SVN and SCONS
The New CLAS6 Software Control System

D. P. Weygand

"It was long past time for autotools to be replaced, and SCons has won the race to become the build system of choice. Unified builds and extensibility with Python — how can you beat that?" - Eric S. Raymond, author of "The Cathedral and the Bazaar"

SCons *is a fantastic build system, written in Python (1.5.2) that does lots of nice things like automated dependencies, cross platform operation, configuration, and other great stuff. I would have to say that it is probably going to be the best thing for building C/C++ projects in the near future."*

— Zed A. Shaw, Bombyx project lead

"We are using [SCons] on Windows (MSVC and Intel compilers), Linux, IRIX and Mac OS X (gcc and two versions of CodeWarrior). Handles all of those with ease. It can do things like properly handle dependencies on auto-generated source and header files, which would be a nightmare in make."

—SilentTristero (Slashdot user), 10 July 2003 post

What makes SCons better?

- Configuration files are Python scripts--use the power of a real programming language to solve build problems.
- Reliable, automatic dependency analysis built-in for C, C++ and Fortran--no more "make depend" or "make clean" to get all of the dependencies. Dependency analysis is easily extensible through user-defined dependency Scanners for other languages or file types.
- Built-in support for C, C++, D, Java, Fortran, Yacc, Lex, Qt and SWIG, and building TeX and LaTeX documents. Easily extensible through user-defined Builders for other languages or file types.
- Building from central repositories of source code and/or pre-built targets.
- Built-in support for fetching source files from SVN.
- Built-in support for Microsoft Visual Studio .NET and past Visual Studio versions, including generation of .dsp, .dsw, .sln and .vcproj files.
- Reliable detection of build changes using MD5 signatures; optional, configurable support for traditional timestamps.
- Improved support for parallel builds--like make -j but keeps N jobs running simultaneously regardless of directory hierarchy.
- Integrated Autoconf-like support for finding #include files, libraries, functions and typedefs.
- Global view of *all* dependencies--no more multiple build passes or reordering targets to build everything.
- Ability to share built files in a cache to speed up multiple builds--like ccache but for any type of target file, not just C/C++ compilation.
- Designed from the ground up for cross-platform builds, and known to work on Linux, other POSIX systems (including AIX, *BSD systems, HP/UX, IRIX and Solaris), Windows NT, Mac OS X, and OS/2.

SCons
From Wikipedia, the free encyclopedia

SCons

Developer(s)	Steven Knight
Stable release	1.2.0 / December 20, 2008
Operating system	Cross-platform
Type	Software development tools
License	MIT License
Website	www.scons.org

SCons is an open source software build tool. SCons is a substitute for the classic Make utility with integrated functionality similar to autoconf/automake and compiler caches such as ccache. Compared to earlier tools, SCons aims to be easier to use and more reliable.

SCONS Features

Configuration files are Python scripts This provides much more flexibility for solving difficult build problems than traditional build tools.
Reliable, automatic dependency analysis C, C++ and Fortran are scanned for dependencies, eliminating the need for a separate make depend step or a make clean to get all of the dependencies. Avoids the time waste from debugging phantom problems that mysteriously disappear after you make clean; make. Easily extended to scan for other languages or file types.
Built-in support for multiple languages C, C++, D, Java, Fortran, Yacc, Lex, Qt and SWIG. Can also build TeX and LaTeX documents. Easily extended for other languages or file types.
Cross-platform Known to work on Linux, other POSIX systems (AIX, *BSD, HP/UX, IRIX, Solaris), Windows (NT, 2000, XP), Mac OS X, and OS/2.
Fetch files from SCM systems or central directory trees Built-in support for SCCS, RCS, CVS, BitKeeper and Perforce. On-disk directory trees can be searched for source files or pre-built target files.
Reliable detection of file changes using MD5 signatures Use of traditional file timestamps instead of MD5 can be configured.
Parallel builds Keeps up to N jobs running simultaneously regardless of directory hierarchy.
Global view of dependencies Simplifies builds by eliminating multiple passes or re-ording targets to build everything correctly.
Multi-platform configuration (like Autoconf) Support for finding #include files, libraries, functions and typedef declarations.
Shared built-file cache Speeds up multiple builds by allowing developers to share pre-built targets (like ccache, but for any type of target file, not just C/C++ compilation).

