Simulation Software Distribution

Overview

Because of the number and complexity of its external dependencies, the easiest way to build the slic binary is using the *SimDist* CVS project.

Depending on your system speed, the entire build process will take around 2 hours, but starting the automated build only takes about 5 minutes.

Packages in the Distribution

There are nine software packages built by SimDist, only one of which is optional.

These are described in the table below.

Package	Version	Description	Website
CLHEP	2.0.4.7	physics utilities for Geant4	http://proj-clhep.web.cern.ch/proj-clhep/
Geant4	9.4.0	physics simulation toolkit	http://geant4.cern.ch
LCIO	head	ILC data format	http://lcio.desy.de/
HepPDT	3.3.2	extended physics data	https://savannah.cern.ch/projects/heppdt/
Xerces	3.1.1	XML toolkit	http://xerces.apache.org/xerces-c/
GDML	head	geometry XML format	http://gdml.web.cern.ch/GDML/
LCDD	head	detector description XML format	lcdd
SLIC	head	simulation front-end	slic
LCDetectors	head	detector data library (optional)	LCDetectors

Versions are current as of 1/12/2011.

Supported Platforms

The SimDist project requires a Unix-like environment. The CVS head currently builds successfully on Linux (tested on Redhat Enterprise Linux), OSX, and Windows using Cygwin. The recommended compiler version is gcc 4.1 or greater, though depending on your platform, a gcc 3 series compiler might work.

Build Instructions

Required Build Tools

SimDist requires a number of different build tools beyond the standard ones you may have on your system. This is because the project has a lot of external dependencies, each of which has its own build system.

Here is the list of required tools to build SimDist successfully.

- gcc
- make
- cmake (optional)
- java
- wget

If one or more of these is not present, then you or your admin should install it before proceeding with these instructions. If you are unsure of which version to use, just get the most recent version that works on your platform. This should work fine.

> **CMake on Windows**
> CMake does not seem to work well with Windows. Specifically, it does not like paths with spaces in them. To build SimDist successfully using Cygwin, uninstall CMake from your Cygwin installation and make sure that you don't have the CMake binary in your PATH.

Obtaining SimDist

SimDist is a module in the SLAC CVS repository. You can check it out with the following command.

```
cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcd co SimDist
```

This will create a directory called *SimDist* in the directory where this command is run. The next sections will cover building SimDist once you have checked it out.

Basic Build Procedures

If you want to execute each step of the build yourself, these are the commands used by the build script (covered in next section).

```
./configure [options]
make &> build.log &
tail -f build.log
```

The configure command can be modified with options. (see available options with './configure --help')

The last command will tail the log file, so that you can see the progress of your build.

Build Script

There is also a short bash script for running the build.

```
cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcd co SimDist
cd SimDist
chmod +x build.sh
./build.sh
```

This will create the log files **configure.log** and **build.log**, which might be useful if your build fails.

Configure Script

Simdist uses a configure script generated by Autoconf. This script only accepts a few options, as the slic binary has a standard set of external dependencies, none of which are optional.

The **--disable-opengl** option can be used to turn off OpenGL visualization support by Geant4. This may be useful if you only need to run slic in a batch computing environment.

```
cd SimDist
./configure --disable-opengl
```

The performance of the Geant4 package can also be improved if its informational messages are turned off.

```
./configure --enable-g4verbose=no
```

The arguments listed under "Fine tuning" and "Optional Packages" are generally not supported. For instance, you cannot disable individual packages, as they are all required. Also, the "influential environment variables" are generally ignored by SimDist, as each package has a different set of compiler arguments.

Disabling Geant4 Visualization

Sometimes it may be useful to disable Geant4 visualization entirely, as at has many dependencies that may not all be present on your host system. Execute this command from the *SimDist* directory to disable Geant4 visualization.

```
./configure --enable-g4vis-none
```

This should set the variable *G4VIS_NONE* to *1* in the Geant4 environment configuration file at *packages/geant4/env.gmk*.

Testing the Build

Try executing this script when the build completes.

```
./scripts/slic.sh
```

If there is an error running the slic binary using this script, then check your build logs (starting at the top) for error messages.

Starting a Clean Build

To cleanup an existing build and start a new one from scratch, execute these commands in the *SimDist* directory.

```
make distclean
./configure [options]
```

This should remove *all* package binaries and return the project to a clean state. The build can then be executed in the normal fashion.

Creating a Tarball

The slic package includes a target for creating a *.tar.gz* file containing the slic binary and its runtime dependencies.

```
make dist
```

The tarball will be named based on the slic and Geant4 versions and your platform. It will appear one directory below your *SimDist* directory.

Troubleshooting Common Build Problems

The most common problems are related to configuration of Geant4, because it is a large, complex package. And it has no standard configure script. Make sure that the options generated by SimDist at *packages/geant4/env.gmk* make sense for your platform. For instance, if you don't have the OpenGL headers and libraries, then all the GL variables should not have a value. A blank variable means that the corresponding option will not be enabled. These variables are basically passed directly to Geant4's build system. Refer to the Geant4 documentation of these settings for more details.

Commands to the slic package only can be used to debug problems that occur with other dependencies. For instance, if the slic build complains about missing LCIO headers, then the LCIO library build probably failed. Or if there are a bunch of error messages about linking against Geant4, then part of Geant4 likely did not compile correctly.

If all of the above fails to get a working SimDist, then send email to jeremym AT slac DOT stanford DOT edu with the following.

