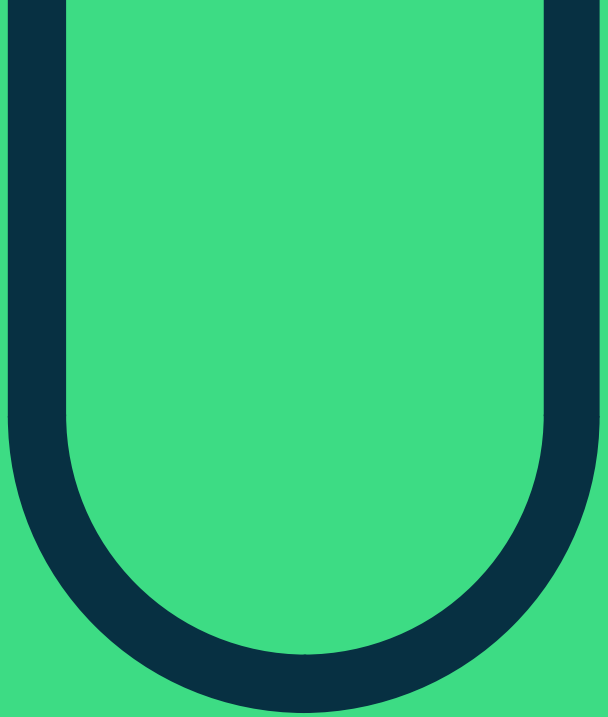
Incremental FS

Systems - Functionality Talk



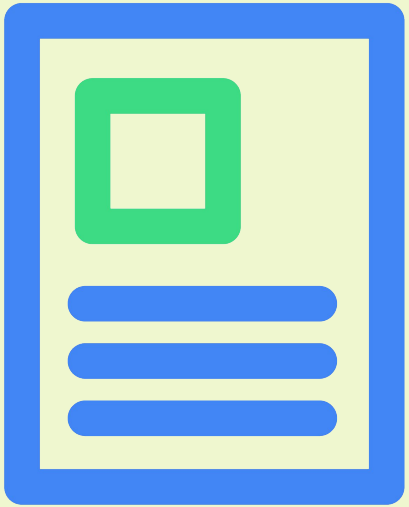
android

Paul Lawrence

Software Engineer

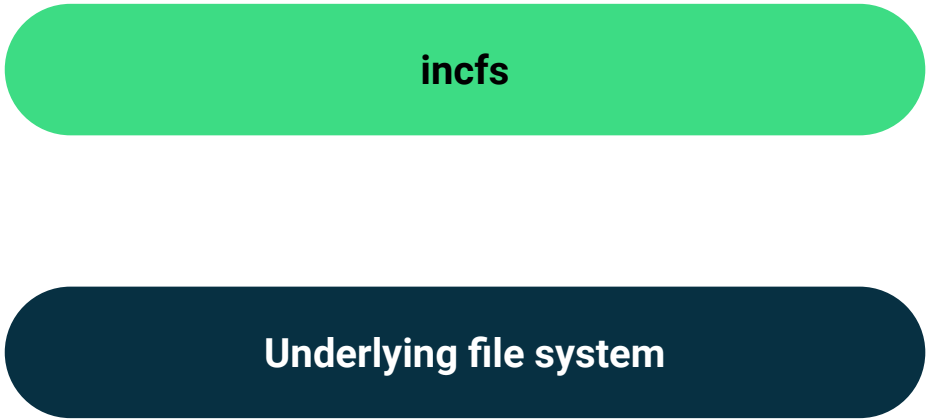
Requirements

- Read only files, read/write file system
- Files can be delivered at the block level incrementally and out of order
- Reads from undelivered parts of a file trigger a notification to user space, holding the read until fulfilled or timed out
- File contents can be compressed
- Files are verified on read
- No limit on file count



