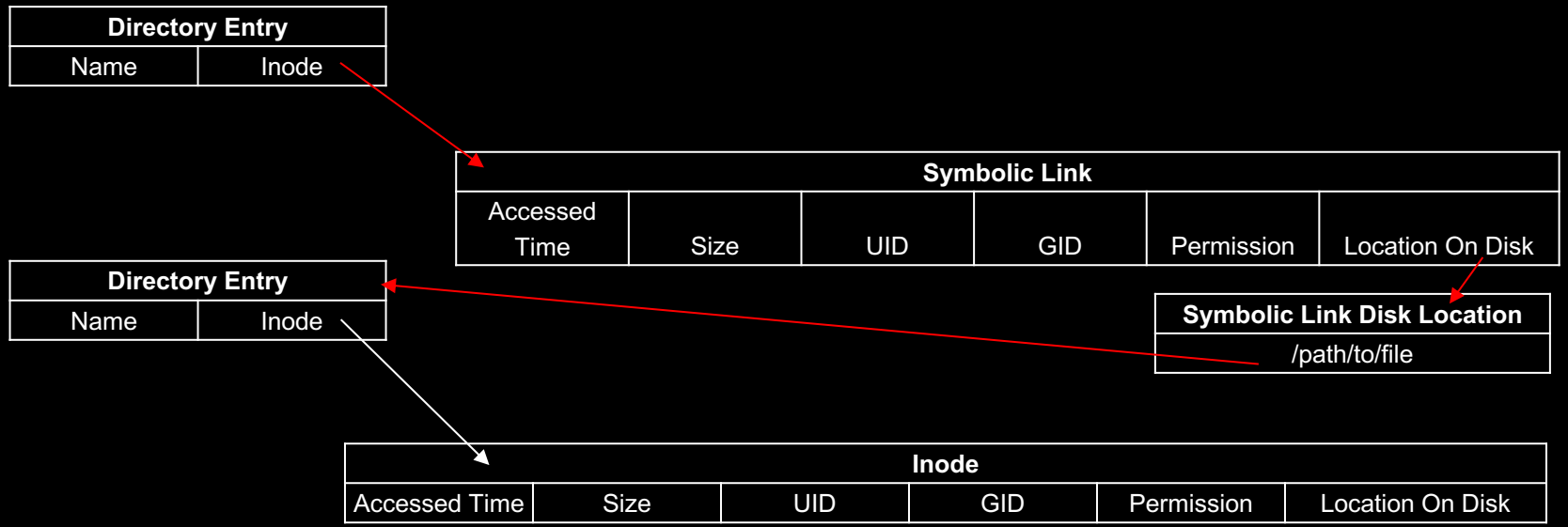
Symbolic Links (symlinks)

- Regular file with complete path in content
- Can point anywhere, including other filesystems, can break



Symbolic Links

- Requires extra steps to resolve



Linux Special Files

- Map hardware devices to the “everything is a file” paradigm.
- Kernel objects represented as files.
  - Character Device
  - Block Device
  - Named Pipes
  - Sockets

Linux Special Files

- Character Device
  - Linear queue of bytes
  - Example: Keyboard
- Block Device
  - Array of bytes, addressable in a sector
  - IE Hard Disk
- Named Pipes/Sockets
  - Interprocess Communication

Linux Filesystem

- Collection of files in a hierarchy
- Specific types supported, tied to storage types
  - NFS - network file storage
  - ext4 - block device storage
  - fat - Microsoft defined storage format for disks

Linux Filesystem

- Mounted/Unmounted to add/remove from the root filesystem.
- Smallest unit addressable is a block
  - block is a power of two multiple of sector size
  - Typical/historical sector size is 512 bytes

Linux Blocks/Sectors

- Consider a 1TB Filesystem.
  - 1,000,000,000,000 bytes.
  - 0x3B98CCA00 bytes.
  - 34 bits required to address individually.

Linux Blocks/Sectors

- 34 bits to address each byte.
- With 512 byte blocks:
  - Only 1,953,125,000 (0x746a5288) blocks.
  - 30 bits required to access each block.
- With 4K blocks:
  - Only 244,140,625 (0xE8D4A51) blocks.
  - 28 bits required to access each block.