Guide to BOOK Chapter 3 Examples:
Gabriel Maldonado

Working with Audio Streams

HelloSine

To build the first example, all you need is to type the following command:

```
$ gcc -o hellosine hellosine.c
```

To run it, just type:

```
$ ./hellosine
```

You will get the sinewave stream as a series of numbers to the screen. If you prefer to send this stream to a text file, type:

```
$./hellosine > sound.txt
```

Libraries

The next examples use three libraries, two of which you can build from the sources supplied here, *portsf* and *tiny*. The third, *portaudio*, can be downloaded and installed from its main distribution site.

portsf

To build this library, go to the *portsf* sub-directory and run make:

```
$ make
```

To install it ready for your own use, just type:

```
$ make install
```

tiny

To build this library, go to the *TinyAudioLibrary* sub-directory and run make:

```
$ make
```

To install it ready for your own use, just type:

```
$ make install
```

portaudio

Instructions for downloading and installing Portaudio (v.19) are found at http://portmedia.sourceforge.net/

The HelloTable Examples

You can now build four versions of *HelloTable*: for stdio output, for binary-raw output, for wavefile output or realtime output.

stdio output

This is the most basic example, producing an output similar to *HelloSine*. The command to build it is also the simplest; we will call the executable *hellotable1*:

```
$ gcc -o hellotable1 hellotable.c
```

You can run this program as follows:

```
$ ./hellotable1
```

binary-raw output

This example will produce its output stream as a binary (raw) soundfile. You will need to add the token definition 'BINARY_RAW_FILE', using the flag `-D`:

```
$ gcc -o hellotable2 hellotable.c -DBINARY_RAW_FILE
```

To produce the raw soundfile called *hellotable.raw*, you can run this program as follows:

```
$ ./hellotable2
```

wave file output

This example will produce its output stream as a binary wave-format soundfile. You will need to add the token definition 'WAVE_FILE', using the flag `-D`, and tell the compiler where to find the headers and the library file for *portsf*:

```
$ gcc -o hellotable3 hellotable.c -DWAVE_FILE -I./include
     -L./lib -lportsf
```

To produce a wave soundfile called *hellotable.wav*, you can run this program as follows:

```
$ ./hellotable3
```

realtime output

This example will send its output stream to the soundcard (dac). You will need to add the token definition 'REALTIME', using the flag `-D`, and tell the compiler where to find the headers and the library file for *tiny* and *portaudio*:

```
$ gcc -o hellotable4 hellotable.c -DREALTIME -I./include
    -I/usr/local/include -L./lib -L/usr/local/lib -ltiny -lportaudio
```

To play the sound in realtime through your default soundcard, you can run this program as follows:

```
$ ./hellotable4
```

The Soundfile Player Examples

The two soundfile players will require the three libraries used in the previous examples. In order to build the first one, you will need the command:

```
$ gcc -o player player.c -I./include -I/usr/local/include -L./lib
  -L/usr/local/lib -lportsf -ltiny -lportaudio
```

This program can be run with the simple command:

```
$ ./player soundfile
```

where *soundfile* is the name of an existing WAVE or AIFF soundfile.

To build the second soundfile player, you will need a similar command, with the extra source code *crack.c*:

```
$ gcc -o player2 player2.c crack.c -I./include -I/usr/local/include
  -L./lib -L/usr/local/lib -lportsf -ltiny -lportaudio
```

This program can be run with the simple command:

```
$ ./player2 [-tTIME -dDUR] soundfile
```

where *soundfile* is the name of an existing WAVE or AIFF soundfile; and where TIME and DUR are start-time and playback duration in seconds (optional).

HelloRing

To build this example, you will need only the portaudio library:

```
$ gcc -o helloring helloring.c -I/usr/local/include -L/usr/local/lib
       -lportaudio
```

To run it, just type

```
$ ./helloring
```

and follow the instructions on the terminal. Enjoy!