Worch Build System for LBNE Software

Brett Viren

Physics Department

BROOKHAVEN NATIONAL LABORATORY

LBNE Pre-meeting Workshop 2014/1/31

Outline

What is worch

worch is a system for:

build orchestration worch is a meta-build system for the installation of a suite of software by scheduling the execution of the installation commands that are native to each package (`configure`, `make`, etc).

configuration management all configuration information for the suite is captured in a few text files allowing for release management, reproducible installation. All installation policy/convention is expressed in the configuration.

reliable build automation once started, runs fully unattended unless an error occurs in which case the failure is made clear with log files for postmortem and a generated shell to reproduce the failure in-situ.

The pieces of worch

Worch is made up of these parts:

`waf` a build tool ("`make` on steroids") and an engine for dependency resolution and parallel execution. Implemented in pure-Python. A copy is included with worch.

`orch` a Python module extending waf and providing worch's functionality. Additional, suite-specific modules may also be needed.

configuration simple text files that describe a software suite to install.

When performing an installation the user needs to bring these three parts together.

A Simple Example - cmake + GNU bc & hello

worch provides several example software suites configurations

```
$ git clone https://github.com/brettviren/worch.git
$ cd worch/

$ ./waf --prefix=/tmp/worch-simple-example \   (1)
        --out=tmp                              (2)
        --orch-config=examples/simple/*.cfg \  (3)
          configure                            (4)
$ ./waf                                        (5)
```

(1) Sets the root directory for the final installation.

(2) Sets the directory holding all temporaries

(3) Sets the worch configuration file(s)

(4) The command `configure` is needed whenever the configuration files change

(5) Finally, do the installation. This can be combined with #3 by appending the command “`build`”

A Useful Example: Geant4 + ROOT

```
$ ./waf --prefix=/tmp/g4root \
        --orch-config=examples/g4root/all.cfg \
          configure build
```

- Builds Geant4, ROOT, Python, IPython, XercesC, gccxml, cmake
- Includes Environment Modules (`http://modules.sf.net/`) for end-user environment setup
- Illustrates:
  - the configuration “include” file mechanism
  - extending the basic set of waf tools provided by worch
  - build groups and package dependencies
  - build- and run-time environment settings

Intro to Worch Configuration

As a user, if you modify configuration at all, you will mostly modify an existing one:

- Change a version of a package
- Try to fix a build problem by modifying a configuration flag (for example)

Basics:

```
# this is a comment
[start]
groups = buildtools, all
includes = defaults.cfg, root.cfg, geant.cfg
tools = extras.modulesfile

[group all]
packages = python, ipython, gccxml, root, xercesc, geant
environment = group:buildtools, package:cmake
```

Snippet from g4root suite configuration for Geant4

```
[package geant]
version = 4.9.6.p02
features = tarball, cmake, makemake, modulesfile
source_archive_file = {source_unpacked}.tar.gz
source_url =http://geant4.cern.ch/support/source/{source_archive_file}
source_unpacked = {package}{version}
unpacked_target = CMakeLists.txt
depends = prepare:xercesc_install
prepare_cmd_options = -DG4_ENABLE_CXX11=ON -DBUILD_STATIC_LIBS=ON -
    DGEANT4_USE_OPENGL_X11=ON -DGEANT4_USE_GDML=ON -DXERCESC_ROOT_DIR:
    STRING={xercesc_install_dir}
build_cmd = make
build_cmd_options = -j10
build_target = outputs/library/{kernelname}-g++/libG4track.so
install_target = include/Geant4/G4Track.hh
export_LD_LIBRARY_PATH = prepend:{install_dir}/{libbits}
export_PATH = prepend:{install_dir}/bin
```

Installation Layout - temporary files

All temporary files held under the “out” directory (`tmp/` by default).
Sub-directories:

c4che	waf cache, treat as opaque
urlfiles	files holding the package’s source archive URL
downloads	downloaded source archives
sources	unpacked source directories
builds	native build working directory
logs	log files for each step of a packages installation
controls	worch control files

- The last two are useful for debugging.
- You will not likely need to know about the rest.