1. OS variety and kernel version e.g. for Linux
   ```
   cat /etc/redhat-release; uname -a
   ```
2. gcc version
   ```
   gcc -v
   ```
3. 32 or 64 bit
   ```
   uname -m
   ```
4. log of global configure command
   ```
   cd SimDist; ./configure &> myconfig.log
   ```

Please attach or inline the snippets containing build errors from your logs. I don't want full log files, if possible. Usually, the first error in the log is the most relevant.

Advanced Build Commands

The SimDist build system is based on GNU Autoconf and Make. Individual packages can be manipulated using make commands from their appropriate package directories.

For instance, to cleanly recompile slic, execute these commands.

```
cd SimDist/packages/slic
make packageclean
make
```

Most packages support the following make targets through a generic packaging system.

```
make              # execute standard build targets in order
make cvs          # checkout from cvs
make packageclean # removes the current build
make download     # download a tarball of the package
make config       # execute configure script or commands
make compile      # compile the libraries or binary
make install      # install the package
```

Each package supports either a *cvs* or a *download* command, depending on whether it is contained in the SLAC cvs repository. External packages are retrieved using the *wget* program.

How SimDist Works

Package System

SimDist operates on a loose system of package configuration, which sets a number of standard build variables in the configuration stage for each package in the *packages* directory. A configure script in the base directory generates the actual Make file fragments for the packages, depending on the configuration selected and on the properties of the local host. Following the Autoconf build system's conventions, there are *.in* files in the package directories that are used by the configure script to generate these Make fragments. The configuration of individual packages is defined in the Autoconf file at *build/packages.m4*. This sets and substitutes the necessary variables for the build system by calling the Autoconf macro associated with each package.

Each package has a file called *package.gmk*, where the default settings for packages are overridden or additional settings can be added.

For instance, sometimes it is useful to enable verbose compilation and linking when building slic (the final package to be built by an in-order build).

To do this, you can manually add an option to its *package.gmk* file in *packages/slic/package.gmk*.

```
PACKAGE_CONFIG_OPTS="--enable-compile-verbose"
```

Other options available in the slic build only can be viewed in the slic directory.

```
cd slic/HEAD
./configure --help
```

For instance, there are options for making gdb-enabled binaries. These could be passed to SimDist by adding this option to slic's *package.gmk* fil e.

```
PACKAGE_CONFIG_OPTS="--enable-gdb=yes"
```

Each package has different possible options for this particular variable. The above examples will not work on other packages with SimDist, only the slic package.

Package Variables

The core of the SimDist build system is in *build/package.gmk* where a standard set of build procedures is defined for a generic software package. These procedures include automation of fetching package source codes, configuration, execution of build commands, and installing binaries and headers.

Each package may have some or all of the following defined within its *package.gmk* file generated by SimDist's global configure script. Most of these variables are given reasonable defaults and do not need to be specifically defined by a given package.

Variable	Meaning
PACKAGE_NAME	short name of package
PACKAGE_BASEDIR	base directory of package sources
PACKAGE_BUILDDIR	build directory
PACKAGE_CONFIG	configure command
PACKAGE_CONFIG_OPTS	extra options supplied to the configure command
PACKAGE_CVSMODULE	name of the package's cvs module
PACKAGE_CVSROOT	the CVSROOT for the package's cvs repository
PACKAGE_DIR	SimDist package directory containing package.gmk and env.gmk
PACKAGE_DIR_ORIG	original directory name, e.g. if downloaded and unzipped from a tarball
PACKAGE_DOWNLOAD	URL of the package tarball
PACKAGE_TGZ_LOCAL	name of the tarball locally
PACKAGE_VERSION	version tag of the package or HEAD to use its cvs head
PACKAGE_DIST_TGZ	name of tarball if distributing this package
PACKAGE_DIST_FILES	list of files to be included in distribution of this package

These variables can be overridden or added by editing the file *package.gmk* within each package's SimDist directory (see above example for slic).

Additionally, each package includes variables necessary for its build in the file *env.gmk*. This should generally not be edited by hand, whereas editing *package.gmk* can sometimes be useful.

Both the *package.gmk* and *env.gmk* files are rewritten when a global configure command is executed from the SimDist base directory. This is by design, following the convention of Autoconf. If you really want changes to stay, the .in files can be edited. But only a few of the variables are likely going to be useful, such as *PACKAGE_CONFIG_OPTS*, for passing options to the package's configure script.

Package Versions

The version of each package is specified in the file *VERSION* within each package's directory. For example, the slic version is at *packages/slic/VERSION*. The format of this file is as follows.

```
[package_name] [package_version]
```

For example, this is used to specify slic's CVS head.

```
slic HEAD
```

Not all packages will support the *HEAD* argument, mostly just SLAC cvs packages like slic, lcdd, gdml, and lcio.

Specific tags should generally be in this format.

```
v[version]r[release]p[patch]
```

For instance, to use slic version 1.2.3 (a made-up example), the version file should contain this.

```
v1r2p3
```

SimDist can usually figure out how to manipulate these version tags into the format required for accessing the package's sources via cvs or download. The only exception to this currently is the HepPDT package. You should not change the version of this package, anyways. For other

packages, it may be useful to override the defaults. For instance, selecting a different Geant4 version might be necessary. You can do this by editing the Geant4 version file at *packages/geant4/VERSION*. The Geant4 version string may also have a *b* instead of a *p* to designate beta versions.