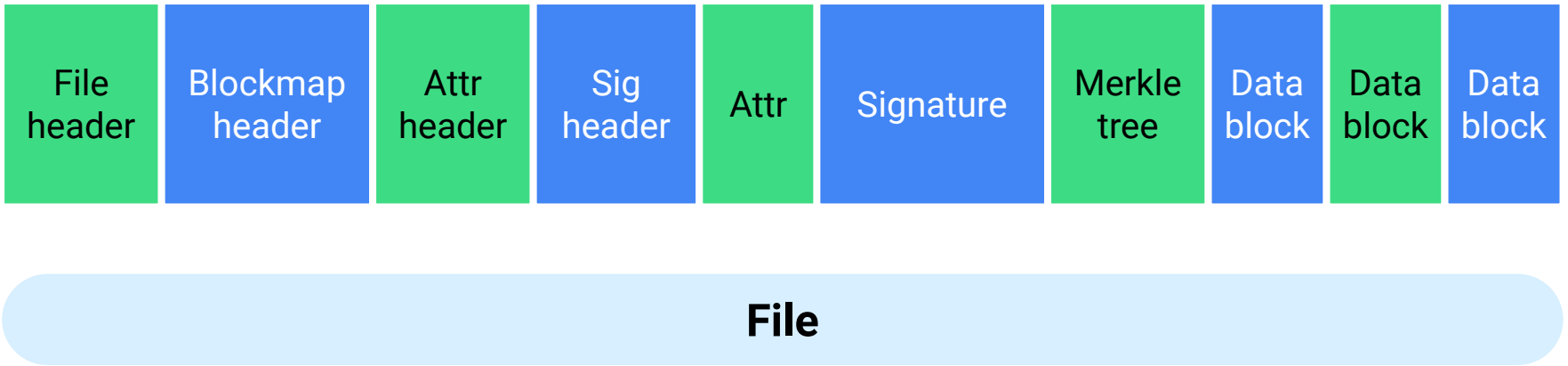
Implementation

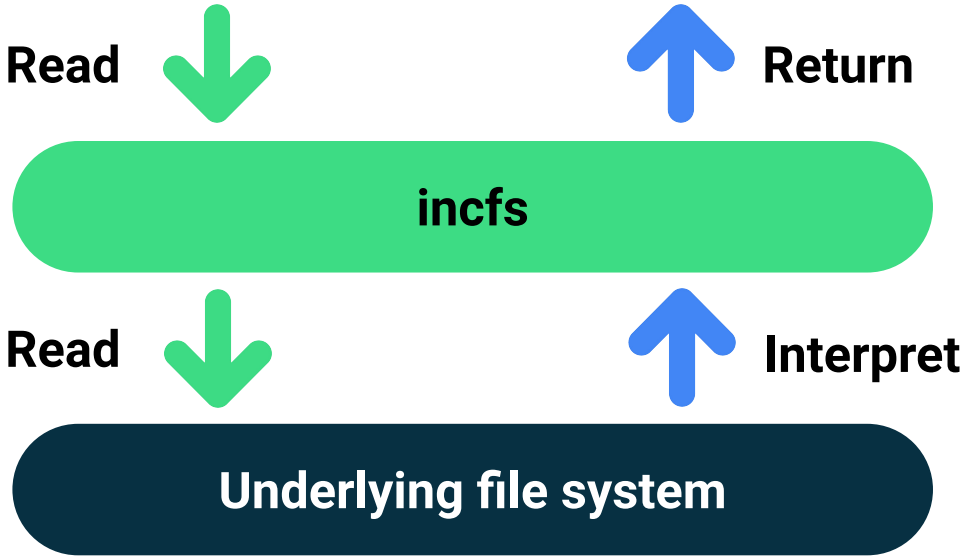
- Stacking file system
- Each file has a corresponding file in the underlying file system
- Directory operations (move, delete, link, etc.) are passthrough
- Reads interpret the underlying file and return the correct data
- Writes are interpreted specially and provide the channel for incremental delivery
- An index directory exists to manage incremental delivery
- Two command files and two IOCTLs round out the interface