Worch logging

Every step makes a log file:

```
tmp/logs/worch_<package>_<step>.log.txt
```

A log file contains various sections labeled with “WORCH” and:

- **CMD** the full command line that was run
- **CWD** the current working directory for the command
- **TSK** the talk name and it’s declared input and output files
- **PKG** internal worch variable settings (many!)
- **ENV** the environment used to execute the command
- **command output** any output produced by the comand

Worch control files

- Waf dependency tree can be expressed as a hierarchy of tasks joined by input/output files.
- Most task “target” (output) files can be declared in the worch configuration
- In addition, all tasks have one conventional output file that worch declares.

`tmp/controls/<package>_<step>`

- Removing a control file (and possibly other intermediate temporaries) will cause the corresponding package step to rerun.
- A “`_<version>`” will soon likely be added.

Overview of full-LBNE Software

LBNE has multiple, partly overlapping software suites:

larsoft	Geant4, ROOT, Art, UPS and most recent GCC
g4lbne	Geant4, ROOT
fast MC	ROOT, Genie, GLoBES
ND	Geant4, ROOT, (Art?)

- Focus has been on building larsoft.
- Suites not needing Art are relatively easy to build.
- Can support both independent software suites and unified ones which leverage each others packages.

Reliance on Fermilab Build Scripts

- The CET/CMake build system for Art and now Larsoft has a build-time dependency on UPS for all external packages.
- Effectively requires us to use the Fermilab build scripts and accept the configurations they provide.

The consequences of this to be aware of:

- We are reliant on a few good but oversubscribed Fermilab experts to provide support for these build scripts.
- At the moment, we are not free to adjust the versions of packages directly by ourselves.
- Not directly related but Fermilab releases are not built in a “green field” and this leads to some problems being found by the installer.

We will discuss this afternoon with CETxperts and ARTists about some possibilities for improvements.

Status of worch-based Larsoft build

Ingredients:

- Larsoft 1.00.01 just released (24 Jan)
  - cetbuildtools/CMake build mechanism (no more SRT)
  - repositories in git (no more SVN/CVS)
- Art 1.08.10
- UPS 5.0.4 (bumped to fix some bugs found during worch install)
- 3rd party packages: CMake, GCC, Boost, ROOT, Geant4, Cry, Genie, Pandora, Xrootd, Python, 3 databases(!)

Status:

- Dependencies build!
- Art builds!
- Nutools builds!
- Larsoft builds!
- ... on SL6.4

Current steps to install Larsoft with worch

Get worch LBNE config files, do a bit of setup and then run waf.

```
$ git clone https://github.com/brettviren/worch.git
$ git clone http://cdcvs.fnal.gov/projects/lbne-software-worch
$ export PYTHONPATH=`pwd`/worch:`pwd`/lbne-software-worch
$ copy worch/wscript .  # copy main waf driver to cwd
$ alias waf=`pwd`/worch/waf
$ waf --version         # trigger an unpack

$ waf --out=tmp --prefix=install \
      --orch-config=lbne-software-worch/config/suite-larsoft.cfg \
        configure build
```

- More verbose than ultimately needed (bundling to be used)
- One known issue (intermittent failure of libxml2 tests)
- Got this far just before the meeting - some bumps may remain.

Current Short-comings and To Do

- **Just** got it built, more testing by me and brave souls needed
- Only SL6 tested, want also Mac and Debian/Ubuntu, others.
- Some things left out for now (`mrb` and `gitflow`).
- Binary UPS “product” tarballs of results not yet being produced.
- Waf/worch bundling to simplify installation instructions.
- Integration of worch to the Fermilab build procedures to be discussed with CETxperts and ARTists this afternoon.

With that said:

A worch-based LArSoft build is coming soon to a CPU near you!