Scons and Dependency

Scons has a top notch dependency system

This is one of the reason people go from make to scons. Although make does handle dependency, you have to set up the dependencies in the rules, for example, for a simple object file hello.c which has a header hello.h:

```
1. hello.o : hello.c hello.h
2. $(CC) -c hello.c -o hello.o
```

If you don't set the hello.h, and changes hello.h later, make will not detect it as a change, and will consider hello.o as up to date. This is quickly becoming intractable for large projects, and thus several softwares exist to automatically handle dependency and generate rules for make. Automake (used in most projects using autotools) does this, for example; distutils itself does this, but it is not really reliable. With make files, you have to regenerate the make files every time the dependency changes.

On the contrary, scons does this automatically: if you have #include "hello.h" in your source file, **scons will automatically add hello.h as a dependency to hello.c**. It does though by scanning hello.c content. Even better, scons automatically adds for each target a dependency on the code and commands used to build the target; concretely, if you build some C code, and the compiler changes, scons detects it.

Subversion (software)
From Wikipedia, the free encyclopedia

Subversion

Developer(s)	**CollabNet**
Initial release	**October 20, 2000**
Stable release	**1.6.5 (2009-08-21)**
Written in	**C**
Operating system	**Cross-platform**
Type	**Revision control**
License	**Apache License**
Website	**http://subversion.tigris.org/**

Subversion (**SVN**) is a version control system initiated in 2000 by CollabNet Inc. It is used to maintain current and historical versions of files such as source code, web pages, and documentation. Its goal is to be a mostly-compatible successor to the widely used Concurrent Versions System (CVS).

Subversion is well-known in the open source community and is used on many open source projects, including Apache Software Foundation, Free Pascal, FreeBSD, GCC, Python, Django, Ruby, Mono, SourceForge.net, ExtJS, Tigris.org, and PHP. Google Code also provides Subversion hosting for their open source projects. BountySource systems use it exclusively. Codeplex offers access to both Subversion as well as other types of clients.

Subversion is also being adopted in the corporate world. In a 2007 report by Forrester Research, Subversion was recognized as the sole leader in the Standalone Software Configuration Management (SCM) category and a strong performer in the Software Configuration and Change Management (SCCM) category.[1]

Subversion is released under the Apache License, making it open source.

svn: directory based, while cvs is file based
when moving files and directories in svn the history is preserved!

Getting the ball rolling...

CLAS Offline Software CVS to SVN Transition
In the process of preparing for the 12 GeV upgrade, members of the CLAS offline software group have concluded that the current CVS repository is no longer sustainable. It lacks documentation, contains dozens of obsolete and orphaned packages, and has become increasingly difficult to maintain over the past 20 years. In addition, the build system is equally cluttered and difficult to use, especially for students and new users.

In order to continue the process, the former CVS repository needs to be retired. I've set a time limit of 3 weeks to convert the current CVS repository to read-only. Of course this may create some hardship, but we can no longer keep two versions of the software. And of course the CLAS6 offline development team will help users through the transition period.
The date of the transition is set to September 1, 2009. If you experience difficulties, please contact me immediately: weygand@jlab.org
Dennis Weygand

Building your own user_ana

```
setenv WORKSPACE /work/clas/disk9/${USER}/user_ana_scons_test
mkdir -p ${WORKSPACE}
cd ${WORKSPACE}
setenv CLASTRUNK https://jlabsvn.jlab.org/svnroot/clas/trunk
svn co ${CLASTRUNK}/detector/st
svn co ${CLASTRUNK}/reconstruction/user_ana
svn co ${CLASTRUNK}/reconstruction/ana
svn co ${CLASTRUNK}/reconstruction/recsis

source /group/clas/local/current/scripts/environment.csh

cd ${WORKSPACE}/st
scons prefix=../local install -j8
cd ${WORKSPACE}/user_ana
scons prefix=../local install -j8
cd ${WORKSPACE}/ana
scons prefix=../local install -j8
cd ${WORKSPACE}/recsis
scons prefix=../local install -j8

setenv PATH ${WORKSPACE}/local/bin:${PATH}
```

Directory Structure Organization

The new directory structure splits the library packages into understandable categories

Instead of everything in one directory, we now have groups of libraries in directories with names such as:

- Reconstruction
- Calibration
- Detector
- Analysis
- Simulation
- Test
- Pcor
- Scripts