Installing SystemC

The following Installation notes are compiled from information obtained from different sources. These may not be accurate but these should give you some guidelines as to how SystemC can be installed.

1. Installation in Visual C++ (Windows)

The following installation instructions are based on Visual C++ 6.0. For Visual C++ . NET, use msvc70 sudirectory understand systemc and do similar operations as described below.

A. Installing To Your Local Computer

1. Download SystemC from http://www.systemc.org
2. Unzip SystemC files to a hard drive with "plenty" space. Preferably, unzip the files to the hard drive's directory (ex; C:\SystemC or E:\SystemC)
3. The SystemC distribution includes project and workspace files for Visual C++. If you use these project and workspace files the SystemC source files are available to your new project. For Visual C++ 6.0 the
   project and workspace files are located in directory: **...\systemc-2.1v\msvc6x** , where "..." is whatever parent directory you saved SystemC to.
4. Click on the subdirectory: `systemc' which contains the project and workspace files to compile the `systemc.lib' library. Double-click on the `systemc.dsw' file to launch Visual C++ with the workspace file. The workspace file will have the proper switches set to compile for Visual C++ 6.0.
   Select `Build systemc.lib' under the Build menu or press F7 to build `systemc.lib'.

B. Creating a new design

1. Start Microsoft Visual C++ 6.0.
2. Create a Project Workspace:
   a. Click on "File", then "New", select "Projects", then click on "Win32 Console Application".
   b. In the "Location" box, click on the down arrow, then on "Drives", and choose "h:\\systemcfiles\aproject" , then click "OK".

c. For the "Project Name", We will use "lab1a" as the example.

d. Type "OK".

e. Choose "An empty project" and click "Finish". Then click "OK".

3. You can now see a folder named "lab1a classes" in the workspace window. (Left part of screen)
4. Port SystemC libraries to Microsoft Visual C++ 6.0:
    a. Click on “Project”, then “Settings”, then select the C/C++ tab, and then finally select the “C++ Language” category. Make sure that the “Enable Run Time Type Information (RTTI)” checkbox is checked.

    b. Also make sure that the SystemC header files are included by switching to the “Preprocessor” on the C/C++ tab and then typing “C:\SystemC\systemc-2.1\src” in the text entry field labeled “Additional include directories”.

    c. Next click on the “Link” tab, and make sure the SystemC library is included to your project by typing “C:\SystemC\systemc-2.1\msvc60\systemc\Debug” in the text entry field labeled “Additional library path”.

    d. Add the SystemC object files by first clicking on “Project”, then “Add to Project”, then “Files”. In the File Browser navigate to the “C:\SystemC\systemc-2.1\msvc60\systemc\Debug” directory. In the text entry field labeled “File Name” type “*.obj” and press enter. Click on the file “sc_attribute.obj” and then simultaneously press the “Ctrl” & “A” keys (CTRL+A). Click the OK button to add the files.

    e. In your workspace window under the “File View” Tab, you should see a number of object files with the “sc_” prefix such as sc_attribute.obj, sc_bit.obj, etc. Find the file “sc_isdb_trace.obj”, click that file name, and press “delete” on your keyboard.

If you are about to install SystemC on your platform you should pause one moment and think about the compiler selection. At a first glance, SystemC is built on top of ANSI C++, so a standard compiler should work. Unfortunately this is not the case, i.e. you can't simply use **any** compiler.

In the following we suppose you want to install the actual version of SystemC (2.1) and the SystemC Verification Library (1.0p1). Go to the www.systemc.org and select the appropriate packages.

2. Installation under UNIX-like systems

For using SystemC under e.g. Linux/Solaris/MacOS the GNU compilers are the right choice. The compiler of your choice could either be gcc 2.95.3 or gcc 3.2.3, for instance. It should be able to install the compiler of your choice.

If there occur any errors, please refer to the GNU (www.gnu.org) manual pages. Binary packages should also exist for most distributions and operating systems.

Note: SystemC was developed and tested using different operating systems and compilers. So, you are advised to check out the `INSTALL` and `README` file coming with the SystemC package. All possible combinations are listed there, so have a look at that!

A. Installing SystemC with a specified compiler

Next, we want to install SystemC 2.1 using our dedicated compiler. The compiler should be in your environment search path. You can check the compiler version by typing:

- `gcc --version`

If your compiler was installed to another directory (let's assume `/usr/local/`), you may need to set the environment variables `CXX` and `CC` to the location of your compiler.

