

# Libsupcxx

From OSDev Wiki



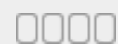
This page is a work in progress and may thus be incomplete. Its content may be changed in the near future.

The factual accuracy of this article or section is disputed.

[Please see the relevant discussion on the talk page.](#)

Libsupc++ is a support library for g++ that contains functions dealing with run-time type information (RTTI) and exception handling. If you attempt to use either exceptions or RTTI in a C++ kernel you have compiled with a GCC Cross-Compiler you will also need the libsupc++ library. In general, you should be able to use the one provided as part of a Linux distribution. If, however, you run into problems and need to compile your own, you can follow these steps.

**Difficulty level**



Not rated

## Contents

- 1 Compiling libsupc++
  - 1.1 Create a working GCC Cross-Compiler.
  - 1.2 Configure gcc
  - 1.3 Edit the libstdc++ configure script
  - 1.4 Configure and make libsupc++
  - 1.5 Usage
  - 1.6 Additional requirements
  - 1.7 Tested on
- 2 Full C++ Runtime Support Using libgcc And libsupc++
- 3 Linking a kernel with libsupc++

## Compiling libsupc++

### Create a working GCC Cross-Compiler.

This tutorial assumes it is entitled 'i686-elf-gcc'

### Configure gcc

Enter the gcc source directory, run

```
./configure --target=i686-elf --prefix=/usr/local/cross --enable-languages=c,c++ \
--without-headers --disable-nls
cd libstdc++-v3
```

## Edit the libstdc++ configure script

Now you need to edit the configure file in the libstdc++-v3 directory. Open it up in the editor of your choice (which preserves unix style line endings) and find a section similar to (it is around line 108,000 in gcc 4.2.1, searching for 'combination' is probably the easiest way to find it):

```
{ { echo "$as_me:$LINENO: error: No support for this host/target combination." >&5
echo "$as_me: error: No support for this host/target combination." >&2;}
{ (exit 1); exit 1; }; }
;;
```

and alter the third line so that it reads:

```
{ { echo "$as_me:$LINENO: error: No support for this host/target combination." >&5
echo "$as_me: error: No support for this host/target combination." >&2;}
}
;;
```

## Configure and make libsupc++

```
CPP=i686-elf-cpp ./configure --host=i686-elf --prefix=/usr/local/cross --disable-hosted-libst
--disable-nls
cd include
make
make install
cd ../libsupc++
make
make install
```

## Usage

Libsupc++ should now be installed into /usr/local/cross/lib. To use it, you will need to add

```
-L/usr/local/cross/lib -lsupc++
```

to your linker command line.

## Additional requirements

Libsupc++ also requires that libgcc.a be included in your link as well. This is usually found (if you followed the cross compiler directions) in /usr/local/cross/lib/gcc/i686-elf/<gcc version>. Finally, it has a number of dependencies which your kernel must provide, including (but not limited to) malloc, free, abort and strlen.

## Tested on

These steps were tested on g++ 4.2.1 under Cygwin with a cross compiler targeting i686-elf

## Full C++ Runtime Support Using libgcc And libsupc++

The following description is valid for i386, GCC 3.2 and libgcc/libsupc++ compiled for Linux/glibc (you can use the static libgcc/libsupc++ libraries compiled for your Linux for your kernel).

If you want Exceptions, RTTI, new and delete altogether, you also could use libgcc and libsupc++. libgcc contains the unwinder (for exceptions), while libsupc++ contains the C++ support. These functions look very complex (gcc\_sources/gcc/unwind\*, gcc\_sources/libstdc++-v3/libsupc++/\*), so it might be better to port them instead of trying to write them yourself.

To get full C++ support, you only have to do the following:

- Provide some libc functions (e.g. abort, malloc, free, ...) because libsupc++ needs them. There are even more functions you could support, like pthread\_\*, but since these are weak symbols, you don't need to define them.
- There's also a strange function dl\_iterate\_phdrs. You don't need this so let it simply return -1. It's usually used to find exception frames for dynamically linked objects. You could also remove calls to this function from the library.
- To make use of exception handling, you also have to tell libsupc++ where the **.eh\_frame** section begins. Before you throw any exception: `<verbatim>__register_frame(address_of_eh_frames); </verbatim>`.
- Terminate the **.eh\_frame** section with 4 bytes of zeros (somehow). If you forget this, libsupc++ will never find the end of **.eh\_frame** and generate stupid page faults.

Please note that you still have to call the constructors/destructors by yourself as documented in Calling Global Constructors.

## Linking a kernel with libsupc++

You can use your libsupc++ to get exception handling and RTTI in a C++ kernel (no more passing -fno-exceptions -fno-rtti to g++!) so you can use things like throw and dynamic\_cast<>. Libsupc++ depends upon libgcc for stack unwinding support. Passing the -nostdlib option to gcc when linking caused libgcc.a and libsupc++.a to not be included, so you need to specify -lgcc -lsupc++ on the command line (no need to specify the directories; gcc knows where it installed them to). In addition, you need to include a .eh\_frame section in your linker script and terminate it with 32 bits of zeros (QUAD(0) is a useful linker script command). The symbol start\_eh\_frame should point to the start of the eh\_frame section, and it should be aligned by 4. In addition you need to include your constructors and destructors in the link (see C++ for details). You also need to provide \_\_register\_frame() (or call the function provided by libgcc with the start of your .eh\_frame section), void \* \_\_dso\_handle;, \_\_cxa\_atexit() and \_\_cxa\_finalize (again see C++). Something along the lines of

```
#include <reent.h>
static struct _reent global_reent;
struct _reent *_impure_ptr = &global_reent;
```

somewhere in your kernel will keep libgcc happy, because it expects these bits to be provided by the standard library (which you aren't linking into your kernel - but you can provide them in your libk). Libgcc expects a number of (simple) C library functions to be provided by your kernel, including abort, malloc, free, memcpy,

memset and strlen. Libsupc++ also requires write, fputs, fputc, fwrite, strcpy and strcat for debugging output.

Retrieved from "<http://wiki.osdev.org/index.php?title=Libsupcxx&oldid=17502>"

Categories:      In Progress | Disputed Pages | Level 0 Tutorials | Tutorials

---

- This page was last modified on 28 January 2015, at 07:50.
- This page has been accessed 23,383 times.