Designing a clean-sheet API for tooling/interception in the dynamic linker

Use cases

1. Thread creation and process creation
2. Library loads
3. Memory allocation
4. GPU Kernel Launch
5. Specify load order using JSON or a separate ELF section
6. Context sensitive patching for function calls
7. Redirect library opens
8. Per user/application library caching?
    a. Search hierarchy for caches
    b. Specify particular cache
    c. Change cache semantics of ld.so (choose key to the thing that is loaded)
9. Stacking and composability (eg. Spack + HPCToolkit + Other tool)
10. Library interception
    a. Eg. intercept all calls to all MPI calls
11. Function interception instead of LD_PRELOAD
12. Parameter and return value munging (similar to func intercept)

Use cases ctd

1. Debugging – Ability to prevent loading a library
2. TLS events
3. ABI verification at runtime
    a. Match function signatures
4. ifunc resolution
5.

Notification of dynamic linker events

1. Stackable and composable callbacks
2. Must be read-only interface
3. Event history
    a. Enumerate current state
    b. List of currently running threads
    c. Libraries loaded
    d. State of thread local storage and thread stack
4. TODO:
    a. Someone do a detailed draft of design

Manipulating library load semantics

1. Not read-only interfaces
2. Replace filename
3. Replace a potential file load with an address in memory
4. Provide search algorithm
    a. Replace depth-first search
    b. Do optional caching
    c. Composability
5. Map search paths to new search paths
6. Map library paths to new library paths
7. Map library name to new library name
8. Need an ordering system for tools to manipulate library load ordering
    a. TBD

Function interception and wrapping

1. Semantics of LD_PRELOAD with composability
2. Generalization of Gotcha (by LLNL)
    a. Are Gotcha abstractions are the right ones? (Not the implementation)
3. ifunc resolution

Conclustion

LD_AUDIT + bug fixes + With path interception and rewriting

Gotcha moved into libdl.

Better synchronization for 3rd party debuggers and 1st party tools

More work on fat binaries and interface for program headers needed.