Getting & Building the netCDF-C libraries

NetCDF for New Users
2012

Overview

- This talk will cover getting and building the netCDF-C library and utilities.
- We will focus on building in a Unix-like environment (Linux or Cygwin/MSYS for Windows).
- We will discuss two different build systems, ‘autotools’ and ‘CMake’.

Getting netCDF-C

- Latest Stable release (4.2.1.1):
  - http://www.unidata.ucar.edu/downloads/netcdf/
- Latest Developer Snapshot:

```
$ svn co http://svn.unidata.ucar.edu/repos/netcdf/trunk
```

Supported Build Systems

- netCDF-C can be built using two different build systems:
  - Autotools
  - CMake

Autotools

- Autotools-based build chain:
  - Provides support for Unix, Linux through the use of make-based builds.
  - Typical './configure; make; make install' process.
  - Provides very limited support for Windows (Cygwin & MSYS).

CMake

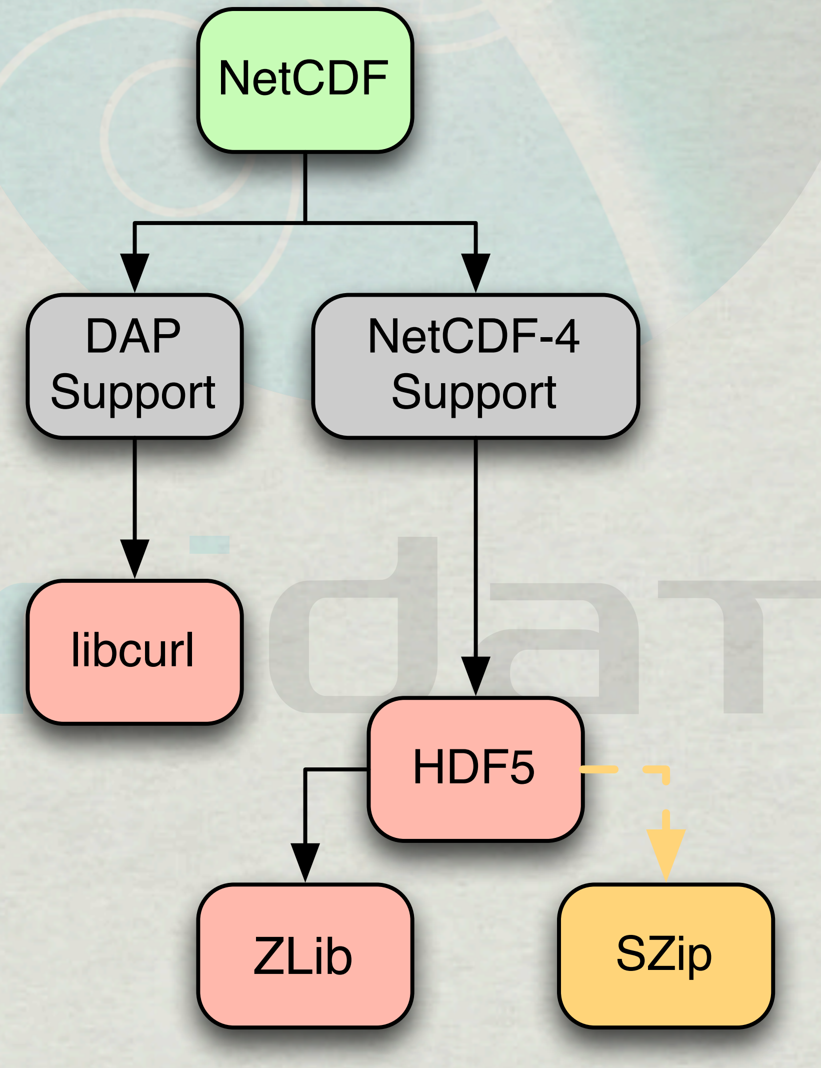
- CMake-based build chain:
  - Provides support for the same systems as the Autotools-based build chain, **plus** Visual Studio builds for windows-native netCDF-C.
  - Provides additional tools for unit and regression testing.

Build Process Overview

1. Configuration: Before compiling, the software is configured based on the desired options.
2. Building: Once configuration is complete, the libraries are compiled.
3. Testing: Post-build, it is possible to run tests to ensure the functionality of the netCDF-C libraries.
4. Installation: If all the tests pass, the libraries can be installed in the location specified during the 'Configuration' step.

