ld.so in the 2020's

Ben Woodard
Senior Principal Consultant

Red Hat

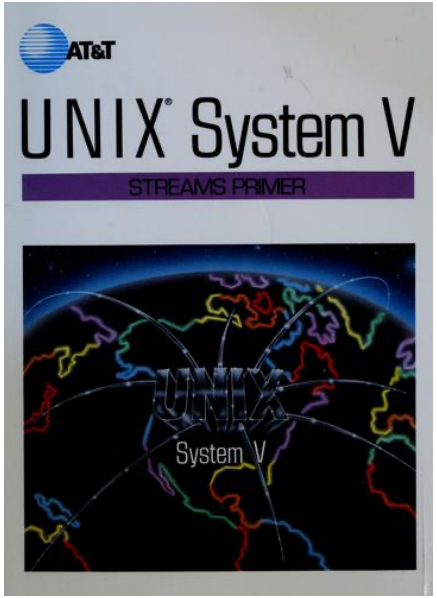
Ben Woodard Senior Principal Consultant Red Hat Inc.

GNU/Linux professionally since 1995

Red Hat since 2001

Onsite Engineer for US DOE/Lawrence Livermore National Lab

(They break everything — all the time. Ergo: Jack-of-all-trades.)



1987

UNIX Goes Commercial

1990's

The UNIX Wars

- Balkanization…but lots of good engineering was done.
- Customer feedback loop running in parallel.

2000's

GNU/Linux begins taking over

- Years of catching up
- Playing by the rules faithfully implementing the standard

2020

GNU/Linux runs the world

🥂🎉👏

Then and Now

1987

- Centrally managed by professional system administrators
- Managed as individual bespoke systems
- Slow release cycle for OS
- Stable software built for the OS
- Binary distribution of software
- Limited software
- Fairly small simple pieces of software with few dependencies

2020

- DevOps
- Containers & “the cloud”
- Your favorite workstation
- Managed in groups by automated tools
- High rate of OS updates
- Vast diversity of software
- Software built from source
- Huge pieces of software with many dependencies

The failings of ld.so

- ld.so.cache
- rpath & runpath
- search order
- ISA subversions
- file servers
- tools interface
- C++’s fragile ABI

ld.so.cache

Nice piece of engineering — for the 90's.

Does anybody really use the cache anymore — everything is in /usr/lib64?

- /opt never really caught on
- There is no sysadmin curating binary software.
- Devops
- The cache doesn't help containers
- Was the point to treat non-root installed software as 2nd class?

Why not?

```
[ben@localhost ~] ld.confg -C .ld.so.cache -f ~/.ld.so.conf
[ben@localhost ~] LD_USER_CACHE=~/.ld.so.cache program
```

Why doesn't ld.so learn?

rpath & runpath

RUNPATH

- RUNPATH seems like it is from a different time when software was centrally managed.
- Assumes app and library developer knows what they are doing — Many times they don't.
- Then a sysadmin or distro maintainer carefully updates and curates the collection

Living in a devops world.

- RPATH is a big hammer.
  Defeats at least a couple of the purposes of dynamic linking
- Big software is fragile
- Everything ends up in a container

But RPATH ends up being overused — because nothing else works.

Why containers and RPATH are overused:

When containers arrived, good software version management stopped. Now nothing is backwards compatible and each package requires specific versions. I run four CUDA and three gcc versions - it stops being fun when they require different driver interfaces 😕 #darksideofcontainers

- Torsten Hoefler

Source:
https://twitter.com/thoefler/status/1295715625550188545

Library versioning

Library versioning doesn’t work in practice

- Developers forget or don’t care - what exactly is a release?
- Package version or library version
- Developers have trouble identifying changes that impact ABI.
- Multiple versions rarely get installed in same dir
- Tooling is cumbersome and not well documented

A library’s version is its ABI which isn’t captured in a number

- Toolchain difference
- Compilation options

Source:
https://www.gnu.org/software/libtool/manual/html_node/Libtool-versioning.html

Library versioning gone wrong

Dependency chain fiasco:

App requires libB, libC

libB requires libD >= 2

libC requires libD >= 3

libD 2 & 3 exist in the search path but 2 is found first

1st person tool's use of libraries must match

tool uses library, app uses same library but different version

Case in point: MPI

MPI

- Standard API
- Fortran ABI compiler and sometimes compiler version dependent
- Regressions & library versioning
- Different vendor implementations
  - OpenMPI vs. mvapich2
  - OPA vs. Mellanox

Workarounds

- Lmod only goes so far
- Parallel builds with each compiler for each OS version with each MPI

Source:
https://lmod.readthedocs.io/en/latest/
https://spack.io
https://openhpc.community

C++ Fragility

Templates are fragile

- GCC 5 dual ABI libstdc++ — https://bit.ly/2EuvRHe
- ABI now or Never — https://bit.ly/3hoxNzA

Library Search:

library search and file servers

Big Apps

- Literally hundreds of libraries
- Un-cached
- NFS
- Between LD_LIBRARY_PATH, RPATH search path hundreds of directories long

Effects on file servers

- Doesn’t use readdir or cache directory contents
- Literally millions of ENOENTs
- Very slow job startup

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

- Very tricky LD_AUDIT library
- Weaknesses in glibc tools interface made it difficult to debug
- Debugging apps nearly impossible due to tools interface weaknesses

ISA Subversions

IFUNCs

- Work at the level of a function
- Inlines don’t work
- Tricky when same source different compilation options

hwcaps

- Not well documented
- Excessive directory searches required - kills file servers

Workaround: archspec https://github.com/archspec

- Outgrowth of spack
- Views ISA as DAG
- Flattens hwcap

Source:
https://sourceware.org/pipermail/libc-alpha/2020-May/113757.html
https://sourceware.org/pipermail/libc-alpha/2020-July/116135.html
https://github.com/spack

Tool interfaces:

Tools Interfaces

Tool interfaces are basically an afterthought and not well maintained

- dlmopen, LD_AUDIT, private namespaces not debuggable — r_debug
- First party interfaces
- library for wrapping functions https://github.com/LLNL/GOTCHA
- A way to embed LD_AUDIT libs

Tool interfaces to other libraries

- example: what's the correct libthread_db - container debugging
  embed the tool interface, register the tool interface with the loader
- OpenMP runtime has the same problem OMPD, OMPT
- C++ parallel runtime does too but it's not solved yet
- Who else needs one?

Source:
https://gbenson.net/?p=407
https://www.openmp.org/spec-html/5.0/openmpch5.html
https://www.openmp.org/spec-html/5.0/openmpsu15.html

Case Study:

Distro building

Making a distribution is fundamentally about ABI.

- A collection of software that maintains an internally coherent ABI through a release.
- Maintaining and updating software while maintaining ABI is a big part of what RH does

This is now a devops containerized world

- Sidesteps the problems
- Misses out on the advantages of dynamic linking

Source:
https://sourceware.org/libabigail/

Conclusion

It's 30 years after SysV, the Unix Wars are over, yes standards are important but we are the standard now. The computing world is changing. Is it time to get back engineering solutions to customer needs?

- Is the rise of DevOps and containerization due to the weaknesses of the loader?
- Are we meeting the needs of software developers now?
  - More robust linking
  - Faster linking
  - Performance tools
  - Debugging tools

Crazy Ideas:

Crazy Ideas

These are more thought experiments for discussion rather than code

- ABI aware loader
- Give users tools that allow them to build and use their own cache

  Automatically learn?
- Tool interface registration in the loader
- Early fork in loader to enable new behavior while preserving old
- Modular library search algorithms
- Modular library requirement solver

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

Red Hat