

# **Geant4 Installation Guide: For setting up Geant4 in your computing environment**

**Geant4 Collaboration**

---

# **Geant4 Installation Guide: For setting up Geant4 in your computing environment**

by Geant4 Collaboration

---

---

---

# Table of Contents

1. Installation Introduction .....	1
1.1. Supported Computers and Operating Systems .....	1
1.2. Required Software .....	1
1.3. Visualization Software .....	1
1.4. Software for Analysis and Histogramming .....	2
2. Installation Procedures .....	3
2.1. Using the <code>Configure</code> Script for installation .....	3
2.2. Configuring the Environment to Use Geant4 .....	3
2.3. Installing Geant4 Manually .....	4
2.3.1. Required Environment Variables .....	4
2.3.2. Optional Environment Variables .....	4
2.4. Integrating Geant4 into a Generic Framework .....	5
3. Tips for Installing on Windows .....	7
3.1. Windows with the Cygwin Toolset and Microsoft Visual C++ .....	7
3.2. Building Kernel Libraries DLLs .....	8

---

# Chapter 1. Installation Introduction

This section describes the global computing environment required for installing the Geant4 toolkit. To set up your specific computing environment for Geant4, refer to Section 2 of this Guide.

A shell script (`Configure`) is provided to facilitate the installation procedure, and to help the user set up the environment correctly.

## 1.1. Supported Computers and Operating Systems

Geant4 is supported under the following operating systems:

- Linux on PC with g++ (gcc compiler)
- Flavors of Unix (from vendors: SUN)
- Windows/XP with MicroSoft Visual C++

Currently, this is the set of flavors which can be associated with the environment variable `$G4SYSTEM` to identify the system architecture and compiler used:

```
Linux - Scientific Linux CERN, SLC3 or SLC4 g++ gcc 3.2.3 or 3.4.5 G4SYSTEM:
Linux-g++ UNIX - SUN-SunOS v.5.8, CC Forte Developer 7 C++ 5.4 Patch 111715-05
G4SYSTEM: SUN-CC Windows - Windows XP and Cygwin32, MSVC++ 7.1 .NET G4SYSTEM:
WIN32-VC
```

For a more detailed description of supported platforms/compilers and versions of third party software, please refer to the release notes available with the current release you're using in `geant4/ReleaseNotes` (also accessible from the web distribution page ).

## 1.2. Required Software

To run Geant4, the following software must be properly installed in your computing environment:

- C++ compiler  
(compiler from Unix vendor, g++ or Visual C++ for Windows systems)
- CLHEP library  
(see CLHEP reference guide [cern.ch/clhep/manual/RefGuide](http://cern.ch/clhep/manual/RefGuide) )
- GNU Make (note: g++ preprocessing is used to build file dependencies) is also used and a UNIX shell
- The Geant4 toolkit source code

## 1.3. Visualization Software

The above list of software is the minimum required for a non-graphical setup of Geant4. To use visualization graphics in Geant4 and/or a graphical user interface (GUI), at least one of the following graphic systems or packages is required in your computing environment:

- X Windows
- OpenGL or MesaGL (free software compatible with OpenGL)
- DAWN , a PostScript renderer
- Open Inventor (free software from SGI)
- Open Scientist (Interactive environment, including GUI)
- WIRED3 HepRep Browser ( HepRep Browser)
- WIRED4 JAS Plug-In ( HepRep Browser)
- Momo (a Java -based GUI environment, GGE, GPE, ...)
- VRML browser

Alternatively, you can produce an ascii file for VRML or DAWN.

More information is available in Section 8.6, Visualization Drivers, of the User's Guide for Application Developers.

## 1.4. Software for Analysis and Histogramming

Histogramming facilities are provided through the **AIDA** abstract interface, which in this case needs to be installed as well:

- AIDA (Abstract Interfaces for Data Analysis)

External, AIDA-compliant packages which provide the necessary functionalities for doing histogramming (and therefore, should be optionally installed) are:

- JAS (Java Analysis Studio)
- PI (AIDA Interactive Analysis Environment)
- Open Scientist (Interactive Analysis Environment).

---

## Chapter 2. Installation Procedures

Before installing Geant4, the required software listed in Section 1.2 (and Section 1.3 in the case of graphics drivers) of this Installation Guide must already be installed on your system.

In this section, a short tutorial on how to install the toolkit's kernel libraries is given. The installation of the Geant4 kernel libraries and the proper configuration of the environment can be achieved either manually (by setting the proper environment variables) or through the `Configure` shell script, which will allow the installation of just the necessary source code and libraries in a specified installation area.

