What is Software Configuration

The IDEAS Scientific Software Productivity Project
ideas-productivity.org/resources/howtos/

Motivation: Installing scientific libraries or applications from source requires a system for setting up (configuring) the package to compile and link the code according to the user's specific platform and needs. This document introduces three of the most common approaches used by scientific libraries and applications.

Method 1: Makefile Options File: The simplest way to communicate the options and machine parameters when building a library or application is to have the person installing the software (henceforth, the installer) directly edit a text file that will be read by the compilation scripts. For example, the installer may edit a file "Make.Incl" (or perhaps even Makefile itself) in the source directory to set the location of compilers and other information needed, such as the location of BLAS and LAPACK libraries. Often this file is read directly by the package's make system and uses standard makefile syntax. An example file may look like the following.

```
# Contents of Make.Incl file
CC=gcc
CFLAGS=-L/usr/local/lib -llapack -lblas
SHARED=0
```

The advantages of this method are the following.

- For small projects it is simple for the developer to maintain.
- It can be used on prototype systems that have a minimal software stack since it only depends on make.

The disadvantages are the following.

- There is no separate configuration step prior to compiling the package during which the options provided by the installer can be tested.
- Thus, compilations will typically be interrupted, as the installer likely will need to repeatedly fix the options that are provided.
- Any errors will be generated by the compiler, which has no knowledge of the meaning of the configuration options. These errors will create error messages that are difficult for the installer to trace back to a specific incorrect configuration option.
- If a package requires specific information about the system, such as the existence of certain mathematical functions or include files, it is necessary for the installer to determine the correct values to set and manually provide the information.

A more progressive method for setting configuration options is through a script that collects from the user (for example, from command line options and environmental variables) and from the system the information necessary to build the library, tests the information to make sure that it is valid, and then utilizes the information to compile and link the software. Such scripts can be more powerful

This material is based upon work supported by the U.S. Department of Energy Office of Science, Advanced Scientific Computing Research and Biological and Environmental Research programs.

than having installers edit a file, but they require more upfront effort to write and require learning a new scripting language.

Method 2: GNU *Autotools* (a.k.a. configure) is the most commonly used configuration system. The installer enters the following.

```
./configure [--prefix] [options]
```

```
make
```

```
make install
```

GNU Autotools includes Autoconf for generating the configure script and Automake and Libtool used by the package developer to simplify providing makefiles and building libraries for the package.

Method 3: *CMake* is a more recent alternative to GNU Autotools utilized by some scientific and mathematical libraries. The installer enters the following.

```
./cmake [options]
```

```
make
```

```
make install
```

One CMake feature (which GNU Autotools does not have) is that it can generate all the data files needed for IDE systems such as *Eclipse* and Microsoft *Visual Studio*.

Several common difficulties occur with GNU Autotools and CMake.

- Software developers need to learn an entirely new language syntax, *M4* or *CMake*. Thus, this script commonly is pieced together from other projects, contains errors, and can be difficult to maintain.
- If the configure or CMake fails, debugging is difficult even for experts.

Advantages of the CMake and GNU Autotools systems over a basic makefile system include:

- Abstracting software installation details from users.
- Determining many machine parameters automatically and providing conventions for setting standard options.
- Automating many parts of the dependency finding and testing process.
- Simplifying the generation of shared libraries.
- Simplifying the management of large complex projects with many source directories and dependencies.

This material is based upon work supported by the U.S. Department of Energy Office of Science, Advanced Scientific Computing Research and Biological and Environmental Research programs.

Other partial configuration systems include the use of third-party utilities to keep track of what libraries have been installed and what options they used (for example, *pkg-config*, or *setup.py*).

This document was prepared by Jason Sarich with key contributions from Roscoe Bartlett, Todd Gamblin and Barry Smith.

This material is based upon work supported by the U.S. Department of Energy Office of Science, Advanced Scientific Computing Research and Biological and Environmental Research programs.

Version 0.2, April 25, 2016