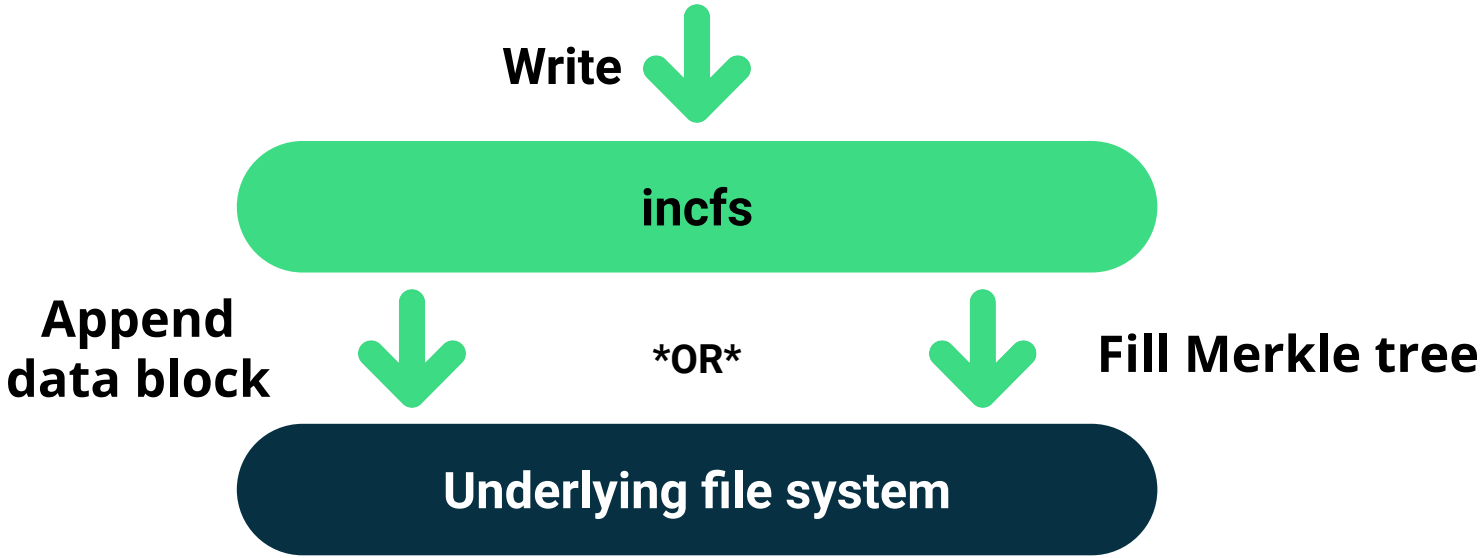

incfs
Underlying file system

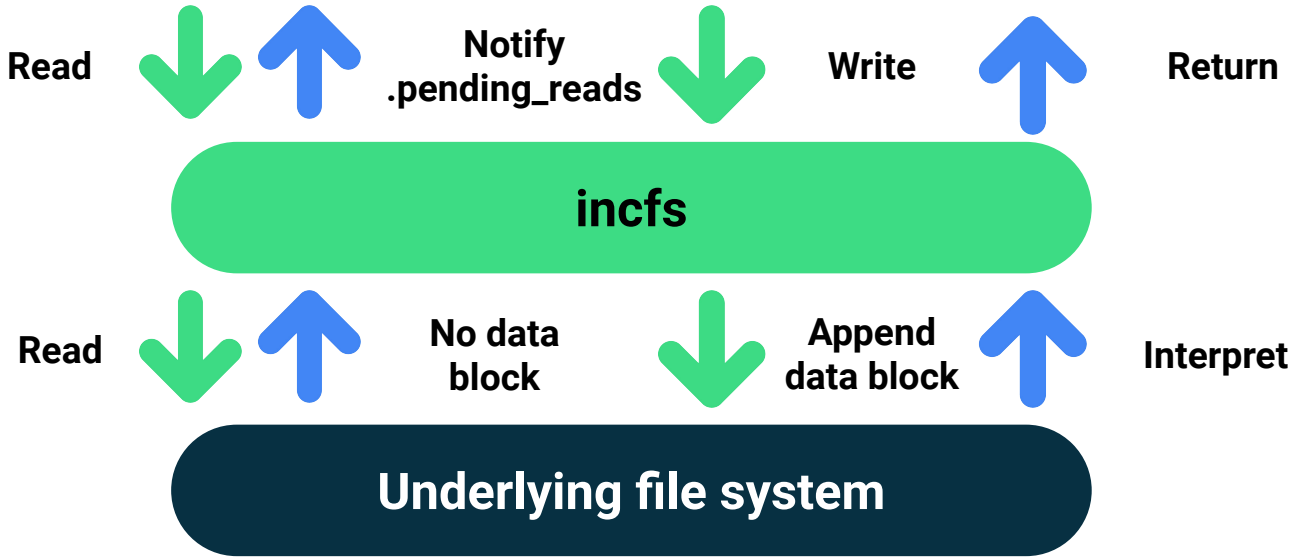
Identical files/directories +
.pending_reads and .log files in root

