ECE 531 – Advanced Operating Systems
Lecture 32

Vince Weaver

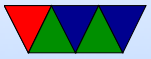
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

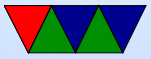
21 November 2025

Announcements

- Homework #8 will be posted
- Don't forget project update is due

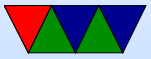


Advanced Filesystems



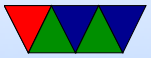
XFS

- Developed by SGI for Irix. Many iterations since then
- Designed for high I/O throughput
- Extents-based and B+ trees
- XFSv5 newest version (v4 deprecated)
- Bigtime – timestamps 64 bit nanoseconds (good to 2486)
- Up to 8 exabytes

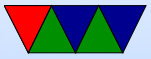


btrfs

- Butter-fs? Butter-fuss? B-tree fs?
- Started in 2007 at Oracle (by Chris Mason, who had worked on Reiserfs)
- Address scaling
- Lack of pooling, snapshots, checksums in Linux
  - Pooling – preallocate resources so they can be quickly handed out when needed
  - Snapshots – instead of taking full backup (long) just take a snapshot of current state and then keep using

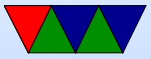


filesystem
  - Checksums – mathematically check to make sure values in files are what they should be
- $2^{64} = 16$ Exabyte file size limit (Linux VFS limits you to 8EB)
- Space-efficient packing small files
- Dynamic inode allocation

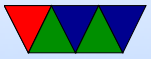


btrfs details

- Primary data structure is a copy-on-write B-tree
  - B-tree similar to a binary tree, but with pages full of leaves
    allow searches in logarithmic time
  - Btrees also used by ext4, NTFS, HFS+
  - Goal is to be able to quickly find disk block X
  - Copy-on-write when writing to file, rather than over-write (which is what traditional filesystems do)
  - Since old data not over-written, crash recovery better

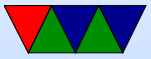


Eventually old data garbage collected

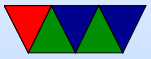


More btrfs details

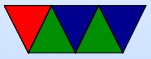
- Data in extents
- Forest of trees:
  - sub-volumes
  - extent-allocation
  - checksum tree
  - chunk device
  - reloc
- On-line defragmentation
- On-line volume growth



- Built-in RAID
- Transparent compression
- Snapshots
- Checksums on data and meta-data, on-line data scrubbing
- De-duplication
- Cloning, reflinks
  - can make an exact snapshot of file, copy-on-write
  - different inodes, initially point to same blocks
  - different from hardlink (different dir entry, point to same inode)

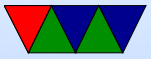


- In-place conversion from ext3/ext4
- Superblock mirrors – at 64k, 64MB,256GB, and 1PB. All updated at same time. Has generation number. Newest one is used.

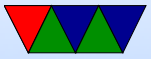


ZFS (zettabyte fs)

- Advanced FS from Sun/Oracle
- 128-bit filesystem (opposed to btrfs which is 64-bit)
  Running out of space would require 10^{24} 3TB hard drives
- Not really included in Linux due to licensing issues (CDDL vs GPL2)
  Was originally proprietary, then open source, then proprietary again (with open fork)
- Vaguely similar in idea to btrfs
- indirect still, not extent based?

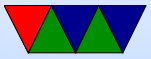


- Acts as both the filesystem *and* the volume manager (RAID array)
- Aim is to be super reliable, to know the state of underlying disks, make sure files stay valid, drives stay healthy
- Can take snapshots. Can roll back if something goes wrong.
- Checksums. Stored in parent. Other fs stores with file metadata so if that lost then checksum also lost
- Limitations: needs lots of RAM and lots of free disk space (due to copies and snapshots). If less than 80%



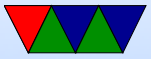
free then goes to space-conserve mode rather than high-performance

- Supports encryption (btrfs doesn't yet)



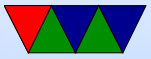
ReFS

- Resilient FS, codename “Protogon”
- Microsoft’s answer to btrfs and zfs
- Windows 8.1
- Initially removed features such as disk quotas, alt data streams, extended attributes (added later?)
- Uses B+ trees (not same as b-trees), similar to relational database
- All structures 64-bit
- Windows cannot be booted from ReFS



