Building and Installing Software

On UD HPC Community Clusters

William Totten
Network & Systems Services
Conventions when Installing Software

- **Installation base directory**
  - /opt/shared
  - /home/work/lab/sw/name/version

- **Keep the source code somewhere safe**
  - /opt/shared/name/attic
  - /home/work/lab/sw/name/attic

- **Software name conventions**
  - Don't change the name unless you have to do so
  - All lowercase letters
  - Use prefixes like "python-", "r-", "perl-", etc. if language specific collisions occur

- **Software version conventions**
  - Use the author's version whenever possible
  - For collections, use the date collected (e.g. 20160414)
Methods of Installing Software

- Prepared Installers
- Simple step-by-step instructions
- Compile with make and GNU Autoconf
- Compile with CMake
- Build java code with ant
- Add on packages for existing software
  - Python
  - Perl
  - R
- Site-specific conventions and wrappers
  - RPM
  - UD’s “udbuild” wrapper on Farber cluster
What is GNU Autoconf

- Autoconf is used by programmers to create a script called “configure”
- It is designed to support variants of C, Fortran, and Erlang
- Configure looks at the system and figures out how it works
- Configure writes a file “config.h” out for programmers to use
- Configure writes a “Makefile” for you to build the software
- You compile and install the software using the program “make”

```bash
$ PREFIX=/home/work/lab/sw/pkg/1.3 ./configure --help | more
...$ ./configure --prefix=$PREFIX checking for a BSD-compatible install... /usr/bin/install -c checking whether build environment is sane... yes ... config.status: creating Makefile config.status: creating config.h config.status: executing depfiles commands config.status: executing libtool commands $ make ...
$ make test ...
$ make install ...
$ 
```
Common GNU Autoconf Options

Command-line parameters

--help
Get a list of the configure script’s options and environment variables

--prefix=/path/to/install
Use this option to specify the directory in which the products will be installed

--enable-feature
Use this option to enable feature or change how the software is compiled

--with-package=[path]
Use this option to select a package to link against, often you can specify the install location of those packages

Environment variables

CC=gcc
Specify the C compiler to use

CFLAGS=-O
Specify options to the C compiler which may be needed
Let’s install HDF5

```bash
$ workgroup --cd --group it_nss
$ cd sw
$ mkdir hdf5; cd hdf5; mkdir attic; cd attic
$ cd ..
$ mkdir 1.8.16; cd 1.8.16
$ PREFIX=$PWD
$ tar xjf ../attic/hdf5-1.8.16.tar.bz2
$ mv hdf5-1.8.16 src
$ cd src
$ ./configure --help
$ vpkg_list szip
$ vpkg_devrequire szip
$ ./configure --prefix=$PREFIX --with-szlib=$SZIP_PREFIX --enable-fortran --enable-cxx
$ make
$ make test
$ make install
$ cd $PREFIX
$ ls *
```
#!/bin/bash -l

PKGNANE=hdf5
VERSION=1.8.16
PREFIX=/home/work/it_nss/sw/hdf5/1.8.16/src

vpkg_devrequire szip

./configure --prefix=$PREFIX --with-szlib=$SZIP_PREFIX --enable-fortran --enable-cxx

make &&
make test &&
make install
What is CMake

- CMake is used by developers to manage the software build process natively on UNIX like and Windows operating systems.
- It is designed to extensible beyond C and Fortran.
- CMake likes to work out of a private build directory.
- CMake writes a “Makefile” for you to build the software.
- You compile and install the software using the program “make.”
- CMake has a menu system to help select build parameters "ccmake".

```
$ PREFIX=/home/work/lab/sw/pkg/1.3
$ mkdir build; cd build
$ cmake -DCMAKE_INSTALL_PREFIX=$PREFIX ..
-- The C compiler identification is GNU 4.4.7
-- The CXX compiler identification is GNU 4.4.7
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Configuring done
-- Generating done
-- Build files have been written to: /home/work/lab/

$ make

$ make test

$ make install
```

Common CMake Options

Command-line parameters

- **-G “Generator”**  Specify a pre-configured environment generator if available
- **-DPARAM=value**  Use this option to define build parameters
- **-DCMAKE_INSTALL_PREFIX=...**  Specify the installation directory

Environment variables

- **CC=gcc**  Specify the C compiler to use
- **CFLAGS='-O'**  Specify options to the C compiler which may be needed
- **PKG_INSTALL=/path**  Convention to specify install location of PKG (different packages use different conventions here, PKG_HOME, PKG_ROOT, ...)
Let’s install HDF5

