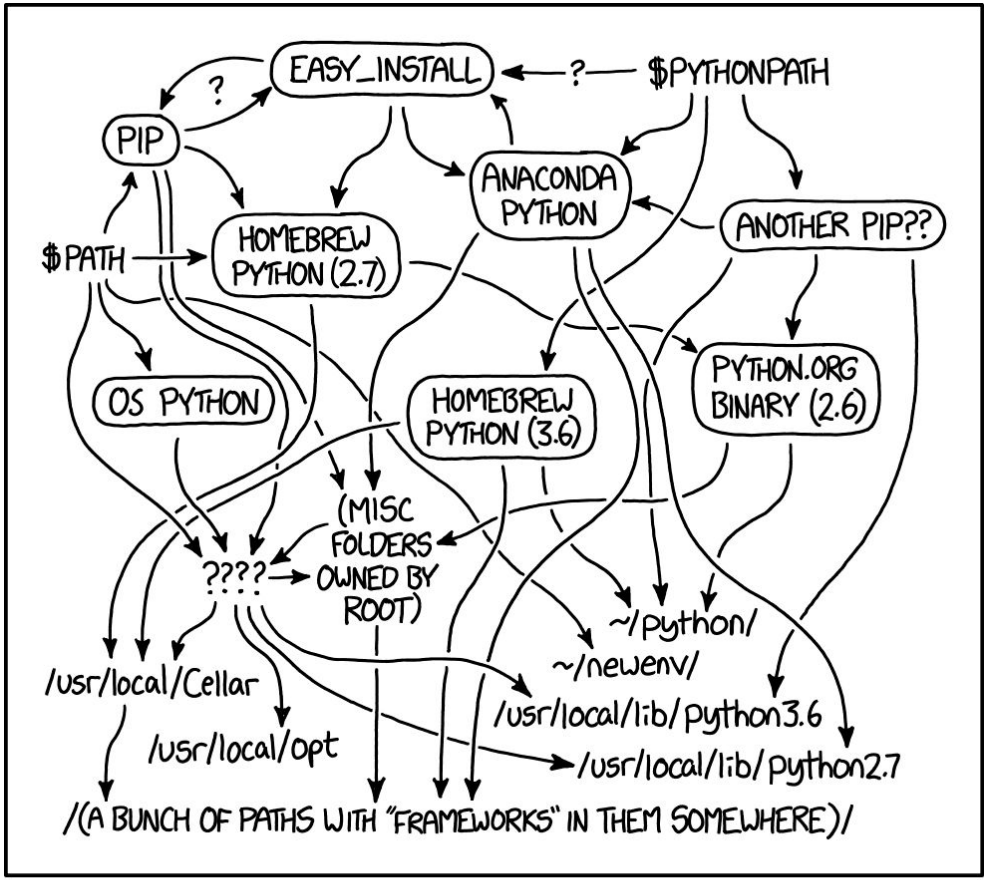
Robust Linux Binaries

How to use Portage to provide a solid base stack for HEP

G. Amadio



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987

Why do we fall into this kind of situation?

- We want software that’s not part of the system
- Use pip, conda, homebrew, etc to get it somewhere
- Not compiled with same compiler → incompatible ABI
- Often requires setting LD_LIBRARY_PATH and/or PYTHONPATH
- Problems if a package is both in the system and in add-ons
- Updates to the system do not take add-ons into account

Classic example: ROOT, Python, and Anaconda

- User has Anaconda installation with Python, wants ROOT
- User then tries to build ROOT with system compiler and link with Python from Anaconda installation
- **libPyROOT.so** has to link against system's **libstdc++.so** and anaconda's **libpython2.7.so**
- Problem: **libpython2.7.so** from Anaconda is not guaranteed to be ABI-compatible with system libraries
- Solution: install GCC from anaconda that was used to build Python and build ROOT with that compiler instead

Why LD_LIBRARY_PATH should be avoided

- **LD_LIBRARY_PATH** is commonly used to add directories to the linker's search path
- Problem: **LD_LIBRARY_PATH** takes precedence, overrides important system libraries
  - See e.g. https://sft.its.cern.ch/jira/browse/SPI-1083
- Solution: use a wrapper script or even better, don't use **LD_LIBRARY_PATH**