Configuration

- Common Configuration Options:
  - netCDF-4 support. This requires that the HDF5 and zlib libraries are installed on the system.
    - If HDF5 was built with SZip support, the szip libraries (included with HDF5) will also need to be linked against.
  - DAP support. This requires that the libcurl libraries are installed on the system.



NetCDF

DAP Support

NetCDF-4 Support

libcurl

HDF5

ZLib

SZip

Configuration

```
$ cd netcdf/
$ ./configure --prefix=[PREFIX]
```

AUTOTOOLS

```
$ cd netcdf/
$ mkdir build
$ cd build/
$ cmake .. -D"CMAKE_INSTALL_PREFIX=[PREFIX]"
```

CMAKE

Configuration

- The autotools-based toolchain creates Makefiles; post-configuration, netCDF-C is built by issuing the ‘make’ command.

- The CMake-based toolchain can create a variety of build types. NetCDF-C is built using the specific tool associated with the build type, or by issuing the ‘cmake --build .’ command.

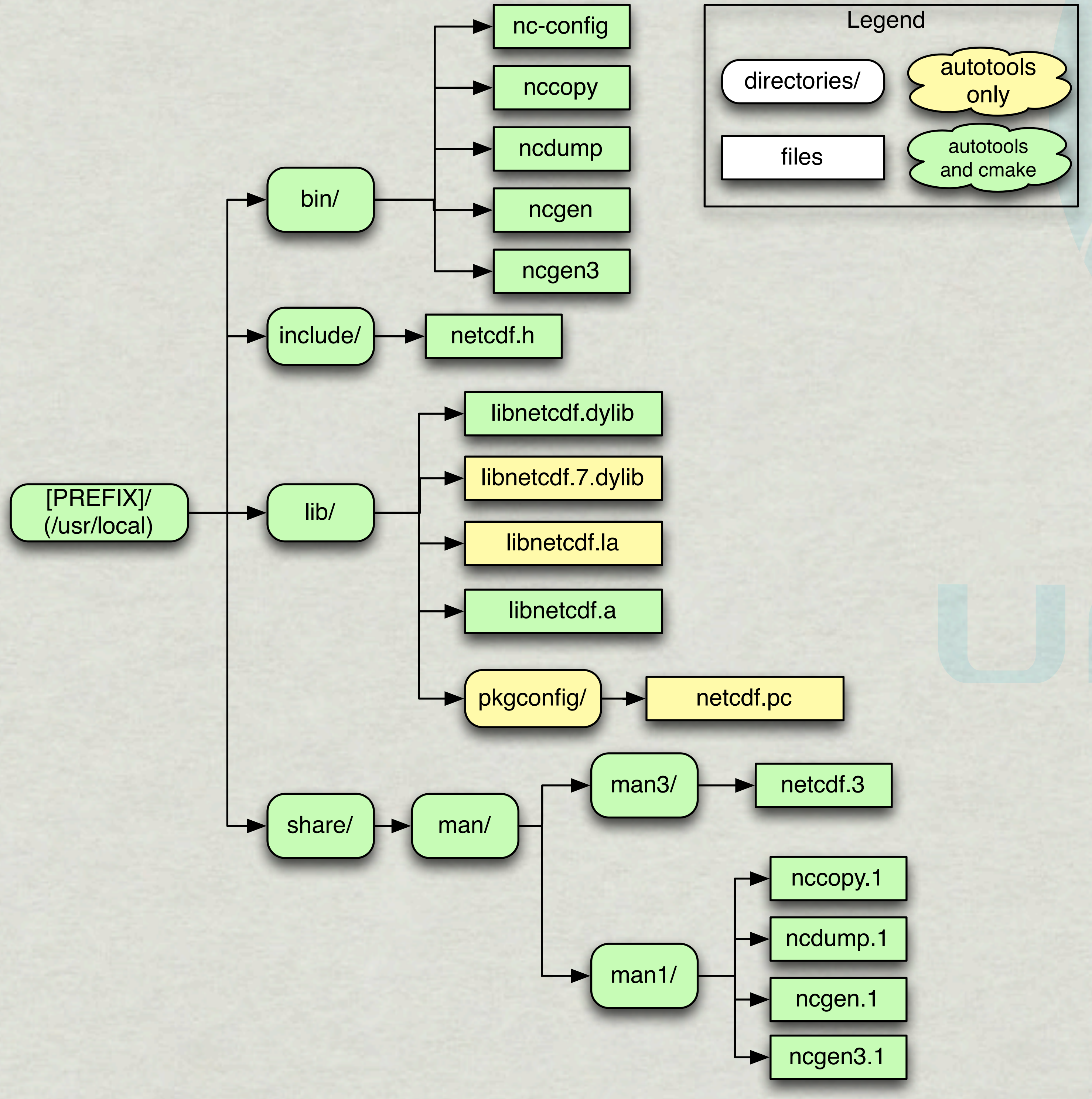
Building

- Build using the ‘make’ command.
- What is generated during the build step?
  - netCDF-C Library
  - netCDF-C Utilities (nccopy, ncdump, ncgen, ncgen3, nc-config).
