Guide to DVD Chapter 12 Examples: *Joo Won Park*

Compiling Software in the OS X Terminal: Unix-style

The *examples* folder contains two subfolders: the *Dobson-MakefileOnly* folder, that contains *.c* files from the Dobson chapters and their corresponding *Makefiles*; and the *Dobson-Complete* folder that contains all the files in *Dobson-MakefileOnly* folder plus working Unix executable files compiled from the corresponding code.

How To Compile the Code In the Dobson-Makefileonly Folder

To make an executable file from the code provided in this folder, go to a subfolder and type **make** in the Terminal. For example, if you go to a folder named *c1-1* , list folder contents with **ls**, and then type **make**, you will get the following message in your OS X Terminal:

```
$ ls
Makefile    dobson1.c
$ make
cc -c -g dobson1.c
cc -o Hello dobson1.o
$ ls
Hello       Makefile    dobson1.c   dobson1.o
```

After running **make,** you will notice that there are two new files, *Hello* and *dobson1.o*. To test an executable file, type the file name after "**./**" (dot-slash). For example, you will get the following message in the Terminal when you run the executable file *Hello*:

```
$ ./Hello
What A Wonderful World!
```

Important: For the Dobson Book Chapter 2 examples, you need to run the *Makefile* in the *portsf* folder first as the *portsf* library is needed for audio-related Dobson code.

How To Use the Dobson-Complete Folder

All executable files contained in the *Dobson-Complete* folder should run without additional configurations. The code and executables in this folder are the result of running the *Makefiles* in the *Dobson-MakefileOnly* folder. For some subfolders, I have also included example `.txt` files that are necessary to test the executables.

For the code that corresponds to Dobson Book Chapter 2, I have included an executable file called ***Testrun***. *Testrun* is a script that shows one way of using the corresponding executable file with *flags* and *arguments*. For example, when I run *sfpan* in folder *c2-3* without any arguments, I get the following syntax error messages:

```
$ ls
Makefile    breakpoints.o     sfpan       sfpan.o
Testrun     libportsf.a       sfpan.c     testenv.txt
$ ./sfpan
SFPAN: pan mono sound into stereo
insufficient arguments.
usage:
      sfgain infile outfile posfile.brk
      posfile.brk  is breakpoint file with values in range
      -1.0 <= pos <= 1.0 where -1.0 = full Left, 0 = Centre,
      +1.0 = full Right
```

However, when I run *Testrun*, *sfpan* creates an audio file with a preset argument:

```
$ ls
Makefile    breakpoints.o     sfpan       sfpan.o
Testrun           libportsf.a sfpan.c     testenv.txt
$ ./Testrun
THE FOLLOWING COMMAND WAS EXECUTED
./sfpan ../testvox.aif sfpan.aif testenv.txt
SFPAN: pan mono sound into stereo
processing....
Done: 0 errors
PEAK information:
CH 1: 0.8028 at 5.8574 secs
CH 2: 0.7534 at 5.6837 secs
$ ls
Makefile    breakpoints.o     sfpan       sfpan.c
testenv.txt Testrun           libportsf.a sfpan.aif   sfpan.o
```

To figure out what is going on in *Testrun*, open the file with any text editor. You may have noticed from the above example that running *Testrun* is as same as running the following command in *sfpan* example:

```
./sfpan ../testvox.aif sfpan.aif testenv.txt
```

Important: Do not move or delete the *common*, *include*, and *portsf* folders or the *testvox.aif* file in the *Dobson-Complete* folder as the examples will not work without them in the specific directory.

Suggested Usage

I suggest to do the following if you want to study the *Makefile* examples included in here.

1. Go to *Dobson-MakefileOnly* folder and open a *Makefile* with a text editor
2. Study the script in the selected *Makefile*
3. Compile the code using **make**
4. Test the executable file.
5. If the executable does not work, compare your *Makefile* and other necessary code with the files in *Dobson-Complete* folder.
6. Once you understand the mechanisms of *Makefile*, create your own *Makefile* for some of the Dobson code or, even better, your own codes.