### 2.1. Using the `Configure` Script for installation

A shell script is provided for building the libraries and to allow easy installation in a specified area. The `Configure` shell script is placed in the top directory tree of the distribution (`geant4/Configure`) and allows the user or system administrator to install the Geant4 toolkit in a semi-automatic way. Some knowledge of the system is required for the installation, such as:

- the compiler to be used
- the path where the Geant4 toolkit is to be installed (`$G4INSTALL`)
- definition of installation directory paths (optional)
- kind of graphics/analysis system(s) installed in the system, and paths to the installation of each graphics/analysis package
- the kind of library to be generated: static/shared, compound/granular.

To run the installer script for installation of the Geant4 toolkit, one must type the following from the top directory `geant4`:

```
> ./Configure -build
```

and follow the on-screen instructions. The script will ask for the path in the system where the Geant4 libraries should be installed later on. The script provides default settings for most of the environment variables to be set. By pressing `-RETURN-`, the default values will be selected; otherwise the proper selection (or path, in case a path is requested) must be typed in.

In case the installation procedure fails for some reason, or you realise the selected options were not correct at the time the installation started, you can repeat the whole process by manually removing the current installation with:

```
> cd geant4/source
> gmake clean
```

where the `$G4SYSTEM` environment variable (specifying the kind of architecture and compiler used) is manually set in your environment (according to the flavors listed in Section 1.1).

In case new modules must be added to an existing build (for example a module for visualization), this can be done manually by re-running `Configure` and providing the new settings.

Once the process of building the libraries has been completed successfully, the Geant4 toolkit can be installed in the specified (already existing) installation area by typing:

```
> ./Configure -install
```

Libraries and necessary source code will be installed in `lib/geant4`, `include/geant4` (if selected to install all headers in a single directory), `src/geant4`, respectively.

`$G4INSTALL` will be set to `[INSTALLATION_AREA]/src/geant4`.

### 2.2. Configuring the Environment to Use Geant4

Once libraries have been installed, the user's environment must be correctly set up for the usage of the Geant4 toolkit. The `Configure` script provides a way to check the existing installation and provide the correct configu-

ration for the user's environment. Configuration scripts `env [ .sh .csh ]` can be generated, and should be sourced by the final users in order to configure their environment according to the performed installation.

To generate the configuration scripts, the user should run `Configure` placed in the installation area, as follows:

```
> $G4INSTALL/Configure
```

This will generate the shell script `env.csh` (`env.sh` for bash shell) to be sourced or integrated into the shell login script (`.tcshrc` or `.bashrc`). The shell script will be generated in the user's current directory (`$PWD`). The user can customize it to specify for example her/his proper working directory through the variable `$G4WORKDIR`. Once the generated script is sourced, the user will be ready to start building a Geant4 application.

Refer to section the section *Getting Started with Geant4 - How to Make an Executable Program* of the Geant4 User's Guide for Application Developers for information on how to build an executable in Geant4.

## 2.3. Installing Geant4 Manually

Before proceeding with the installation, some key environment variables must be defined in your user environment in order to specify where all software components are to be placed and to set some compilation options. A complete reference to all environment variables in Geant4 is available in section *Appendix - Makefiles and Environment Variables* of the Geant4 User's Guide for Application Developers .

### 2.3.1. Required Environment Variables

`G4SYSTEM`:

set to one of the flavors listed in section 1.1 to specify the kind of architecture and compiler used

`G4INSTALL`:

path where the Geant4 toolkit tree is installed (ex. `$HOME/geant4`)

`CLHEP_BASE_DIR`:

path to the CLHEP installation

### 2.3.2. Optional Environment Variables

`G4WORKDIR`:

path of the user's working directory (default in `$G4INSTALL`)

`G4LIB`:

path where the kernel libraries should be installed (default in `$G4INSTALL/lib`)

`G4TMP`:

path where temporary files (object files, dependency files) are placed (default in `$G4WORKDIR/tmp`)

`G4BIN`:

path where final executable files are placed (default in `$G4WORKDIR/bin`).

`G4INCLUDE`:

path where source header files may be mirrored at installation by issuing `gmake includes` (default in `$G4INSTALL/include`)

`G4DEBUG`:

flag specifying that libraries be built with debug symbols (requires a lot of disk space). The default is optimised-mode

`G4LIB_BUILD_SHARED`:

flag specifying that kernel libraries be built as shared libraries (libraries will then be used by default). If not set, static archive libraries are built by default

`G4LIB_BUILD_STATIC`:

flag specifying that kernel libraries be built as static archive libraries. Note that you may specify this flag in addition to `G4LIB_BUILD_SHARED` to build shared and static libraries simultaneously.