$ workgroup --cd --group it_nss
$ cd sw
$ mkdir hdf5; cd hdf5; mkdir attic; cd attic
$ cd ..
$ mkdir 1.8.16; cd 1.8.16
$ PREFIX=$PWD
$ tar xjf ../attic/hdf5-1.8.16.tar.bz2
$ mv hdf5-1.8.16 src
$ cd src
$ less CMakeLists.txt
$ vpkg_list cmake
$ vpkg_list szip
$ vpkg_devrequire cmake/3.3 szip
$ export SZIP_INSTALL=$SZIP_PREFIX
$ mkdir build; cd build
$ cmake -DCMAKE_INSTALL_PREFIX=$PREFIX -DHDF5_ENABLE_SZIP_SUPPORT=1 -DHDF5_ENABLE_SZIP_ENCODING=1 -DHDF5_BUILD_FORTRAN=1 -DHDF5_BUILD_CPP_LIB=1 ..
$ make
$ make test
$ make install
$ cd $PREFIX
$ ls *

You can use:
ccmake ..
Here to set these via a menu system
HDF5 setup.sh script

/home/work/it_nss/sw/hdf5/1.8.16/src/setup.sh

#!/bin/bash -l

PKGNAME=hdf5
VERSION=1.8.16
PREFIX=/home/work/it_nss/sw/$PKGNAME/$VERSION

vpkg_devrequire cmake/3.3 szip

export SZIP_INSTALL=$SZIP_PREFIX

mkdir build
cd build

cmake -DCMAKE_INSTALL_PREFIX=$PREFIX -DHDF5_ENABLE_SZIP_SUPPORT=1 -DHDF5_ENABLE_SZIP_ENCODING=1 -DHDF5_BUILD_FORTRAN=1 -DHDF5_BUILD_CPP_LIB=1

make && make test && make install
Python developed its own method for building python modules
- Some python modules are pure-python and are very easy to build, others have portions in compiled languages like C
- You can manually build using the standard "setup.py" file
- Additional tools like easy_install and pip can simplify the build process even more
- pip and easy_install sometimes force you to upgrade other packages
Common setup.py Options

Command-line parameters

- **-O2**
  Create optimized/compiled files ".pyo"

--prefix=/path
  Specify the install location

Environment variables

**CC=gcc**
  Specify the C compiler to use

**CFLAGS=\'-O\'**
  Specify options to the C compiler which may be needed

**PYTHONPATH=/path/.../site-package**
  It is very important to the python installer that it can load the package after installation. Make sure this is set.
Let’s install H5PY

```bash
$ workgroup --cd --group it_nss
$ cd sw
$ vpkg_require python python-six python-numpy python-cython python-setuptools
$ which python
$ mkdir -p python/add-ons/python2.7.8; cd python/add-ons/python2.7.8
$ mkdir h5py; cd h5py; mkdir attic; cd attic
$ wget https://pypi.python.org/packages/source/h/h5py/h5py-2.5.0.tar.gz
$ cd ..
$ mkdir 2.5.0; cd 2.5.0
$ PREFIX=$PWD
$ tar xzf ../attic/h5py-2.5.0.tar.gz
$ mv h5py-2.5.0 src
$ cd src
$ vpkg_devrequire szip
$ export LDFLAGS="-L/home/work/it_nss/sw/hdf5/1.8.16/lib $LDFLAGS"
$ export LD_LIBRARY_PATH="/home/work/it_nss/sw/hdf5/1.8.16/lib:$LD_LIBRARY_PATH"
$ export CPATH="/home/work/it_nss/sw/hdf5/1.8.16/include:$CPATH"
$ PYTHONPATH="$PREFIX/lib/python2.7/site-packages:$PYTHONPATH"
$ python setup.py build
$ python setup.py install -O2 --prefix=$PREFIX
$ cd $PREFIX
$ python -c 'import h5py'
```
#!/bin/bash -l

PKGNAME=h5py
VERSION=2.5.0
HDF5=/home/work/it_nss/sw/hdf5/1.8.16

vpkg_devrequire szip
vpkg_devrequire python python-{six,numpy,cython,setuptools}

PREFIX=/home/work/it_nss/sw/python/add-ons/python$PYTHON_VERSION_LONG/$PKGNAME/$VERSION

export LDFLAGS="-L$HDF5/lib $LDFLAGS"
export LD_LIBRARY_PATH="$HDF5/lib:$LD_LIBRARY_PATH"
export CPATH="$HDF5/include:$CPATH"

PYTHONPATH="$PREFIX/lib/python2.7/site-packages:$PYTHONPATH"

python setup.py build
python setup.py install -O2 --prefix=$PREFIX
Installing h5py with pip

```
$ workgroup --cd --group it_nss
$ cd sw
$ vpkg_require python python-six python-numpy python-cython python-setuptools python-pip
$ which python
$ mkdir -p python/add-ons/python2.7.8; cd python/add-ons/python2.7.8
$ mkdir h5py; cd h5py
$ mkdir 2.5.0; cd 2.5.0
$ PREFIX=$PWD
$ vpkg_devrequire szip
$ export LDFLAGS="-L/home/work/it_nss/sw/hdf5/1.8.16/lib $LDFLAGS"
$ export LD_LIBRARY_PATH="/home/work/it_nss/sw/hdf5/1.8.16/lib:$LD_LIBRARY_PATH"
$ export CPATH="/home/work/it_nss/sw/hdf5/1.8.16/include:$CPATH"
$ pip install --compile --prefix="$PREFIX" h5py
$ python -c 'import h5py'
```
What is udbuild

