Guide to BOOK Chapter 10 Examples:
John ffitch

Understanding an Opcode in Csound

There are two versions of the example used to demonstrate how to create a new opcode in Csound. The first version is designed to be an 'internal' opcode, compiled inside the Csound library, together with all other inner bits of code. The second version is a plug-in, i.e. an opcode that can be loaded by Csound, so it is built separately and independently of the main part of the software.

negate.h and negate.c

These files are the source code for the 'internal' version of the ***negate*** opcode. Building this opcode requires that you have the Csound sources and a few dependencies (**libsndfile** and possibly **PortAudio**). The former can be downloaded from http://www.mega-nerd.com/libsndfile and the latter from http://portmedia.sourceforge.net. These sites contain full instructions for the installation of these libraries. Csound is built using a build system called **scons**, which can be downloaded from http://www.scons.org, where again full instructions can be found for its installation. The sources for Csound can be downloaded from http://csound.sourceforge.net. Once you have these, you can start adding the opcode. These are the steps:

1. Copy *negate.c* into the sources directory `./OOps` and *negate.h* into `./H`
2. Next you will need to edit some of the Csound sources. In the directory `./H`, find the file with the name *entry1.h* and add the following line somewhere (e.g. next to the similar looking lines):

```
#include "negate.h"
```

3. At the end of the file, add the prototype for the negate function:

```
int negate(CSOUND*, void*);
```

4. Now, open the file *entry1.c* in the `./Engine` directory and insert the following line somewhere in between any of the similar looking lines:

```
{ "negate", S(NEGATE), 4, "a", "a", NULL, NULL, negate },
```

5. You will need to add your new source code filename to the *SConstruct* script file (in the top-level directory). Open it and around line 1106 you should see a:

```
else:
      libCsoundSources = Split('''
```

```
Engine/auxfd.c
...
Top/utility.c
''')
```

Placing the name (and directory location) of your source code file anywhere inside this list (for instance after the last line but before the triple quotes) will allow it to be compiled as an internal component of Csound:

```
  else:
      libCsoundSources = Split('''
Engine/auxfd.c
...
Top/utility.c
OOps/negate.c
''')
```

6. Now, build Csound. This is just a matter of running the following command in the top-level directory of the source code:

```
$ scons
```

The opcode now is part of Csound and you can use it in one of your instruments (you can look at the chapter notes on details of running your own custom-built version of Csound).

negate_plugin.c

This is a plug-in opcode and in general you will not need to have Csound installed in order to built it. However, since in this text we have discussed building Csound, a simple way of compiling and linking this file into a dynamic library module is to take advantage of *SConstruct*. These are the steps:

1. Copy the file into the plug-in opcodes source directory, `./Opcodes`.

2. Open *SConstruct* and locate the comment below:

```
############################################################
# Plugin opcodes.
############################################################
```

3. Insert the following call in the line just below the comment:

```
makePlugin(pluginEnvironment, 'negate, ['Opcodes/negate_plugin.c'])
```

4. Build Csound as explained above.

Note that, in order to avoid confusion between two opcodes with the same name, we have called this opcode ‘negate2’ in the source file. Now you should be able to use it in your instruments (but read the chapter notes on running your own custom-built version of Csound).