Project #5: CROFS: Compressed Read-Only File System

Systems Software &
Architecture Lab.
Seoul National University

2026.06.05

xv6 File System On-Disk Layout

- **Boot: not used (1 block)**
- **Super block: basic information of the file system (1 block)**
- **Log: write-ahead logging for crash consistency (LOGBLOCKS blocks)**
- **Inode: inode table (NINODES blocks)**
- **data bitMap: bitmap to track data block usage (size derived from FSSIZE)**
- **Data blocks**

CROFS Overview

- **Read-only file system**
- **The entire image is produced on the host by a new builder, mkcrofs.**
  - Kernel does not allocate or free CROFS blocks
  - Kernel interprets CROFS metadata and returns the uncompressed bytes
- **Compressed contents, uncompressed metadata**
  - Directories and metadata stay uncompressed
  - Regular-file contents may be compressed into independently readable chunks.

CROFS On-Disk Layout

- **B**oot: not used (block 0)
- **S**uper block: CROFS superblock (block 1)
- **I**node (uncompressed)
- **D**ata blocks (files, directories, and other metadata)

CROFS Superblock

- **Required fields for grading (do not remove them):**
  - magic, size, nblocks, ninodes, inodestart, datastart, plainsize
- **You may freely add more fields**
- **size = number of 1 KiB blocks used by the image; basis for the compression ratio**

```
// On-disk CROFS superblock structure (@ kernel/fs.h)
#define CROFS_MAGIC   0x43524653    // "CRFS"

struct superblock {
  uint magic;         // Must be CROFS_MAGIC
  uint size;          // Size of file system image (blocks)
  uint nblocks;       // Number of data blocks
  uint ninodes;       // Number of inodes.
  uint inodestart;    // Block number of first inode block
  uint datastart;     // Block number of first data block
  uint plainsize;     // Size of uncompressed file system image (blocks)
};

  // You may add more fields here...
};
```

CROFS Inode

- **Fixed 64-byte on-disk inode, as in the original xv6 FS**
- **First five fields (type, major, minor, nlink, size) are fixed; the rest may be modified**
- **For regular files, size = uncompressed logical file size**
  - Observed by read(), lseek(), fstat(), and exec()

```
// On-disk CROFS inode structure (@ kernel/fs.h)
struct dinode {
  short type;           // File type
  short major;          // Major device number (T_DEVICE only)
  short minor;          // Minor device number (T_DEVICE only)
  short nlink;          // Number of links to inode in file system
  uint size;            // Size of file (bytes)

  // The remaining fields may be repurposed for CROFS
  uint addrs[NDIRECT+1];   // Data block addresses
};
```

Compression Algorithms

- **Use any standard codec: RLE, Huffman, LZSS, LZ4, etc.**
- **Rules:**
  - A read at any offset returns the original uncompressed bytes
  - A random read near EOF must not decode from the file start
  - Metadata must locate the compression unit for an offset quickly
  - Store a region uncompressed if compression makes it larger
  - State your maximum compression-unit size in the design doc

Interfacing with the Buffer Layer

- **Read all compressed blocks through bread() (release with brelse())**
  - xv6's buffer cache holds the compressed disk blocks
- **Decompressed data is temporary, read-time state only**
- **Your design must NOT:**
  - Bypass bread(), or keep a private compressed cache
  - Cache decompressed data or 1 KiB blocks for reuse
  - Keep whole files / the file system, or hidden copies, in memory

Fast 1 KiB Random Reads

- **Grading measures 1 KiB random reads**
  - Including files whose size is not a multiple of 1024
- **Provide a fast path: a 1 KiB logical block -> its compressed region**
- **Whole-file compression without an index is not acceptable**

Part 1. Implementing lseek() system call (10 points)

- **int lseek(int fd, int offset, int whence)**
- **Changes the current file offset; regular files only (pipes/devices may return -1)**
  - whence: SEEK_SET (0), SEEK_CUR (1), SEEK_END (2)
- **Returns the new offset, or -1 on failure**
  - Negative result is invalid -> -1
  - Seeking past EOF is allowed, but a later read past EOF returns 0
- **Grading does random reads by repeated lseek() + read()**

Part 2. Building CROFS Image with mkcrofs (30 points)

