ECE 531 – Advanced Operating Systems
Lecture 28

Vince Weaver

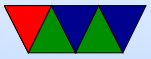
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

12 November 2025

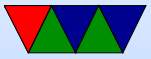
Announcements

- Homework #7 will be posted soon hopefully



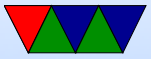
Reminder, Using a Block Device

- We can get blocks of data, but how do we find the blocks that we want...



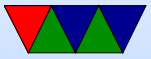
Filesystems

- Why use a filesystem?
  - Why not just open a disk raw, and remember that your senior project is at offset 1,000,000 and your jpegs all start at offset 4,005,434?
  - A good use of abstraction. Naming is useful.



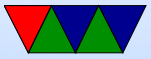
What is a file?

- In Unix is just a stream of bytes. That's not necessarily the only possibility
- Old systems files were just 80-column punch card images.
- Windows also has streams (with colon after name) MacOS has forks (what appears to be a file is more like a directory)



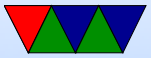
Where does a file live?

- Everything in one place / root directory?
- Hierarchical? Directories?
- Database?
- In the Cloud?



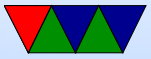
Why not just load everything into memory?

- Too big? Share with other processes?
- Persistent across reboots?
- In the old-days with core-memory your RAM might last between boots
- Modern Complication: NVRAM (non-volatile RAM) if it ever takes off, again contents of RAM survive reboot, how does that interact with caches and filesystems in an OS



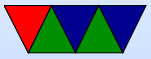
What Lives in a Filesystem?

- Regular files
- Directories
- Char devices, block devices
- Links (hard, soft)
- FIFOs



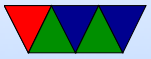
Filesystems metadata

- In addition to the file's contents (data) there is information about a file
- This is called metadata and is also stored in the filesystem



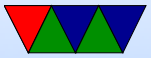
Metadata – Filenames on UNIX

- UNIX has few rules, makes some people unhappy
- Case sensitive, so Bob.txt bob.txt and BOB.TXT all different files
- Anything that is not / or NUL.
- What about "-rf" or * or ?
  What happens if you "rm *" and it matches -rf?
- What about linefeeds, spaces, backspaces?
- What about foreign chars? Emoji?
- What about console escape sequences?

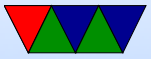


Metadata – Filename on other OSes

- Most other OSes have stricter rules which makes sharing files (network share, checking out from git, etc) fun
- Windows
  - more restrictions. Thins like CON and COM and PTR.
  - Uses backslash for directory separation (why?)
- DOS famously had 8.3 filenames
  Then weird compatibility came in with Win95 long filenames
- MacOS

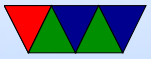


  - used to use : as directory separator.
  - Tries to normalize unicode?
  - Case-insensitive?
- Case sensitive? Bob or bob or BoB?
  What does it mean to be capital/lowercase? i18n

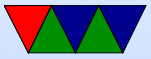


Filesystems metadata – attributes

- Permissions (read, write, execute)
- Owner (user, group)
- Timestamps
  - ctime – last status (metadata) change (originally create time) things like permissions change, ownership change, rename
  - mtime – last modified
  - atime – last access (why bad? convert read to a write? noatime/relatime)

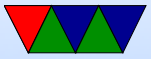


- Exact size
- Locks
- Hidden files (on DOS explicit attribute, Linux a leading dot on a file. Not intentional, ls was broken when trying to skip . and ..)
- Immutable / System
- Extended attributes / capabilities

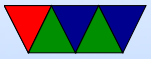


File Related Operations / System Calls

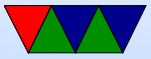
- syscalls that take filenames as parameters
  - Note: you don't have to specify full path. If you don't the OS tracks each process's cwd (current working directory). There can be race conditions because of this, so on Linux there are alternate new versions of these systemcalls with "at" (like openat()) that take the path as a parameter
  - open() – opens file. Can take a large number of arguments, some of which call into others below



most problems come when writing, as you can overwrite, append to end, truncate, etc
  - creat() – if file doesn't exist, create it (Ken Thompson story of missing e)
  - unlink() – delete file (we'll get into how it works later)
  - stat()/lstat() – return info on a file/link
  - rename() – rename (or move) a file
- syscalls that take file-descriptors as parameters
  - close()
  - read()
  - write()

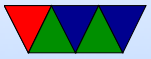


- lseek() – seek to file position
- fstat() – get info on files
- mmap() — Memory Map (map file into memory so can be accesed via loads/stores rather than read/write)



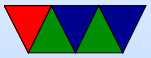
File Descriptors

- When open a file or object, get a number that indexes into a table, each referring to a file.
- Low-level syscalls mostly operate on file descriptors



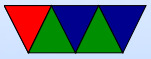
Directories/Folders

- Root directory
- Hierarchical
- Path names
  - '/' root director
  - '.' current directory
  - '..' parent directory



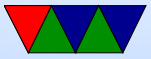
Directory/Folder Syscalls

- `opendir()`
- `closedir()`
- `getdents()` – old way of doing things, use readdir() instead now
- `readdir()`
- TODO: lookup the recent changes to Linux around this interface



Trouble with Readdir

- reading a directory is hard
- Ideally you'd read all entries atomically in one go
- If you don't, how do you restart/ask for the rest?
  What happens if someone is adding/deleting files at the same time
  Can your program handle files reported twice or not at all?
- How do you know how big to make the buffer?
- One way is to just keep trying buffers until it works



- This is still a bit of a mess on Linux, especially when dealing with filesystems that have no inherent idea of file order (old interfaces forced there to be a canonical order so you could restart at arbitrary position)