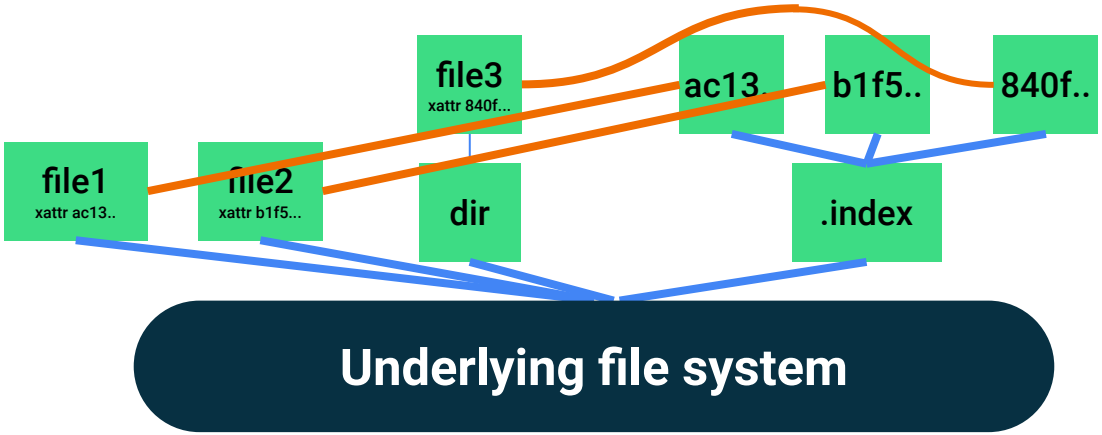
Underlying file system


File header
Blockmap header
Attr header
Sig header
Attr
Signature
Merkle tree
Data block
Data block
Data block
File