`G4LIB_BUILD_G3TOG4`:

flag specifying that the library for the `g3tog4` module be built. By default the library will not be built.

`G4LIB_BUILD_ZLIB`:

flag specifying that an additional library for file compression should be built (not required on Linux/Unix systems). By default the library will not be built.

`G4_NO_VERBOSE`:

defining this flag prevents the compilation of verbosity code (for better performance). The default is with verbosity on.

The Geant4 installation requires native STL (the Standard Template Library) as the base foundation class library. This also implies strict ISO-ANSI language compliance. In addition to the above, you might want to set the proper environment for visualization, such as:

- the kind of graphics driver(s) installed in the system
- the path to the installation of each graphics driver

in case you want to build the Geant4 kernel libraries with the graphics drivers built-in. See *Visualization - The Visualization Drivers* of the Geant4 User's Guide for Application Developers .

At this point, you may choose one of two ways to compile and install the kernel libraries, depending on your needs and system resources. From `$G4INSTALL/source`:

1. `gmake`

This will make one library for each "leaf" category (maximum library granularity) and automatically produce a map of library use and dependencies.

2. `gmake global`

This will make global libraries, one for each major category.

The main advantage of the first approach is the speed of building the libraries and of the application, which in some cases can be improved by a factor of two or three compared to the "global library" approach.

Using the "granular library" approach a fairly large number (roughly 90) of "leaf" libraries is produced. However, the dependencies and linking list are evaluated and generated automatically on the fly. The top-level `GNUmakefile` in `$G4INSTALL/source` parses the dependency files of Geant4 and produces a file `libname.map` in `$G4LIB`. `libname.map` is produced by the tool `liblist`, whose source code is in `$G4INSTALL/config`.

When building a binary application the script `binmake.gmk` in `$G4INSTALL/config` will parse the user's dependency files and use `libname.map` to determine through `liblist` the required libraries to add to the linking list. Only the required libraries will be loaded in the link command.

The command `gmake libmap` issued from `$G4INSTALL/source`, allows manual rebuilding of the dependency map. The command is issued by default in the normal build process for granular libraries.

It is possible to install both "granular" and "compound" libraries, by typing "gmake" and "gmake global" in sequence. In this case, to choose usage of granular libraries at link time one should set the flag `G4LIB_USE_GRANULAR` in the environment; otherwise compound libraries will be adopted by default.

## 2.4. Integrating Geant4 into a Generic Framework

As part of the Geant4 kernel libraries installation, it is also possible to put the entire set of header files in a single place, which is determined by the environment variable `G4INCLUDE` specifying the directory path. Therefore, it's rather straightforward to integrate Geant4 into a generic external framework, by simply knowing the path where header files are located in the system (`G4INCLUDE`) and where installed libraries are placed (`G4LIB`).

In *Appendix - Makefiles and Environment Variables* of the Geant4 User's Guide for Application Developers , you can find a list of all the environment variables, together with a section explaining how to integrate external libraries which may or may not use Geant4 kernel libraries, in the GNUmake mechanism of Geant4.

---

## Chapter 3. Tips for Installing on Windows

### 3.1. Windows with the Cygwin Toolset and Microsoft Visual C++

To compile and run Geant4 under Windows systems, some additional information and tools are required, although the installation procedure is similar to that required on a UNIX based system. On Windows, the Cygwin toolset and the Microsoft Visual C++ compiler are used.

Cygwin32 is a UNIX development environment available for Microsoft Windows. You can freely obtain Cygwin32 from [Cygwin](#) or [Cygwin/X](#) .

We do not support direct use of Visual Studio; i.e. we do not provide Visual Studio workspace (.dsw) or project (.dsp) files, nor we do provide makefiles for the `nmake` application of MS Visual C++.

We use several of the tools provided by the Cygwin toolset:

- `make.exe` as a make tool
- `g++.exe` as a tool to analyse source file dependencies and create dependency (.d) files
- several other unix tools like `cp`, `mv`, `rm`, `touch`, etc.

At the installation of the Cygwin toolset it is therefore required to explicitly select some packages (i.e. `gmake`, `gcc`, `binutils` and `tcsh` from the "devel" category) in addition to those which are part of the default installation.