- **Build a bootable fs.img with the host-side builder mkcrofs**
  - Superblock at block 1; inode region for the included files/dirs
  - Preserve the original xv6 directory-entry format
- **Compress file contents; fall back to uncompressed when it does not help**
- **Create the /console device entry; report size accurately; store nothing past size**
- **Must be deterministic**

Part 2. Building CROFS Image with mkcrofs (30 points)

- **mkcrofs tool must print a summary of the generated CROFS image**
- **The output must be**
  - magic: <CROFS magic number>
  - size: <total number of 1 KiB blocks in the CROFS image>
  - nblocks: <number of data blocks>
  - ninodes: <number of inodes>
  - inodestart: <first block of the inode region>
  - datastart: <first block of the data-block region>
  - plainsize: <estimated number of 1 KiB blocks for an uncompressed xv6-style image>
  - compression ratio: <plainsize / size>

Part 3. Functional Correctness (20 points)

- **Mount & validate: read superblock, check magic/geometry, never read a block >= size**
- **Read: directories, compressed files, sequential + random (via lseek)**
  - exec() from CROFS, correct fstat(), handle /console
- **Reject write operations cleanly (no panic)**
- **Full credit also needs compression ratio plainsize / size >= 1.1**

Part 4. Space-Performance Tradeoff (30 points + bonus 20 points)

- **Scored on R_s x R_t (space improvement x read-time improvement)**
  - R_s = plainsize / size
  - R_t = uncompressed_read_time / crofs_read_time (1 KiB random reads)
- **R_t from disk reads (10,000 IOPS) + decompression cycles (rdcycle, 2 GHz, -icount shift=0)**
  - The baseline is the estimated time to read the corresponding uncompressed 1 KiB blocks directly from disk.
- **No whole-image decode at boot; no full or hidden decompressed copies**
  - bflush() / crofs_stats() used for measurement; ranking among Part 3-correct only

Part 5. Design Document (10 points)

- **Submit a design document in a single PDF file**
- **Your document should include the following sections.**
  1. **On-disk layout**
  2. **Compression design**
  3. **Kernel integration**
  4. **Testing and validation**

Skeleton Code

- **Work on the pa5 branch; set your STUDENTID in the Makefile**
  - git clone https://github.com/snu-csl/xv6-riscv-snu && cd xv6-riscv-snu && git checkout pa5
- **Layout is already read-only CROFS:**
  - log & bitmap removed; superblock has CROFS_MAGIC + plainsize; inode addrs[] repurposed
- **Implement CROFS read path and decompression logic in crofs.c / crofs.h:**
  - crofs_init(), crofs_readi_file(), crofs_decompress()
- **Do NOT modify fshook.c**
  - call the decompress() wrapper, not crofs_decompress() directly (it records stats)
- **mkcrofs creates /console in the image**
  - bread() , bflush(), crofs_stats() are instrumented for grading

Tips

- **Read Chap. 10 of the xv6 book to understand the file system implementation in xv6.**
- **For your reference, the following roughly shows the required code changes; each + denotes about 1~10 lines to add, remove, or modify.**
  - mkfs/mkcrofs.c | ++++++++++++++++++++++++++++++++++++
  - kernel/fs.h | +
  - kernel/fs.c | ++++
  - kernel/sysfile.c | ++++
  - kernel/crofs.h | +++++++
  - kernel/crofs.c | ++
  - **The actual amount of code in mkcrofs.c and crofs.c can vary greatly depending on your compression/decompression algorithm**

Notifications

- **CROFS is read-only**
  - create/delete/write on regular files must fail cleanly (no panic); one data block holds one file
- **Modify only the permitted files**
  - ./mkfs/mkcrofs.c, ./kernel/fs.h, ./kernel/fs.c, ./kernel/sysfile.c, ./kernel/crofs.h, ./kernel/crofs.c
- **Use QEMU 8.2.0 or later; grading runs take time, so start early**
- **Check https://github.com/snu-csl/os-pa5 for details**

Due date

- Due
  - 11:59 PM, June 21 (Sunday)

- Submission
  - Run the make submit command to generate a tarball named xv6-pa5-{STUDENTID}.tar.gz in the xv6-riscv-snu directory
  - Upload the compressed file to the submission server
  - You must also upload your design document as a PDF file
  - The total number of submissions for this project will be limited to 30
  - Only the version marked FINAL will be considered for the project score

Thank you!