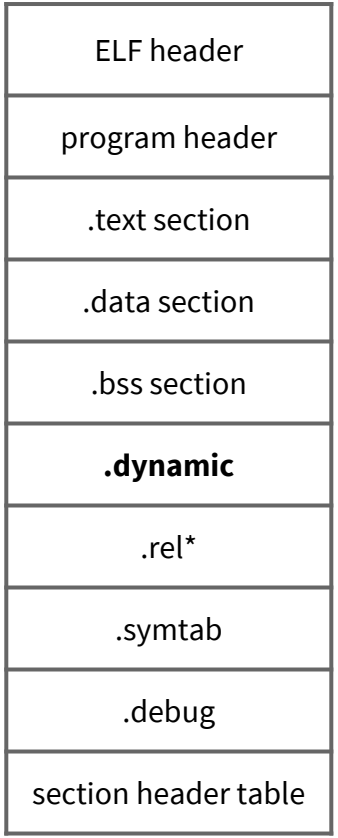
```
$ ssh lxplus7
$ lsb_release -d
Description:    CentOS Linux release 7.5.1804 (Core)
$ source /cvmfs/sft.cern.ch/lcg/views/LCG_latest/x86_64-centos7-gcc7-opt/setup.sh
$ ldd /usr/bin/git
     linux-vdso.so.1 => (0x00007ffdb55b3000)
     libpcre.so.1 => /cvmfs/sft.cern.ch/lcg/views/LCG_latest/x86_64-centos7-gcc7-opt/lib/libpcre.so.1
     libz.so.1 => /cvmfs/sft.cern.ch/lcg/views/LCG_latest/x86_64-centos7-gcc7-opt/lib/libz.so.1
     libpthread.so.0 => /lib64/libpthread.so.0
     libc.so.6 => /lib64/libc.so.6
     /lib64/ld-linux-x86-64.so.2
```

Executable and Linking Format (ELF)

- Linker looks at dynamic section for some tags
- Contains needed libraries, but can contain extra paths where to look for libraries
- **DT_RPATH** and **DT_RUNPATH**
- **DT_RPATH** comes before **LD_LIBRARY_PATH**
- **DT_RUNPATH** comes after
- Both can be changed when installing

```
$ readelf -d $(which bash)

Dynamic section at offset 0xdde08 contains 26 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libtinfo.so.5]
 0x0000000000000001 (NEEDED)             Shared library: [libdl.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000001d (RUNPATH)            Library runpath: [/opt/bash/lib64]
```



https://binvis.io

How to get rid of LD_LIBRARY_PATH

- Add **-Wl,-rpath=${PREFIX}** to your **LDFLAGS** when compiling
- Use --enable-new-dtags (**RUNPATH**) or --disable-new-dtags (**RPATH**) to choose which kind of tag you want
- Better: link with relative **RUNPATH** with **${ORIGIN}** and **${LIB}**
- Not a panacea: if your software depends on environment variables, it will fail when moved to a different location
- Need help from package manager to support relocation

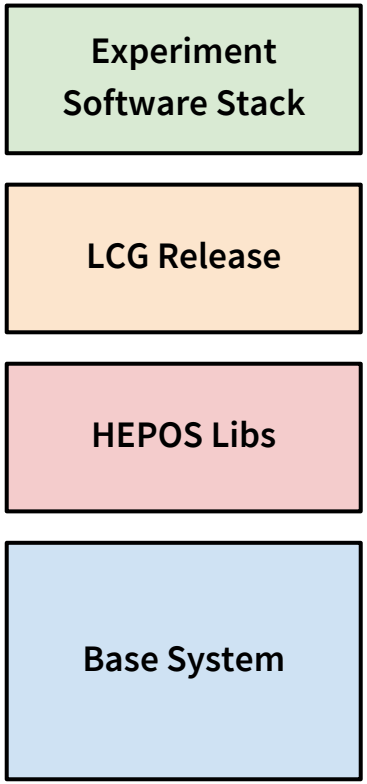
Platform Combinatorics

Currently, HEP softare stacks rely on libraries installed on the base system and on the HEPOS libs meta-package.

Since each Linux distribution uses its own version of libc, binutils, GCC, and includes a different set of system packages, the HEPOS libs meta-package contains different packages for different distros.

For that reason, higher layers must also be adjusted to provide what's missing in the base layers.