For more details on the toolset, please see the documentation available on the [Cygwin pages](#) . The User's Guide has quick start guides and help with setting up Cygwin with `setup.exe`.

Links to some installation tips for the Cygwin toolset and also "step by step" installation guides can be found in the section *Appendix - Build for MS Visual C++* of *Geant4 User's Guide - For Application Developers*

The usage of Cygwin32 for the build environment results in a build procedure similar to that on a UNIX system. The documentation in the User's Guide for Application Developers, Section 10.5, is largely applicable in this case.

The following steps are required to install geant4:

1. Install Cygwin (see also notes above):

We do not depend on any specific feature of the version of Cygwin, so newer versions (or slightly older) are expected to work.

2. Set the environment for MS Visual C++, i.e. set the paths to libraries, include files, and executables for MS Visual C++. This can be done by modifying the start-up batch file `cygwin.bat` of Cygwin32 to include all the MSDOS commands found in the file `vcvars32.bat` provided in MS Visual C++ (in the VC++ .NET compiler installation directory, and usually located inside the `Common7/Tools` directory). This modification should be done after the installation of the Cygwin toolset, so that the environment gets properly set at the time of the invocation of the shell.
3. Unpack the source code into a directory of your choice.
4. Start the `Cygwin` shell from the start menu.
5. At this point you're ready to install Geant4. If manual installation is chosen, you must set the necessary environment variables. There are various ways to do so; for example through a command file for the Cygwin bash shell. This command file could be named `geant4-setup.sh`, and you must execute it with

```
. geant4-setup.sh
```

including the leading dot and blank space. You could also have this as your `.bashrc` file. The commands in the command file should be:

```
# Set G4SYSTEM
```

```
export G4SYSTEM=WIN32-VC
#
# Set Path to CLHEP
export CLHEP_BASE_DIR=C:/usr/local
#
# --- Other optional settings
#Turn on verbose to show command used for compilation
#export CPPVERBOSE=1
#
```

Note, in the example above, CLHEP was installed in C:/usr/local, therefore include/CLHEP and lib/CLHEP.lib must be included therein.

6. Once the environment is correctly set, the libraries are built using make from the geant4/source directory typing make from the Cygwin bash shell prompt.

Examples can be built in the same way as the libraries from examples/novice/N01, for instance: type make from the shell prompt.

Note that, depending on which external software is used, there may be some warnings in linking; e.g., we get "unrecognized source file type -file.o-, object file assumed", or warnings of conflict-ings libraries. This often seems to be caused by an external library compiled for a different run time environment.

The binary of the example is placed by default into the geant4/bin/WIN32-VC directory. You may run it either from this directory or from the examples/novice/N01 directory; sample input and output files are placed in each of the examples/novice directories. Some of the examples will need to read data files, and the place has to be given in environment variables again similar to the following example:

```
#
# Environment variables needed to find geant4 data files:
#
# Data for neutron scattering processes,
# distributed in a separate tar file, then placed under data
export NeutronHPCrossSections=c:/usr/local/geant4/data/G4NDL
#
# Nuclear Photon evaporation data,
# distributed with the source files under data
export G4LEVELGAMMADATA=c:/usr/local/geant4/data/PhotonEvaporation
#
# Data for radiative decay hadronic processes under data,
# distributed in a separate tar file
export G4RADIOACTIVEDATA=c:/usr/local/geant4/data/RadiativeDecay
#
# Data for low energy electromagnetic processes,
# distributed in a separate tar file, then placed under data
export G4LEDDATA=c:/usr/local/geant4/data/G4EMLOW
#
```

All compiler and linker options are set in config/sys/WIN32-VC.gmk. If you require options different from our choice, you can modify this file.

## 3.2. Building Kernel Libraries DLLs

DLLs (Dynamic Link Libraries) on Windows are supported for .NET VC++ and can be built for the compound kernel libraries of Geant4 (see the Installation Procedure of this Guide for a dissertation on global/compound libraries).

The libraries can be built either manually, issuing the command:

```
make dll
```

from the directory \$G4INSTALL/source or by specifying it through the Configure script used for the installation.

Then, to build any application making use of the installed DLLs, the environment variable G4LIB\_USE\_DLL must be set in the environment.

Once the application is built, it is required to specify to the system the path where the DLLs are installed. To do so, add the absolute path (in Cygwin format) of the DLLs installation directory to the `PATH` variable; for example:

```
export PATH=$PATH:/usr/local/geant4/lib/$G4SYSTEM
```

You may then be able to run successfully your application.