DECISION BRIEF · EFFECT V4 FILESYSTEM

An inode core is necessary, but not yet the layer

The reusable suite gives a memory filesystem a concrete target. PR #456 contains a promising storage-engine sketch, but a conforming v4 layer still needs a rebuilt resolver, descriptor lifecycle, adapter boundary, and policy decisions.

23 JULY 2026 RESEARCH ONLY · NO REPOSITORY FILES CHANGED

RECOMMENDED BASE
Reuse the inode / directory-entry / descriptor model, not the PR module unchanged.

LARGEST WORK
Path resolution, link/open-handle lifetime, tree mutation, glob, and watch.

TEST BOUNDARY
51 scenarios plus 10 open-mode cases define core portable behavior.

The suite makes object identity and handle lifetime non-negotiable

VERIFIED The suite requires open descriptors to survive rename and unlink, hard links to share one object, and relative/absolute symbolic links to resolve. A path-to-bytes map cannot meet that contract.

What the suite forces a memory layer to model

Group	Requirement	Consequence
Descriptors	Ten flags, independent cursors, EOF, sparse writes, scoped close.	Descriptor table with mode, position, append and closed state.
Identity	Rename/unlink keeps existing handle usable; links share data and metadata.	Inode link count plus open-handle retention.
Links	Relative/absolute resolution and loop errors.	Component-wise resolver and loop limit.
Derived APIs	Streams/sinks finalise handles.	Use `FileSystem.make`; make `open` correct first.

Inspect FileSystemTest.ts

PR #456 is a credible core sketch, with two mandatory rewrites

VERIFIED Fubhy's WIP uses an inode map, directory-entry maps, and descriptor positions—the right foundations. It is a standalone collection API, however, not a v4 `FileSystem` service or `Layer`.

DO NOT PORT THE RESOLVER OR LIFECYCLE VERBATIM

The prototype normalises `..` before symlink resolution, resolves relative targets from cwd rather than a link's parent, deletes inodes at link-count zero, and does not validate descriptor membership after close. Those rules conflict with the new contract.

What to take from the prototype

Status	Scope	Action
Reuse	Inodes, entries, byte growth, descriptor positions, stat/lstat distinction.	Adapt as native v4 internals.
Repair	Flag mechanics and basic create/truncate/append.	Add v4 mapping, scoped facade, Clock, errors and close checks.
Build fresh	Resolver and reclamation.	Parent-relative links; reclaim only after link and open counts reach zero.
Absent	Layer, copy/rename/remove, links, temps, glob, watch, File facade.	New adapter and tree/event subsystems.

VERIFIED The PR's own notes defer service integration, streaming, watch and temporary resources.

Inspect PR #456 and WIP notes

Keep the public layer thin; isolate nine responsibilities

INFERRED Recommended split: `MemoryFileSystem.ts` owns `make` / `layer` and isolated state; internals own State, Path, Descriptors, Tree, Metadata, Temp, Glob, Watch, and operation-specific PlatformError helpers.

Effort follows semantic coupling, not method count

Effort	Methods	Reason
High	`open` + File; `remove`, `rename`, `copy` / `copyFile`; `glob`; `watch`	Crosses identity, lifecycle, multiple entries, or streams.
Medium	Directories, whole-file I/O, `truncate`, `realPath`, links, symlink, temp APIs	Depend on resolver/tree invariants.
Low	`access`, `chmod`, `chown`, `utimes`, `stat`, `readLink`	Metadata projection/setters, subject to policy.
Derived	`exists`, text helpers, stream, sink	`FileSystem.make` supplies these, but they expose handle bugs.

Decide the virtual contract before filling out the adapter

1. **Paths:** POSIX-rooted namespace, cwd, and relative-path policy.
2. **Metadata/access:** synthetic uid/gid/mode; stored-only permissions or virtual identity.
3. **Symlinks:** final-link rule, loop tag and resolution limit.
4. **Atomicity:** mutation serialization and watcher emission point.
5. **Copy:** overwrite=false, timestamps and symlink behavior.
6. **Glob/watch:** grammar, normalization, ordering, cleanup and backpressure.
7. **API gaps:** seek result, truncate cursor, EBADF normalisation, watch error method.

> SEPARATE CONCERNS
>
> Snapshot, diff, conflict resolution and disk application belong above an interchangeable filesystem layer.

Build in conformance-shaped slices

1. **Core identity:** root, resolver, inode/entry state, errors, bytes and directories.
2. **Descriptors:** flags, scoped File, cursors, sparse I/O, rename/unlink retention.
3. **Tree mutations:** recursive remove, links, realPath, rename and copy.
4. **Completeness:** temps, timestamps, glob and remaining suite cases.
5. **Events:** watch after mutation atomicity and path normalisation.

Add a `MemoryFileSystem.test.ts` entrypoint for the shared suite, then memory-specific adversarial cases: nested relative links, layer isolation, deterministic time and event lifecycle.

Evidence and limits

INSPECTED the current FileSystem constructor/API, Node adapter shape, shared suite, PlatformError, and PR #456 source/spec snapshot. No memory implementation was added or executed.

- FileSystem.make and primitive contract
- Node adapter shape
- effect-smol PR #456

VALIDATION LIMIT A targeted Node suite run was attempted but pnpm stopped at a non-interactive modules-directory purge prompt; this brief is source-inspection-based.