Building everything from the ground up would make it possible to reduce platform combinatorics.




```
$ cd /cvmfs/sft.cern.ch/lcg/views
$ ls LCG_latest
x86_64-centos7-clang60-opt
x86_64-centos7-gcc62-dbg
x86_64-centos7-gcc62-opt
x86_64-centos7-gcc7-dbg
x86_64-centos7-gcc7-opt
x86_64-slc6-clang60-opt
x86_64-slc6-gcc62-dbg
x86_64-slc6-gcc62-opt
x86_64-slc6-gcc7-dbg
x86_64-slc6-gcc7-opt
x86_64-ubuntu1604-gcc54-dbg
x86_64-ubuntu1604-gcc54-opt
```

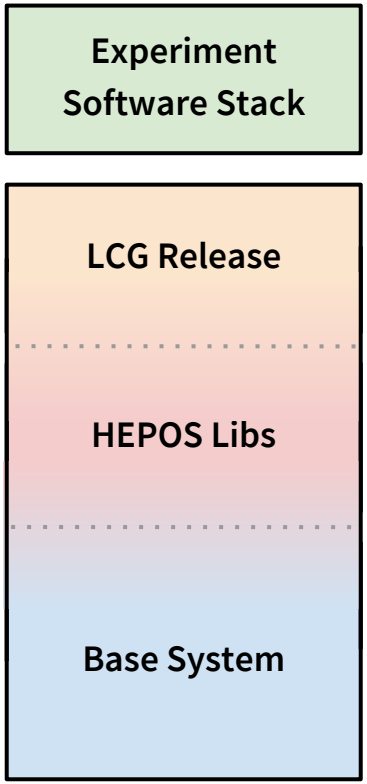
Platform Combinatorics

Currently, HEP softare stacks rely on libraries installed on the base system and on the HEPOS libs meta-package.

Since each Linux distribution uses its own version of libc, binutils, GCC, and includes a different set of system packages, the HEPOS libs meta-package contains different packages for different distros.

For that reason, higher layers must also be adjusted to provide what's missing in the base layers.

Building everything from the ground up would make it possible to reduce platform combinatorics.




```
$ cd /cvmfs/sft.cern.ch/lcg/views
$ ls LCG_latest
x86_64-centos7-clang60-opt
x86_64-centos7-gcc62-dbg
x86_64-centos7-gcc62-opt
x86_64-centos7-gcc7-dbg
x86_64-centos7-gcc7-opt
x86_64-slc6-clang60-opt
x86_64-slc6-gcc62-dbg
x86_64-slc6-gcc62-opt
x86_64-slc6-gcc7-dbg
x86_64-slc6-gcc7-opt
x86_64-ubuntu1604-gcc54-dbg
x86_64-ubuntu1604-gcc54-opt
```

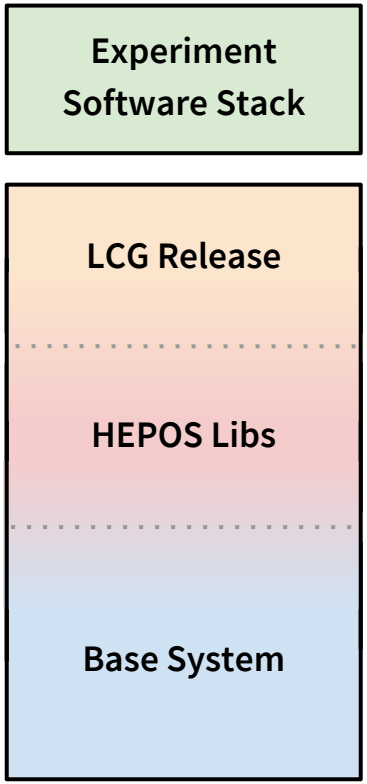
Platform Combinatorics

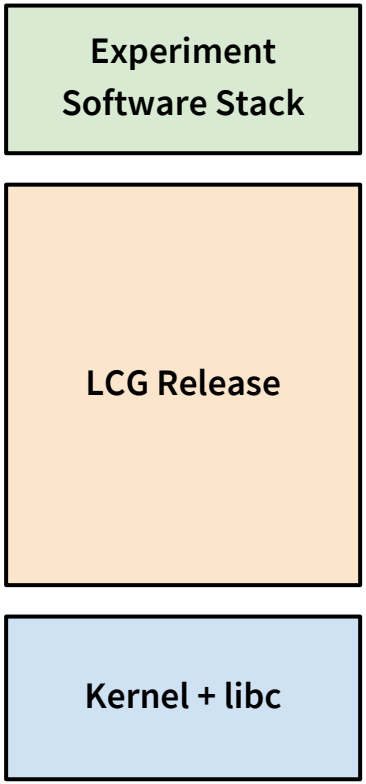
Currently, HEP softare stacks rely on libraries installed on the base system and on the HEPOS libs meta-package.

