A.1 Makefiles and Flex

In this appendix, we provide tips for writing Makefiles to build your scanners.

In a traditional build environment, we say that the `.c` files are the sources, and the `.o` files are the intermediate files. When using `flex`, however, the `.l` files are the sources, and the generated `.c` files (along with the `.o` files) are the intermediate files. This requires you to carefully plan your Makefile.

Modern `make` programs understand that `foo.l` is intended to generate `lex.yy.c` or `foo.c`, and will behave accordingly[12]. The following Makefile does not explicitly instruct `make` how to build `foo.c` from `foo.l`. Instead, it relies on the implicit rules of the `make` program to build the intermediate file, `scan.c`:

```
# Basic Makefile -- relies on implicit rules
# Creates "myprogram" from "scan.l" and "myprogram.c"
#
LEX=flex
myprogram: scan.o myprogram.o
scan.o: scan.l
```

For simple cases, the above may be sufficient. For other cases, you may have to explicitly instruct `make` how to build your scanner. The following is an example of a Makefile containing explicit rules:

```
# Basic Makefile -- provides explicit rules
# Creates "myprogram" from "scan.l" and "myprogram.c"
#
LEX=flex
myprogram: scan.o myprogram.o
        $(CC) -o $@  $(LDFLAGS) $^

myprogram.o: myprogram.c
        $(CC) $(CPPFLAGS) $(CFLAGS) -o $@ -c $^

scan.o: scan.c
        $(CC) $(CPPFLAGS) $(CFLAGS) -o $@ -c $^
```

```
     scan.c: scan.l
             $(LEX) $(LFLAGS) -o $@ $^

     clean:
             $(RM) *.o scan.c
```

Notice in the above example that `scan.c` is in the `clean` target. This is because we consider the file `scan.c` to be an intermediate file.

Finally, we provide a realistic example of a `flex` scanner used with a `bison` parser[3]. There is a tricky problem we have to deal with. Since a `flex` scanner will typically include a header file (e.g., `y.tab.h`) generated by the parser, we need to be sure that the header file is generated BEFORE the scanner is compiled. We handle this case in the following example:

```
     # Makefile example -- scanner and parser.
     # Creates "myprogram" from "scan.l", "parse.y", and "myprogram.c"
     #
     LEX     = flex
     YACC    = bison -y
     YFLAGS  = -d
     objects = scan.o parse.o myprogram.o

     myprogram: $(objects)
     scan.o: scan.l parse.c
     parse.o: parse.y
     myprogram.o: myprogram.c
```

In the above example, notice the line,

```
     scan.o: scan.l parse.c
```

, which lists the file `parse.c` (the generated parser) as a dependency of `scan.o`. We want to ensure that the parser is created before the scanner is compiled, and the above line seems to do the trick. Feel free to experiment with your specific implementation of `make`.

For more details on writing Makefiles, see Top.

Footnotes

[1] GNU `make` and GNU `automake` are two such programs that provide implicit rules for flex-generated scanners.

[2] GNU `automake` may generate code to execute flex in lex-compatible mode, or to stdout. If this is not what you want, then you should provide an explicit rule in your Makefile.am

[3] This example also applies to yacc parsers.