  - Tests may or may not be built during this step.

Testing

- netCDF-C comes with a number of tests to ensure that the library is functional after compilation.
- Depending on the platform and functionality specified during configuration, different tests will be executed.
- Invoking the tests:
  - Autotools: ‘make check’
  - CMake: ‘make test’

Installing



- Installing is as easy as running ‘make install’ with autotools or ‘cmake --build . --target install’ with CMake. The following files will be installed:
  - The netCDF-C utilities: nccopy, ncdump, ncgen, ncgen3, nc-config
  - The netCDF-C library.
  - The netcdf.h include file.
  - The netcdf pkconfig file.
  - Related man pages.

Non-Standard Dependency Locations

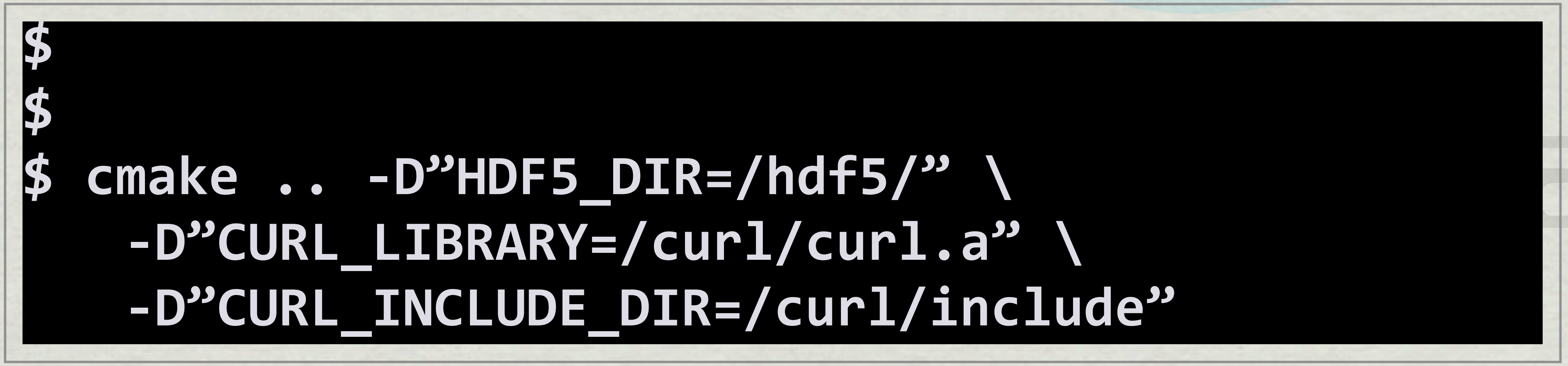
- What if dependencies are in a non-standard location?
  - autotools: Specify using ‘CPPFLAGS’ and ‘LDFLAGS’ when configuring.

```
$ CPPFLAGS=”-I/hdf5/include -I/curl/include” \
  LDFLAGS=”-L/hdf5/lib -L/curl/lib” ./configure
```

AUTOTOOLS EXAMPLE

Non-Standard Dependency Locations

- What if dependencies are in a non-standard location?
    - cmake: Specify using dependency-specific flags when configuring.

