Red Hat
Enterprise Linux

Glibc’s new tool interfaces

the problems they can solve

and where do we go from here.

Ben Woodard <woodard@redhat.com>

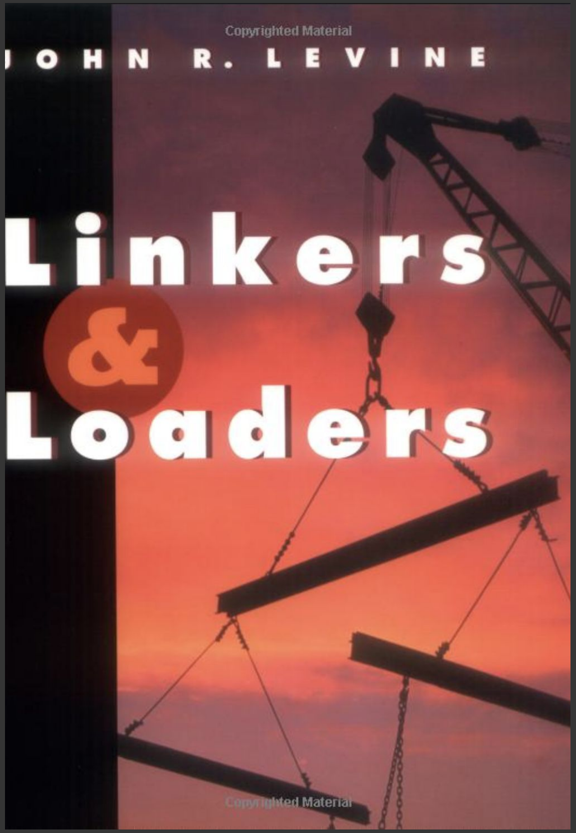
Sr. Principal Consultant

Red Hat

What we'll discuss today

- Dynamic linker problems
- dlmopen() & LD_AUDIT
- Tool support
- Solving HPC problems
- Where do we go from here

Designed in the 90’s not meeting the needs of the 2020’s



Faster loading of large numbers of shared objects

Limitations of LD_PRELOAD for function wrapping.

Impact of search ordering on file servers

RPATH and RUNPATH semantics

Fragility of LD_LIBRARY_PATH

User and project level caching

API and ABI compatibility of shared objects

Linkers and Loaders (The Morgan Kaufmann Series in Software Engineering and Programming) 1st Edition
by John R. Levine (Author)
Publisher : Morgan Kaufmann; 1st edition (October 25, 1999)
ISBN-13 : 978-1558604964

What's new?

dlmopen() and private linkage namespaces

dlmopen() - Required alternative linkage namespaces.

Alternative link maps are on a linked list from the primary.

LD_AUDIT

Reused alternative namespaces from dlmopen()

Incomplete and buggy implementation until glibc 2.35

Fedora 36

RHEL 8.7

Interface of version changed.

DT_AUDIT, DT_DEPAUDIT

ld --audit and --depaudit were not honored until glibc 2.32

Tool support (e.g. gdb, totalview, dyninst)

Underway -Not yet complete.

r_debug.r_version changed and r_debug_extended.r_next pointer.

Man: dlmopen(3)
Man: rtld-audit(7)
Solaris Runtime Linker Auditing Interface https://docs.oracle.com/cd/E19683-01/816-1386/6m7qcoblf/index.htm
/usr/include/link.hl

Solutions to HPC’s problems

Faster loading of shared objects

Spindle https://computing.llnl.gov/projects/spindle

- Pioneered the approach of using LD_AUDIT
- Extremely hard to debug with no debuggers
- Discovered first bugs in glibc’s implementation
  - DT_AUDIT and DT_DEPAUDIT
  - No tool support (i.e. gdb) slowed development
  - Persistent high overhead bug

Limitations of LD_PRELOAD for function wapping

HPCToolkit port in progress

- Started using LD_AUDIT to overcome limitations of LD_PRELOAD
- Many bugs found in glibc's implementation
  - Long and challenging political process to get them fixed.
  - Glibc 2.35
  - Not extremely well tested yet.
- DT_DEPAUDIT for "always on" (LD_NOAUDIT to turn off)
- Private libstdc++ reduces the need for alternative builds.

Impact of search ordering on file servers

Future work: Thundering herd attacks NFS servers with getattrs. Slightly different than spindle in that spindle also addresses transport of the object themselves.

- /etc/ld.so.cache only operates at a system level, no equivalent functionality at the user or project level.
- No directory caching.
- Current behavior can't be changed in glibc
- DT_DEPAUDIT can attach an audit library to an app that:
    - Adds per-user or per-project caching
    - Does file system directory caching

Fragility of LD_LIBRARY_PATH, link order

Future Work: Current heavy use of RPATH is due to poor RUNPATH semantics when applied to HPC enterprises. Underlying problem is a tangle of problems including, no user or project library caching, and the fragility of LD_LIBRARY_PATH

- Very unlikely to change ELF standard
- Evaluate per-user/per-project caching's impact on overall problem
- Use audit to evaluate heuristics or to define better semantics for library searching.
  - e.g. Prefer finding library dependencies in the same directory

ABI aware runtime linking

In development: ABI is ignored at runtime, adding ABI sensitivity through audit, libabigail

- -z, now – basically only detects some API changes
- Requires DWARF
  - Get over size - too many other uses
    - Split DWARF, GNU reflinks, compressed DWARF
    - debuginfod
- Inter-compiler comparisons currently not possible
  - Compiler authors increasingly confident in ABI compatibility
- Very slow. Caching required

What's next

What’s next

- Tool support e.g. gdb
  - Iterating through linkage namespaces done but not merged
  - UI changes needed – disambiguation
- Additional bug fixing
- New features? e.g. Program headers - already backported
- Potential future improvements of audit interface
  - Gestalt view of libraries to be loaded
  - Make lmid accessible
  - Optionally non-ABI registers in plt_enter,exit
  - API to change or finalize GOT entry (like gotcha)

What else do you need?

Proposal: A working group session discussing the needs and desires for tool interfaces going forward?

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat