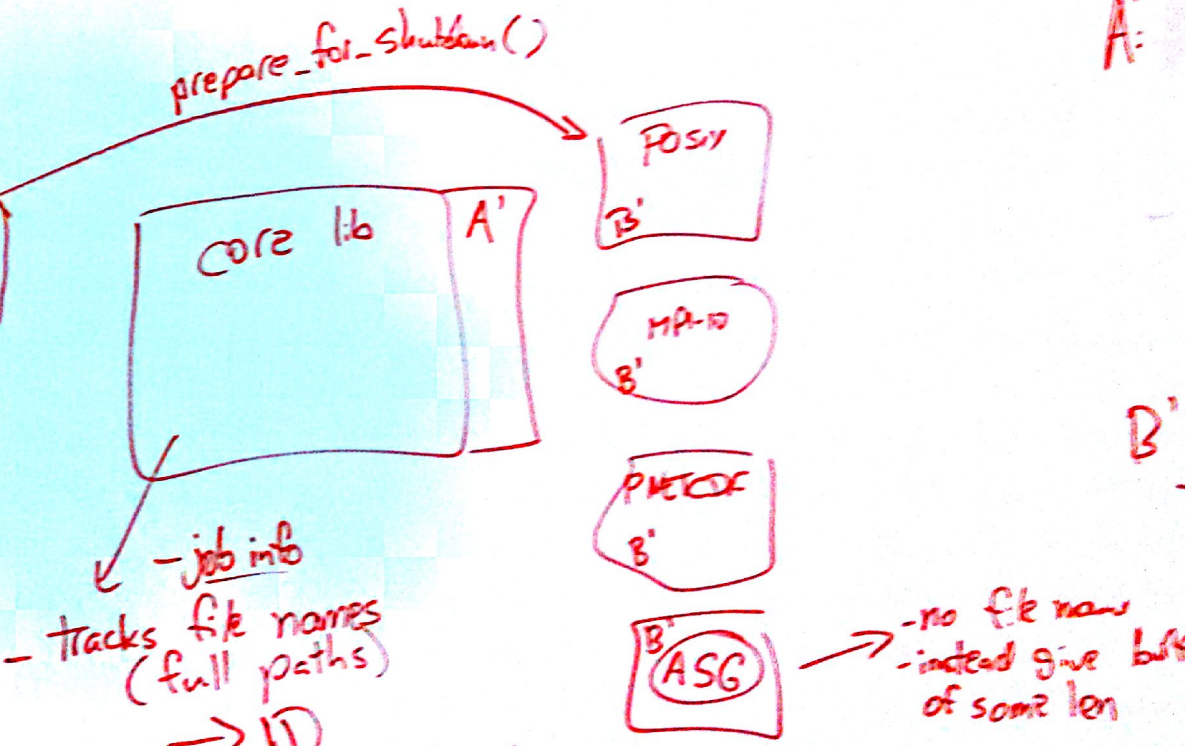
init
- setup data structures

runtime
- tracking names (fds, etc.)
- mem allocation
- counter update
- wrappers

shutdown
- identify shared
- aggregation (reduction)
- compression
- write

Compiler script:
- optionally calls out to files or scripts provided by modules to set link args




- job info
- tracks file names (full paths) → ID
- permits memory allocation (to treat hard bounds)
- kicking off shutdown
- generic (zlib) compression

Common/utility lib
- utility fns for common patterns

Core lib at shutdown:
- spit out filenames → ID mappings
- its own aggregation for shared files

- no file name
- instead give buf of some len

iler script:
itionally calls out to
les or scripts provided
by modules to set
link args

## core lib API

A' - register (char * mod_name, int * runtime_mem_limit, struct mod_fns * mfuncs

- lookup_id (void* name, int len, int64 * ID, int writable_flag)
  - path string or obj ID or ?
  - IDs don't go away once generated

POSIX

MPI-IO

NETCDF

B'

B' ASG → - no file name
- instead give buffer of some len

## B' Module API

- prep_for_shutdown() → triggers aggregation
- get_output (char ** buffer, int size

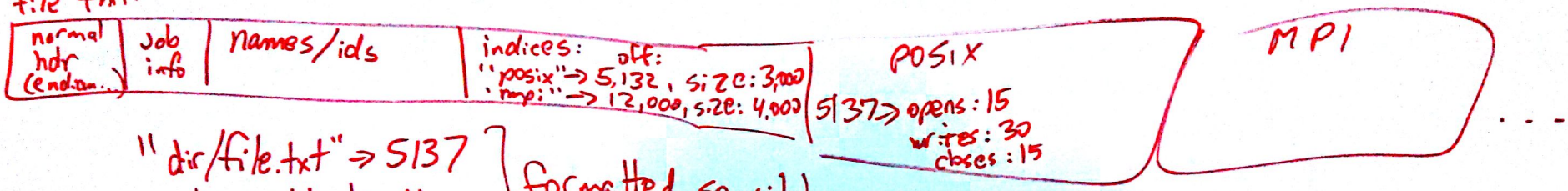
## Common API

- wtime
- detect shared IDs
- ...

ems

# lib util (post processing

file fmt:




"dir/file.txt" -> 5137
↳ variable length

} formatted sensibly + compressed

- each module can define its own parser, grapher, whatever

- devel notes
  - git branch
  - ignore compatibility
  - strip down examples
  - just one or 2 counters to start