Read
Return
incfs
Read
Interpret
Underlying file system

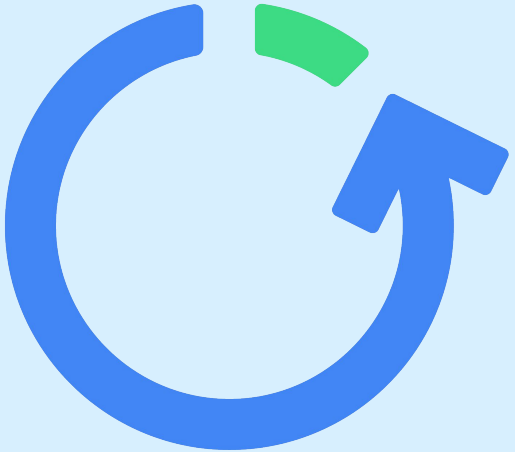

Write
incfs
Append
data block
OR
Fill Merkle tree
Underlying file system

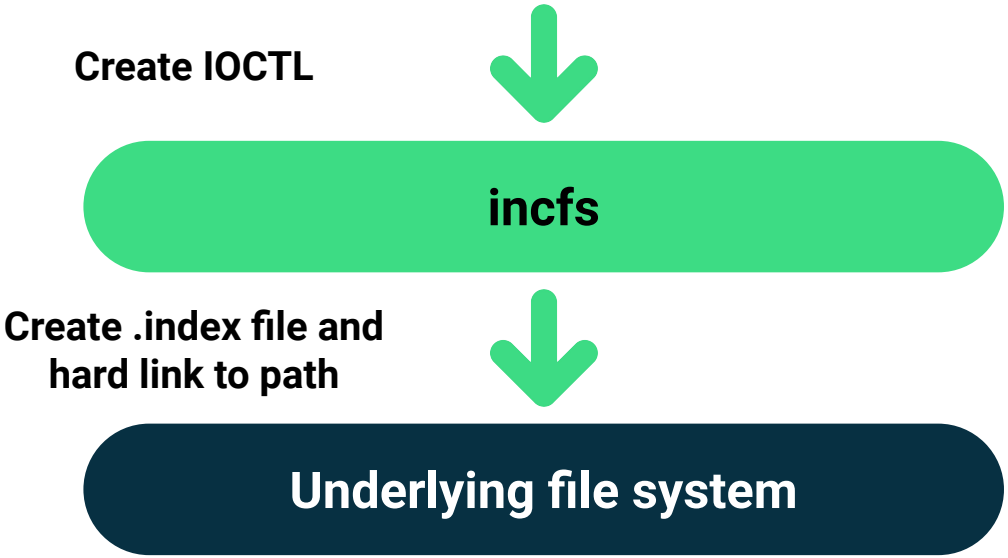

Read
Notify
.pending_reads
Write
Return
incfs
Read
No data
block
Append
data block
Interpret
Underlying file system


file3
xattr 840f...
ac13..
b1f5..
840f..
file1
xattr ac13..
file2
xattr b1f5...
dir
.index
Underlying file system

Directory Operations

- Most directory operations are simply reflected in the underlying file system e.g., move, link, set attributes
- unlink is slightly special - if after unlink the file has only one reference, it's the `.index` reference and the file is removed completely
- File create is blocked, because we need some parameters to create a file. Instead:






```
struct incfs_new_file_args {
        /* Id of a file to create. */
        incfs_uuid_t file_id;

        /* Total size of the new file. Ignored if S_ISDIR(mode). */
        __aligned_u64 size;

        /* File mode. Permissions and dir flag.
        __u16 mode;

        /* A pointer to a null-terminated relative path to the file's
                parent dir. Max length: PATH_MAX */
        __aligned_u64 directory_path;

        /* A pointer to a null-terminated file's name. Max length:
            PATH_MAX */
        __aligned_u64 file_name;

        /* A pointer to a file attribute to be set on creation. */
        __aligned_u64 file_attr;

        /* Length of the data buffer specified by file_attr.
            Max value: INCFS_MAX_FILE_ATTR_SIZE */
        __u32 file_attr_len;

        /* struct incfs_file_signature_info *signature_info; */
        __aligned_u64 signature_info;
};
```

Why .index?

- Pending read notifications have to say which file is being read
- But a file might be moved while the read is pending
- The `.index` directory and the associated `xattr` provide a way of going to and from a file unambiguously

Sundry

- Each block write can pass in the data lz4 compressed, and the block is then stored compressed and decompressed on read; blocks in the underlying file are therefore not necessarily block aligned
- There is one more ioctl, which is to get a file's signature for validation
- The `.log` file tracks file reads which allows us to gather data to optimize the delivery of blocks

Future work

- fs-verity integration
- Mapped files. Many Android files are extracted from incrementally delivered APKs. To avoid having to fully download those sections of the APK, add an ioctl to create a file as a offset/size of an existing file
- User space control of pending read timeout