According to your local shell, type

- ```
  export CXX=/usr/local/gcc-x.y.z/bin/g++
  export CC=/usr/local/gcc-x.y.z/bin/gcc
  ```

or

- ```
  setenv CXX /usr/local/gcc-x.y.z/bin/g++
  setenv CC /usr/local/gcc-x.y.z/bin/gcc
  ```

The variables `CXX` and `CC` point to the installation of our compiler and will be used by the configure script. In other words we advise the installation routine to use our specified compiler. You can check if the variables are set by typing:

- ```
  echo $CXX
  echo $CC
  ```

Now, we are ready to install SystemC:

Create a temporary folder in the SystemC sources folder and switch to that folder, like
  ```

- ```
  mkdir objdir
  cd objdir
  ```

Now, we need to do some more steps (on some systems you might rather use `gmake` instead of `make`):

- ```
  ../configure --prefix=/usr/local/systemc-2.1
  make
  make install
  ```

The call `make` will compile SystemC, `make install` will install the package to the specified target folder and **must** be executed as user root!

It makes sense to provide an installation folder with `--prefix=/usr/local/systemc-2.1`. All necessary and important files are kept together, and there can easily be installed multiple versions of SystemC that way. If you want to get rid of the installation you can simply delete the whole folder.

B. Known issues

If there occur any problems like error messagers during the installation process, you may need to switch to another compiler. If done so, you have to re-configure the build process. As we have created a temporary folder, we simply delete that one and restart the build process.

- ```
  rm -rf *
  ../configure --prefix=/usr/local/systemc-2.1
  make
  make install
  ```

Note: Take care when removing files by typing `rm -rf *`. All files will be deleted irrevocably, so be sure you're in the right folder, like the temporarily created folder `objdir`.

The command `make install` needs to be executed as user root, whereas the other commands do not necessarily need to be executed as user root. Sometimes problems may occur if you do execute `configure` and `make` as root!

Maybe you need to do something more before the build and configuration process. Change to user root and explicitly create the target folders. These folder names can be applied to both SystemC and SCV.

- ```
  cd /usr/local
  mkdir systemc-2.1
  mkdir systemc-2.1/include
  ```

```
mkdir systemc-2.1/lib-linux
mkdir systemc-2.1/doc
mkdir systemc-2.1/examples
```

Note: In this example we want to run SystemC under Linux, as the path to the library is `lib-linux`. According to your platform, other names for folders may be more rational (see table)!

Platform	**Name of folder**
Linux	lib-linux
Sun Solaris (GNU compiler)	lib-gccsparcOS5
Sun Solaris (Sun C++ compiler)	lib-sparcOS5
MacOS X	lib-macosx
HP UX (GNU compiler)	lib-gcchpux11
HP UX (HP C++ compiler)	lib-hpux11
Cygwin	lib-cygwin

C. Installing the SystemC Verification Library (SCV)

With SystemC installed, we can now install the SystemC Verification Library.

You should use exactly the same compiler for installing SystemC and installing SCV!

Remember to set the variables `CXX` and `CC` to your compiler installation, if not done already! Perform the same steps as already stated above.

- ```
  mkdir objdir
  cd objdir
  ../configure --prefix=/usr/local/scv --with-
  systemc=/usr/local/systemc-2.1
  make
  make install
  ```

The SCV installation needs the SystemC installation as a reference. This can be achieved with `--with-systemc=/usr/local/systemc-2.1`. We also choose an installation folder with `--prefix=/usr/local/scv`, so SCV will be installed to that specified folder.

Due to the compiler problem, you should carefully read the `INSTALL` and `README` files in the SCV package. If any problems occur, have a look at the known issues.
  ```

D. Running SystemC and SCV

To run programs you need to do some more things. The SCV library can be loaded dynamically, so your executable specification needs to know the installation path.

There are two possibilities how to facilitate that (in this example we want to run SystemC and SCV under Linux, hence the folder name `lib-linux`. If your platform is a different one, please look here):

According to your shell, type

- `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/scv/lib-linux`

or

- `setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/usr/local/scv/lib-linux`

You need to do that once in your shell. Maybe useful if you are working with multiple versions of the library.

As an alternative, you can add the line above in your `~/.bashrc` or `~/.profile` so the variable is set everytime when you open a new shell.

Another alternative is the following: one of the folders where your system stores dynamically loadable libraries may be `/lib`, `/usr/lib` or `/usr/local/lib`. Check for that -- let's assume it is `/usr/local/lib` on your system:

- ```
  cd /usr/local/lib
  ln -s /usr/local/scv/lib-linux/libscv.so .
  ln -s /usr/local/scv/lib-linux/libscv.a .
  ldconfig
  ```

First we go to that folder. The `ln -s` command will create a symbolic link, that is, you create a reference to the original file without duplicating it. The command `ldconfig` will update the internal dynamic linker database. After that the library will be found and loaded dynamically at runtime.

These commands (especially `ldconfig`) have to be executed as user root.

Every model specification must be build with the same compiler you used to build SystemC and SCV! This can be achieved by appropriate Makefiles.

By the way, you may need to adapt the Makefiles for the SystemC examples to your compiler and installation paths.
  ```

Running SystemC and SCV examples

After having successfully installed the libraries, you can test the installation. Stay in the temporarily created folder `objdir` and check if there are folders named `include` and `lib-linux` (if Linux is not your platform, please look here).

If that folders do not exist, we again create symbolic links by typing:

- ```
  ln -s /usr/local/systemc-2.1/include .
  ln -s /usr/local/systemc-2.1/lib-linux .
  ```

Otherwise, the installation routine has created that folders itself, and we can test the installation. Just type

- ```
  make check
  ```

This will compile and run each and every example.

Another way to check the installation is to visit the example folders separately, like

- ```
  cd examples/systemc/simple_bus
  make check
  ```

Please note that the sources are not contained in the `objdir`
  ```