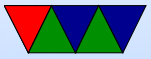
APFS

- New Apple OS for High Sierra and later, iOS 10.3 later
- Fix core problems of HFS+
- Optimized for solid-state drive, encryption
- 64-bit inode numbers
- checksums
- Crash protection: instead of overwriting metadata, creates new metadata, points to it, and only then removes old
- No hard-links to directories (most other OSes are like



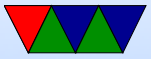
this) but this breaks “Time Machine” backup

- HighSierra auto-converts flash-based drives



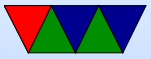
Embedded

- Designed to be small, simple, read-only?
- romfs
  - 32 byte header (magic, size, checksum,name)
  - Repeating files (pointer to next [0 if none]), info, size, checksum, file name, file data
- cramfs
- Filesystems optimized for flash storage?

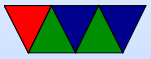


Networked File Systems

- Can you have non-local filesystems, where files come from a network server?
- Allow a centralized file server to export a filesystem to multiple clients.
- Two ways to do this
  - File level access, where syscalls are sent to the server (like open, read, write)
  - Block level access, where the remote server acts like a block device and any OS can be run over it (NAS,

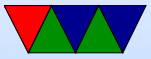


network-attached storage)



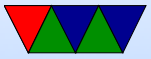
Cluster File Systems

- We learn about these in ECE574 Cluster Computing
- Distributed, fault tolerant, multiple servers working together
- Things like Lustre



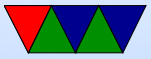
Cloud Filesystems?

- Blur the lines
- Can appear to be part of a filesystem even though maybe served over http requests
- You can actually make an httpfs if you want

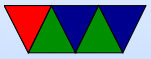


NFS – Network File System (NFS2/3/4)

- Developed by Sun in the 80s.
- Stateless. Means server and client can reboot without the other noticing.
- A server, nfsd, exports filesystems as described in `/etc/exports`. The server can be in userspace or in the kernel
- Needs some sort of "file handle" unique value to specify value. Often cheat and use inode value. Problem with older version of protocol with only 32-bit handles.

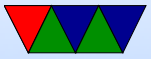


- UDP vs TDP
- Read-ahead can help performance
- Cache consistency a problem. One way is to just have timeouts that flush data regularly (3-30s)
- List of operations (sort of like syscalls) sent to server
  read sends a packet with file-handle, offset, and length
  No open syscall; server has no list of open files. This way there is no state needed, can handle reboots.
- nfsroot



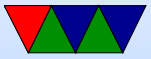
CIFS/SMB

- Windows file sharing.
- Poorly documented
- Samba reimplements it, originally reverse-engineered.



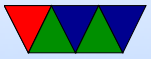
Virtual/Pseudo Filesystems

- Filesystem interface is an easy way for the OS kernel to share information
- Can have files that do not exist on disk
  they are virtual, fake files that the kernel creates dynamically in memory
- Examples: /proc, /sys/, /debugfs, /usbfs
- There are often pushes for faster ways of getting this data (netlink?)

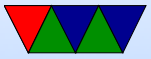


procfs

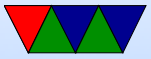
- Originally process filesystem. Each process gets a directory (named by the process id (pid)) under /proc
  Tools like `top` and `ps` use this info.
  - cmdline
  - cwd
  - environ
  - exe
  - fd
  - maps



- Eventually other arbitrary files were also included under proc, providing system information
  - cpuinfo
  - meminfo
  - interrupts
  - mounts
  - filesystems
  - uptime
- ABI issues – these files are part of the kernel, and even though the intention was that they could come and go at will, enough people write programs that depend

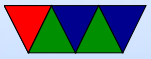


on them, the values cannot be easily changed without breaking the ABI



sysfs

- procfs was getting too cluttered, so sysfs was created
- intended to provide tree with information on devices
- one-item per file and strict documentation rule
- also hoped that it would replace `sysctl()` and `ioctl()` but that hasn't happened



FUSE

- Allows creating filesystem drivers in userspace
- Works on various OSes