ECE 531 – Advanced Operating Systems
Lecture 17

Vince Weaver

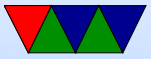
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

2 November 2023

Announcements

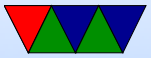
- Don't forget HW#6
- Homework #7 will be posted soon hopefully
- Project topics due
  Be sure to e-mail. This is worth points!



Device Drivers

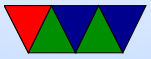
- So far we discussed talking to kernel via syscalls
- How do we talk to hardware from userspace without adding lots of custom syscalls?
- i2c, gpio, etc
- On Linux/UNIX have device drivers that you do file I/O on

  Everything is a File!



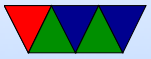
Device Type – Block Devices

- Read chunks of data with random access
- Can seek file location back and forth
- open/read/write/lseek
- Examples: disks, ramdisk, storage
- Filesystems often go on top



Device Type – Character Devices

- Read a sequence of characters incoming, write outgoing
- Can't seek around in this stream
- open/read/write/ioctl
- Examples: serial, i2c, spi, gpio

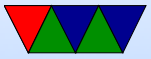


Device Nodes – Accessing Devices

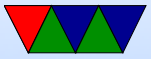
- Access by opening special files (often in /dev) and doing I/O

```
brw-rw---- 1 root disk 8, 1 Oct  6 10:07 /dev/s
```

- Node has a major / minor number which indicate what type of device and which one (if multiple supported)
- Also that b means “block”
- Device nodes are created with “mknod” and make a special device node in filesystem
  `mknod sda b 123 45` or similar

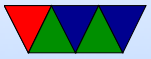


- When you open a device node, kernel looks up who owns it in table and maps this to proper device driver

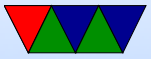


Auto-creation of Device Nodes (Linux)

- Manual creation of nodes
  - Old days if you added new device had to manually add node
  - Otherwise you distribution could just create every possible node under /dev
  - Previously: who maintained the list? Just text file in linux-kernel?
  - Also hit limits: only 16 hard drive partitions?
  - Something automatic preferable

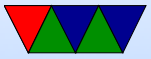


- Automatic creation
  - First attempt on Linux was devfs (flamewar)
  - udev/systemd now will auto-create these
  - Makes life easier

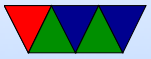


Device Workflow

- At boot, kernel sets up all devices
- The init() method for the device is called, devices "register" which device node they'll respond to
- User opens device node under /dev
- Kernel creates file descriptor entry
- This contains struct with info but also pointers for various methods like read/write/ioctl/close
- When you read/write on an opened device, the kernel used the fd to map your read call to a device_read call

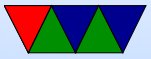


in the device driver in the kernel



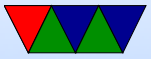
Other Topics

- Block devices – optimizing
  - prefetch
  - block scheduling (what happens when multiple processes reading from same block device. SSD random access not matter, old days spinning disk not efficient to seek around)
- Disk cache – disk access slow. How do we make slow things faster? Cache? Use unused RAM for this?
- Page cache?

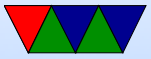


Disk performance

- Traditionally a lot of this came down to hardware.
- Spinning rust disks; head movement, cylinders/sectors. Reading consecutive faster, random access bad (millisecond bad)
  More complicated, fancy disk interfaces and embedded processors. Large caches (why can that be bad), shingled disks?
- Much of this goes away with flash disks, but still emulate old disk interface

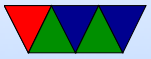


- Name lookup can also be slow.



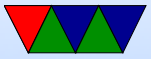
Disk Block Size

- Way too much overhead to have single byte granularity
- For a long time this was 512bytes/block
  Way too many for huge disks so disk drive companies pushing for 4k (but trying to remain backward compatible)
- Filesystems can allocate with larger blocksize.
- Large blocksize good: fewer blocks to track for each file
- Large blocksize bad: waste space on small files



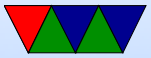
531 OS Device Drivers

- We have some device drivers, but not the common interface for them yet
- We don't even have storage for the device nodes to live



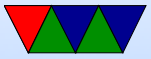
531 Storage / Block Device

- Need somewhere to store data that we can load
- block device, series of blocks
- Disk, but SD card drive complicated to write



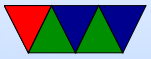
531 Storage / RAM disk

- How about RAM disk? Treat region of RAM as if it were a disk, can read/write blocks of data
- Can load data for block device along with kernel (tacked on)
- This is common thing to do with OS, sometimes called initrd



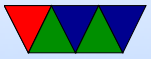
Using a Block Device

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



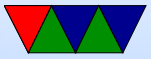
Filesystems

- 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.