Since each Linux distribution uses its own version of libc, binutils, GCC, and includes a different set of system packages, the HEPOS libs meta-package contains different packages for different distros.

For that reason, higher layers must also be adjusted to provide what's missing in the base layers.

Building everything from the ground up would make it possible to reduce platform combinatorics.






```
$ ls LCG_latest
x86_64-linux
x86_64-macos-10.12
x86_64-macos-10.13
```

Including HEPOS libs and other packages picked up from the system into the LCG release would allow to reduce the number of platforms, since the Linux kernel promises to never break userspace APIs.

Unfortunately, for macOS (Darwin + Apple libc), breakage is common between releases, and some distros have really old glibc.

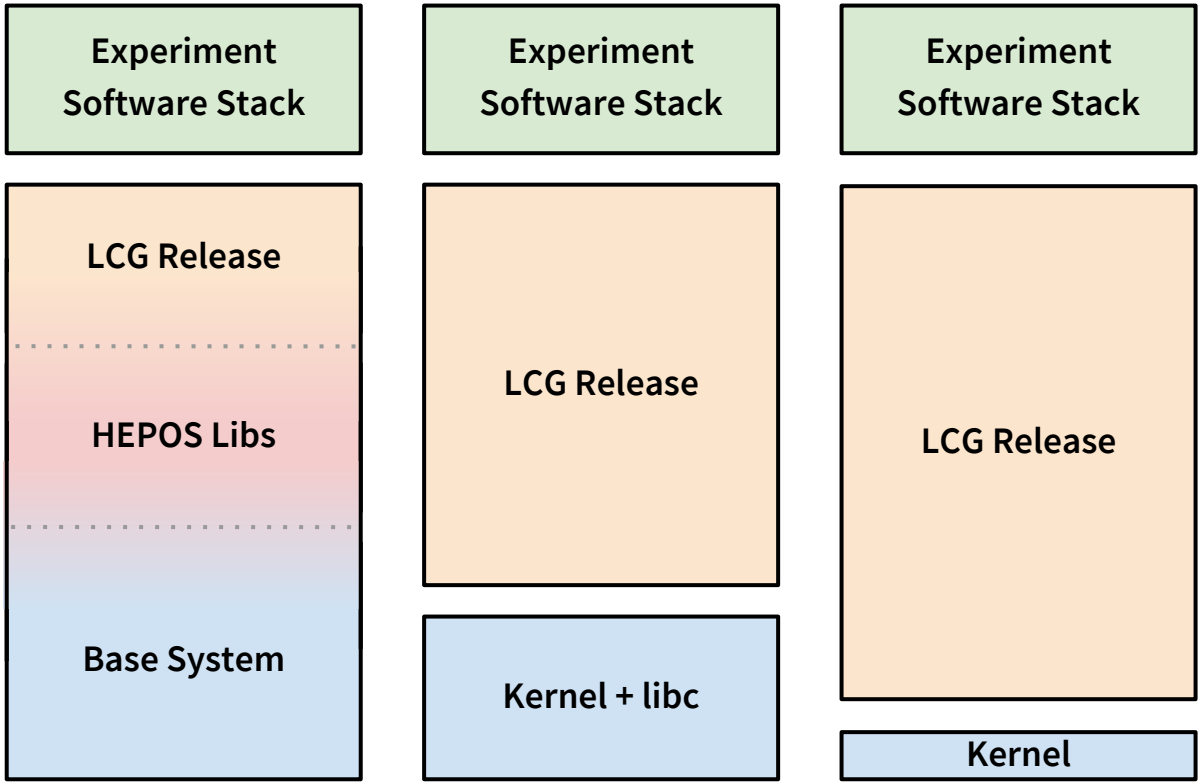
Platform Combinatorics

Currently, HEP softare stacks rely on libraries installed on the base system and on the HEPOS libs meta-package.

Since each Linux distribution uses its own version of libc, binutils, GCC, and includes a different set of system packages, the HEPOS libs meta-package contains different packages for different distros.

