

## Starting With The SDK: A Quick Guide (To Avoiding Hair loss)

Written by nblachford  
Friday, 15 June 2007

If you are used to just firing up an editor, tapping in some code and compiling then you're quite likely to run into problems with the Cell SDK. It doesn't quite work that way...

You'll probably get a "undefined reference" error then think, huh? You do a search of the IBM SDK forums and lo and behold you notice a heap of other people have had similar problems - but not necessarily the exact same problem...

To get code to work you need to use the provided compilers, headers, libraries and make files. The make files are very important because they set up the environment so the compiler can find the files it requires.

These files will almost certainly not be in the normal places so without the make environment the compiler can't find things and you get undefined references - and quite possibly hair loss as you try and figure out what the feck went wrong.

To avoid these you need to do a number of things:

### Build Everything

First, ensure that you have built everything, you should have done this when installing. If you didn't use the command "cellsdk build". You need to do this because some of the libs are prototypes and are built by this command.

You can specify which compiler does the building by adding --gcc or --xlc to this command. Whatever you specify then becomes the default compiler and is invoked when you type "make". You can change the default compiler but you need to look through the documentation on how to do this.

If you don't specify anything it defaults to gcc.

The second thing to do is (optionally) create a sand box. This means copying the contents of /opt/ibm/cell-sdk/prototype to your home directory. Doing this ensures you can change anything you like but if it messes up you always have the original.

### Set \$CELL\_TOP

Once you have done this you need to set the environment variable CELL\_TOP, this points to the prototype directory as it contains the files for setting up the make environment. Without this you're likely to have some headaches.

You'll need to add a couple of lines to your .bashrc file (a hidden file in your home dir), they'll look like this (replace relevant sections of course):

```
CELL_TOP='/your_home_dir/path_to_prototype'  
export CELL_TOP
```

The prototype directory contains a number of files that the make process depends on, by setting CELL\_TOP it can find the relevant files.

When you next start a shell CELL\_TOP will be set. You'll need to add this set up to any user account you are using for development.

## Code Samples

The SDK samples use a certain structure to organise source.

At the top level you have a directory for a sample app, this will contain a make file, it doesn't do a lot but does set up which dirs have to be entered when compiling and the order they are in. In the DIRS line spu should be listed before ppu, if not the linker will attempt to link in spu files which haven't been built yet and you'll get errors.

Below this you'll have 2 dirs: spu and ppu.  
Each of these dirs contain code and their own make files.

If you are getting undefined references this is where you can fix them.

You should really be using the SDK maths functions but if you just want to get an app working quickly you might have some existing math.h functions included, if so you're likely to get an undefined reference.

To fix this edit the make file in the ppu dir.

In the IMPORTS section add -lm to the end of the line (make sure there's a space before it). If you now compile you'll find the maths functions can magically be found and the undefined reference vanishes.

Another area where you may get undefined references is if you use functions such as malloc\_align(), this allocates memory aligned in a power of 2 you specify (Cell is exceedingly fussy about alignment so you'll need this). Make sure you use free\_align() to free the memory, you're likely to get crashes otherwise.

These functions can be found in libmisc. libmisc is not an "official" lib yet as it's still in testing so it and some other libs go in a different directory. Unfortunately the compiler doesn't have the slightest idea where this can be found, unfortunately the make environment files don't appear to help in the matter, you have to add it manually.

The answer is to add a path to the library file in the IMPORTS section of the makefile.  
You can do this in variety of ways, I use this:

```
$(CELL_TOP)/sysroot/usr/lib/libmisc.a
```

You can also use a relative path, but you'll need to change it if your working dir is different e.g.:

```
../../../../sysroot/usr/lib/libmisc.a
```

each ../ means go up a directory level, in this case going up 4 levels gets to prototype. The path below prototype is:

```
sysroot/usr/lib/libmisc.a
```

## Compiler Flags

If you want to give the compiler specific flags for different option you can do this by adding a CFLAGS section to the make file.

e.g.

```
CFLAGS = -Wall -O3 -v
```

These flags are then passed to the compiler.

## **Conclusion**

Hopefully that will help getting compilation working, at least for SDK 2.1. It took me quite some time to figure that lot out and I imagine other people have had the same problems. The SDK is still in flux so expect things to change in future versions, these sorts of problem are only to be expected on a new platform but it'll get easier with time.

Remember there's a load of good SDK documentation to read through and if you run into problems have a look at IBM's Cell SDK forum.