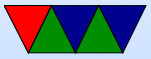
What is a file?

- 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)



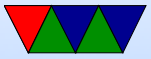
Where does a file live?

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



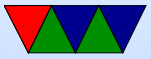
Why not just load everything into memory?

- Too big? Share with other processes?
- Persistent across reboots?



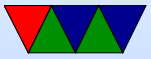
Filesystem File Types

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



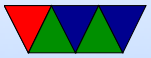
Filesystems metadata – Filenames on UNIX

- 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?

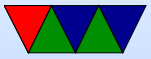


Filesystems metadata – Other OSes

- 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

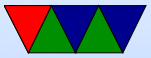


  - used to use : as directory separator.
  - Tries to normalize unicode?
  - Case-insensitive?
- Case sensitive? Bob or bob or BoB?
  What does it mean to be capital/lowercase? i18n



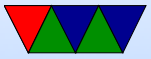
Filesystems metadata – attributes

- Permissions (read, write, execute)
- Owner (user, group)
- Access time (atime, ctime, mtime)
- Current size
- Locks
- Hidden
- Immutable / System
- Extended attributes / capabilities

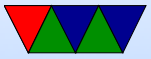


File Related Operations / System Calls

- 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 over-write, append to end, truncate, etc
- creat() – if file doesn't exist, create it (story of missing e)
- unlink() – delete file (we'll get into how it works later)
- close()
- read()

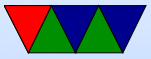


- write()
- lseek() – seek to file position
- stat()/fstat()/lstat() – get info on files
- rename() – rename (or move) a file
- mmap() — Memory Map



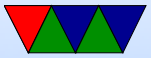
File Descriptors

- 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



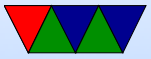
Directories/Folders

- Root directory
- Hierarchical
- Path names. /, ., ..
- Operations
  - opendir()
  - closedir()
  - getdents() – old way of doing things, use readdir() instead now
  - readdir()

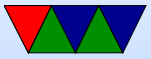


Trouble with Readdir

- 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

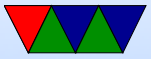


- This is still a bit of a mess on Linux, especially when dealing with filesystems that have no inherent idea of file order (old interfaces forced there to be a canonical order so you could restart at arbitrary position)



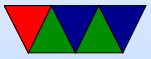
Low-level Disk Layout

- Often a MBR (master boot record) and partition table
- Disks divided into partitions
  - Why partitions?
  - Split up system (/, /boot, /usr, /home)
  - Why is boot separate? Smaller so boot loader can access, maybe a different fs type.
  - Dual-booting operating systems
  - Swap partitions



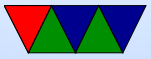
Filesystems – Organization

- A header containing master info (often called the superblock)
- Some sort of free list, saying what areas are free (bitmap or pointers)
- inodes, an array of data structures containing master info for each file (and if file is small, contents of file)
- Directory info: root directory entry, directory layout
- Actual file data



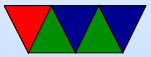
Filesystems – System Calls

- "mount" to put it in the proper place,
- "statfs" gives info on filesystem
  (including disk space, `df`)

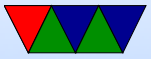


File Layout – Various Ways to Do It

- Contiguous.
  - Files in consecutive blocks.
  - Simple. Fast to read (just read X blocks)
  - Has fragmentation problems like with memory alloc.
  - What happens if append to file?
  - Ever used? read-only, CD-ROMs
- Linked list.
  - Inode points to first part, each block points to next.
  - No fragmentation, seeking through file involves lots of



reads.
  - Waste part of block size for next pointer
- File Allocation Table
  - Like linked list, but the links are stored in a separate area on disk
  - Can also instead have the pointers in one single block, each pointing to next block.
  - Whole thing has to be in mem at once. Makes it faster (no need to do lots of disk reads on seek) but problem if structure is big
- Inode table



- Special structure named inode that holds file attributes and list of blocks.
- If need more blocks then fit, last one points to another block with more.
- Only has to be in memory if file is open

- Database
  - Treat disk as if it were a database, with the files the info you want to retrieve