For that reason, higher layers must also be adjusted to provide what's missing in the base layers.

Building everything from the ground up would make it possible to reduce platform combinatorics.



Portage Package Manager

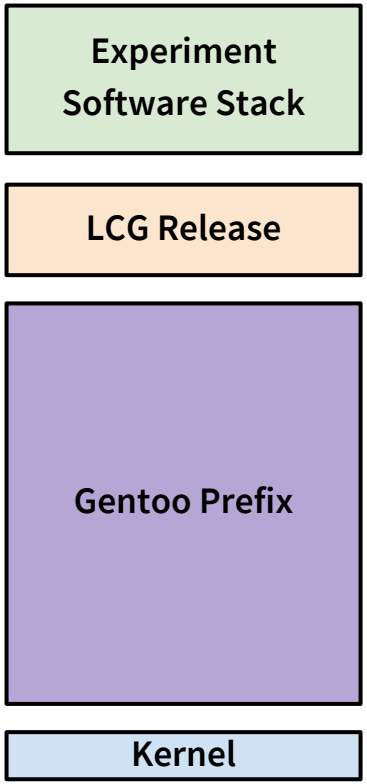
- Official package manager of Gentoo Linux
- Written in Python, based on FreeBSD's ports system
- Packages are special bash shell scripts called **ebuilds**
- Highly flexible configuration/customization
- Parallel and distributed builds (with distcc)
- Easy to support live packaging from git/svn/hg repos
- Able to install to path other than **/usr** with Gentoo Prefix

Why use Portage?

- Leverage work done by others (inherit non-HEP packages)
  - Almost 20,000 packages already available: https://packages.gentoo.org
- Support for multiple platforms and hardware achitectures
- Rebuilds packages only when necessary (i.e. ABI changes)
- Allows to cut completely the dependency on “system packages”
- Single stack works on all Linux distributions
- No software requirements for grid sites (optional requirements on kernel only)
- Can use busybox (1MB) as base container image, use *everything* from CVMFS

A Solid Base for LCG Releases with Gentoo Prefix

- Replace “system” + HEPOS libs with a Gentoo Prefix stack
- Build LCG releases on top of it, same way as currently done
- Cut down on number of stacks (1 Linux + 1 Mac + …)
- Can absorb a lot more software into base stack,
  let LCG and experiments focus on HEP software
- Overcome limitation of 10 year old operating systems
- For more information, please see presentations below
  - Providing an LTS distro with Gentoo Prefix, FOSDEM 2015
  - Unix? Windows? Gentoo! Native Portability to the max!, FOSDEM 2018



Proof of concept stacks available in CVMFS, try it out

```
No setup needed! Just run software by typing full path:

Start ROOT 6.14/00 directly from the prefix (requires kernel 3.2+)
$ /cvmfs/sft.cern.ch/lcg/contrib/gentoo/linux/usr/lib64/root/6.14/bin/root

Start ROOT 6.14/00 directly from the prefix (requires kernel 2.6.32+)
$ /cvmfs/sft.cern.ch/lcg/contrib/gentoo/linux/x86_64/usr/lib64/root/6.14/bin/root

Start a Gentoo prefix shell on Linux (requires kernel 3.2+)
$ /cvmfs/sft.cern.ch/lcg/contrib/gentoo/linux/startprefix

Start a Gentoo prefix shell on MacOS (old stack for 10.12, less packages)
$ /cvmfs/sft.cern.ch/lcg/contrib/gentoo/macos/startprefix

Start a busybox Docker container (on a machine with CVMFS installed)
$ docker run -it -v /cvmfs:/cvmfs busybox \
    /cvmfs/sft.cern.ch/lcg/contrib/gentoo/linux/bin/bash -l
```

Sample of installed software

matplotlib

BULLET PHYSICS LIBRARY

GCC

Qt

R

DASK

IP[y]:

CMake

ROOT
Data Analysis Framework

jupyter

LATEX

CGAL

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Haskell

OpenCV

Java

HDF

protobuf
Protocol Buffers

julia

ERLANG

node JS

scikit learn

Lua

Summary

- Ensure consistency with distributed binaries by using the same compiler to compile your software or use a source distribution
- LD_LIBRARY_PATH can be avoided with RPATH/RUNPATH
- Platform combinatorics can be avoided by building a single stack from the ground up which depends only on the kernel
- Portage can provide such a stack, but it may be possible with other tools as well

Thank you!