- A small set shell functions to make writing install scripts easier
- It is designed to help when the same package must be installed with different dependencies, compilers, configurations, or versions
- It only exists on the Farber HPC Community Cluster
- Using udbuild is not required
- It automatically sets your PREFIX variable for you
- It helps with package naming standards
- Environment variables can be conditionally updated based on valet packages or the current package version
- Helps monitor what is updated upon installation
udbuild Requirements and Conventions

Variables  *(set prior to init_uubuildenv)*

**PKGNAME**  This should be an all lowercase version of the package name and will be the first part of the install directory

**VERSION** The version of the package to be installed, this is used by the "version" function described later, and is the second part of the install directory

**UDBUILD_HOME** This should be set to the software install home for your workgroup, by default it would try to install to /opt/shared, and you don't have access to do that

Conventions

**vpkg_devrequire** Call the "devrequire" valet command to load udbuild and any packages needed to install the software which should not be part of the install directory name

**init_uubuildenv** Any package which should be part of the install directory name should be added to your init_uubuildenv line, they will loaded with "vpkg_devrequire" for you
## udbuild Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>init_udbuildenv</code></td>
<td>Initialize the udbuild environment, set the PREFIX environment variable, set auto export, set standard compile variables, and set exit-on-error</td>
</tr>
<tr>
<td><code>udbuildcapture</code></td>
<td>Store all of the output from the build in a file called &quot;udbuildcap.log&quot; for later review, this file is appended to if run multiple times</td>
</tr>
<tr>
<td><code>udbuildmon</code></td>
<td>Store all the files opened for write in a file called &quot;udbuildmon.log&quot; using a system program called &quot;strace&quot;</td>
</tr>
<tr>
<td><code>valet</code></td>
<td>Test if a valet package is loaded into the current environment</td>
</tr>
<tr>
<td><code>version</code></td>
<td>Test if the version of software currently being installed is at a particular level</td>
</tr>
<tr>
<td><code>ifvalet</code></td>
<td>A quick way to call &quot;if valet <code>pkg</code>; then <code>command</code>; fi&quot;</td>
</tr>
<tr>
<td><code>ifversion</code></td>
<td>A quick way to call &quot;if version <code>number</code>; then <code>command</code>; fi&quot;</td>
</tr>
</tbody>
</table>
#!/bin/bash -l

PKGNAME=hdf5
VERSION=1.8.16
UDBUILD_HOME=/home/work/it_nss/sw

vpkg_devrequire udbuild szip
init_udbuildenv
#init_udbuildenv openmpi/1.10.2-gcc-4.9.3

if valet openmpi; then
    CONFIGURE_FLAGS='--enable-fortran --enable-parallel'
    CFLAGS="-L$OPENMPI_PREFIX/lib -I$OPENMPI_PREFIX/include"
    CC=mpicc CXX=mpicxx FC=mpif77
else
    CONFIGURE_FLAGS='--enable-fortran --enable-cxx'
fi

./configure --prefix=$PREFIX --with-szlib=$SZIP_PREFIX $CONFIGURE_FLAGS
make
udbuildmon make install
H5PY udbuild script

/home/work/it_nss/sw/python/add-ons/python2.7.8/h5py/2.5.0/src/udbuild

```bash
#!/bin/bash -l

PKGNAME=h5py
VERSION=2.5.0
PY_VERS=2.7.8
HDF5=/home/work/it_nss/sw/hdf5/1.8.16

vpkg_devrequire udbuild szip
vpkg_devrequire python/$PY_VERS python-{six,numpy,cython,setuptools}/python$PY_VERS

init_udbuildenv python-addon

export LDFLAGS="-L$HDF5/lib $LDFLAGS"
export LD_LIBRARY_PATH="$HDF5/lib:$LD_LIBRARY_PATH"
export CPATH="$HDF5/include:$CPATH"

python setup.py build
udbuildmon python setup.py install -O2 --prefix=$PREFIX
```
Notes on Installing Software

- Read the author's website first for instructions
- Look for a file with instructions after you unpack the source bundle
  
  INSTALL, INSTALL.txt, README, README.txt, README, README.1ST
- Make sure you have all the dependencies installed first
- Look at the install options before beginning (configure --help, CMakeLists.txt, etc)
- The Internet is your friend, try to find the real error message
  - The author's site
  - Google
  - Stack Overflow
- Use the software's forum or author email links if you are having trouble
Questions and Open Forum