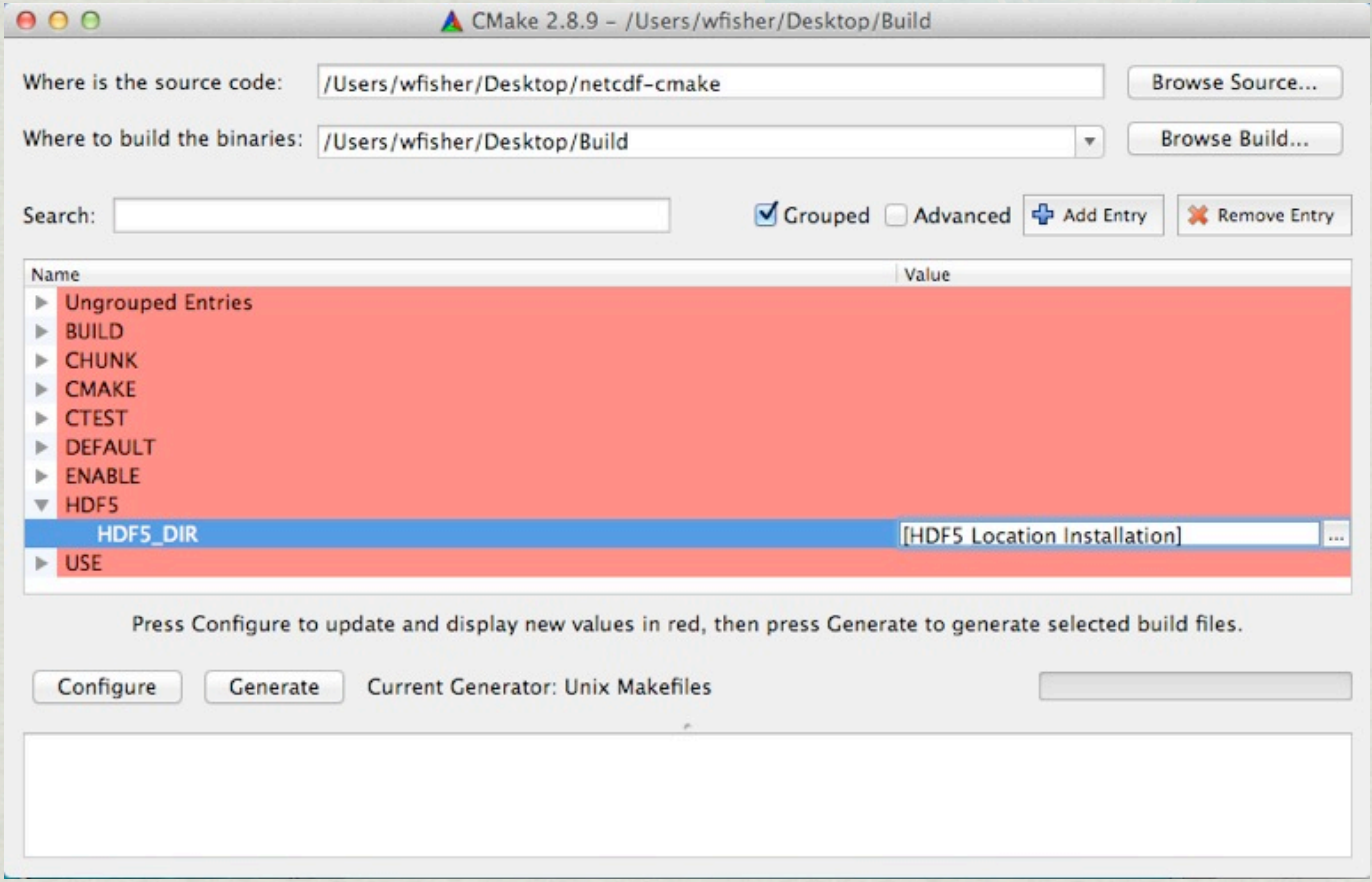


```
$
$
$ cmake .. -D”HDF5_DIR=/hdf5/” \
    -D”CURL_LIBRARY=/curl/curl.a” \
    -D”CURL_INCLUDE_DIR=/curl/include”
```

CMAKE EXAMPLE

- Alternatively, the CMake GUI can be used.

Non-Standard Dependency Locations



Where is the source code: /Users/wfisher/Desktop/netcdf-cmake

Where to build the binaries: /Users/wfisher/Desktop/Build

Search:

☑ Grouped □ Advanced

Name	Value
▶ Ungrouped Entries	
▶ BUILD	
▶ CHUNK	
▶ CMAKE	
▶ CTEST	
▶ DEFAULT	
▶ ENABLE	
▼ HDF5	
HDF5_DIR	[HDF5 Location Installation]
▶ USE	

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles

Summary

- We have discussed
  - Getting netCDF-C library source code.
  - Configuring the source code.
  - Building the netCDF-C library.
  - Checking the build for errors.
  - Installing the libraries.