A build system for multiple package development utilizing Spack

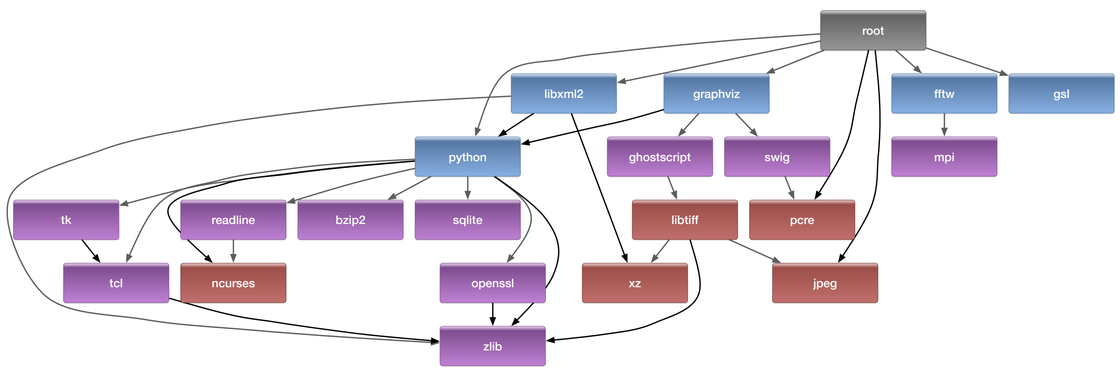
James Amundson and Patrick Gartung, Fermilab

Spack

https://github.com/LLNL/spack

- Spack is a flexible package manager for high performance computing
  - Linux, OSX, *and various supercomputing platforms*
- Has a rigorous model that includes multiple platforms, compilers, versions and variants
- Heavily uses RPATH
  - Relocatability through patchelf (Linux) or install_name_tool (OSX)
- Environment management is factored out
  - Multiple options: modules, dotkit, lmod and ups in development
- Has a community including many developers from multiple fields

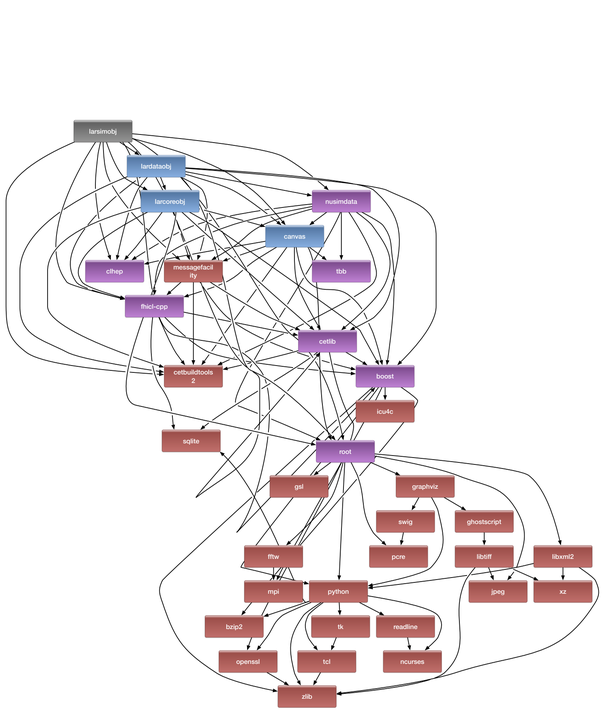
*For more information, see poster "*Towards more common build tools - experience with using spack in HEP"



Spack dependency tree for Root

```
class Root(Package):
    """ROOT is a data analysis framework."""
    homepage = "https://root.cern.ch"
    url      = "https://root.cern.ch/download/root_v6.07.02.source.tar.gz"

    version('6.06.06', '4308449892210c8d36e36924261fea26')
    version('6.06.04', '55a2f98dd4cea79c9c4e32407c2d6d17')
    version('6.06.02', 'e9b8b86838f65b0a78d8d02c66c2ec55')

    if sys.platform == 'darwin':
        patch('math_uint.patch', when='@6.06.02')
        patch('root6-60606-mathmore.patch', when='@6.06.06')

    variant('graphviz', default=False, description='Enable graphviz support')

    depends_on("pcre")
    depends_on("fftw")
    depends_on("graphviz", when="+graphviz")
    depends_on("python")
    depends_on("gsl")
    depends_on("libxml2+python")
    depends_on("jpeg")
    if sys.platform != 'darwin':
        depends_on("libpng")
        depends_on("openssl")
        depends_on("freetype")

    def install(self, spec, prefix):
        build_directory = join_path(self.stage.path, 'spack-build')
        source_directory = self.stage.source_path
        options = [source_directory]
        if '+debug' in spec:
            options.append('-DCMAKE_BUILD_TYPE:STRING=Debug')
        else:
            options.append('-DCMAKE_BUILD_TYPE:STRING=Release')
        options.append('-Dcxx14=on')
        options.append('-Dcocoa=off')
        options.append('-Dbonjour=off')
        options.append('-Dx11=on')
        options.extend(std_cmake_args)
        if sys.platform == 'darwin':
            darwin_options = [
                '-Dcastor=OFF',
                '-Drfio=OFF',
                '-Ddcache=OFF']
            options.extend(darwin_options)
        with working_dir(build_directory, create=True):
            cmake(*options)
            make()
            make("install")

    def setup_dependent_environment(self, spack_env, run_env, dspec):
        spack_env.set('ROOTSYS', self.prefix)
        spack_env.set('ROOT_VERSION', 'v6')
        spack_env.prepend_path('PYTHONPATH', self.prefix.lib)

    def url_for_version(self, version):
        """Handle ROOT's unusual version string."""
        return "https://root.cern.ch/download/root_v%s.source.tar.gz" % version
```

Spack Root package description



Spack dependency the larsimobj package from LArSoft

SpackDev

- SpackDev is a development manager for packages installable with Spack
  - SpackDev uses Spack
  - SpackDev assists in the compilation of one or more dependent packages
  - The packages themselves depend on neither Spack nor SpackDev

SpackDev does what Spack does not

- Spack is designed to build and install
  - Incremental build are not part of the model
- While Spack does allow for hand builds of packages…
  - …hand builds are not necessarily identical to those performed by Spack
  - …no easy way to work on more than one package at once

SpackDev creates a build environment

- Installs external dependencies
- Enables incremental builds of dependent packages
  - Dependencies handled correctly
- Creates a build area that depends only on standard build tools (CMake, Ninja and/or Make)
- Builds exactly as Spack would
  - including Spack's compiler wrappers
- IDE-friendly
  - IDE's such as QtCreator and CLion can be used for development

SpackDev examples

- *spackdev init corge*
  - installs quux and garply, then creates build area for corge
- *spackdev init corge quux*
  - installs garply, then creates build area for corge and quux
- *spackdev init corge garply*
  - creates build area for corge, quux, and garply
    - SpackDev discovers that including quux is necessary for consistent builds, so it automatically adds it to the build area