Details on the PL/0 Lexer
(`$Revision: 1.27 $`)

Gary T. Leavens
Leavens@ucf.edu

October 6, 2023

1 Purpose

This document gives some details about the lexical analyzer for the PL/0 language, and building it with the `flex` lexical analyzer generator [2].

2 What to Read

You may want to refer to *The flex Manual* [2].

You might also want to read *Systems Software: Essential Concepts* [1] in which we recommend reading chapter 5 (pages 81–91).

3 Overview

The PL/0 language itself is described in the *PL/0 Manual*, which is available in the files section of webcourses. The *PL/0 Manual* defines the grammar of the language and its semantics.

The following subsections specify the interface between the Unix operating system (as on Eustis) and the lexer as a program.

3.1 Inputs

The lexer is passed a single file name as its only command line argument; this file should be the name of a (readable) text file containing the program that the lexer should produce tokens for. Note that this program file is not necessarily legal according to the grammar for PL/0, and part of the task you have is to produce appropriate error messages for lexical errors. (This homework is not about checking the context-free grammar of programs, it is only checking the lexical grammar.) For example, if the file name argument is `hw2-test1.pl0` (and both the lexer executable, `./lexer`, and the `hw2-test1.pl0` file are in the current directory), then the lexer should work on `hw2-test1.pl0` and send all its output to standard output, except for error messages, which are sent to standard error output. The follow command redirects both output streams to the file `hw2-test1.myo`:

```
./lexer hw2-test1.pl0 >hw2-test1.myo 2>&1
```

The same thing can also be accomplished using the `make` command on Unix:

```
make lexer hw2-test1.myo
```

3.2 Outputs

The lexer prints its normal output, as specified below, to standard output (`stdout`). However, all error messages (e.g., for unreadable files or illegal tokens) should be sent to standard error output (`stderr`). See subsection 4.4 for more details about error messages.

3.3 Exit Code

When the lexer finishes without any errors, it should exit with a zero error code (which indicates success on Unix). However, when the lexer encounters an error it should terminate with a non-zero exit code (which indicates failure on Unix).

3.4 Tokens

The primary function that the parser uses to obtain tokens is `yylex`, which has the following interface:

```
extern int yylex();
```

The function is automatically generated from the specifications given in the file `pl0_lexer.l`, which generates code in the files `pl0_lexer.c` and `pl0_lexer.h`.

Although the interface above says that `yylex` returns an **`int`** value, the tokens that the lexer returns are actually members of an enumerated type (`yytokentype`) defined in the provided file `pl0.tab.h`.

(The `pl0.tab.h` header file was generated by the (context-free) parser generator `bison`, and helps define the communication between the parser and lexer.)

4 Output

The output consists of a listing of the tokens read from the input file, as printed by the `lexer_output` function shown in Figure 1. This function, along with the others defined in `lexer.h` should be put into the "user code" section of the flex input file `pl0_lexer.l` for this project. This code is also found in the file `pl0_lexer_user_code.c`, from where it can be copied into the user code section of the flex input file `pl0_lexer.l`. Some of this code must be in the user code section of the lexer, because it uses some variable names that are only available in the generated lexer (especially those whose names start with the characters "`yy`" such as `yytext`, `yylval`, and `yyin`).

in the `hw2-tests.zip` file in the course homeworks directory.

4.1 A Simple Example

Consider the input in the file `hw2-errtest1.pl0`, (note that the suffix is lowercase 'P', lowercase 'L', and the numeral zero, i.e., '0') shown in Figure 2, which is included in the `hw2-tests.zip` file in the files section of webcourses.

This is expected to produce the output found in the Figure 3 (this is in the provided file `hw2-errtest1.out`).

4.2 Provided Driver

We provide a driver (which is in the provided file `pl0_lexer_user_code.c` and should be copied into your `pl0_lexer.l` file) to run tests with the proper output formatting. After your program opens the given file in the lexer, by calling `lexer_init`, your program should call `lexer_output` to run the lexer. The `lexer_output` function asks for each token (in the lexer's file), until the lexer reaches an end-of-file, and it also prints the tokens in a standard format.

```
/* Read all the tokens from the input file
 * and print each token on standard output
 * using the format in lexer_print_token */
void lexer_output()
{
    lexer_print_output_header();
    AST dummy;
    yytoken_kind_t t;
    do {
        t = yylex(&dummy);
        if (t == YYEOF) {
            break;
        }
        lexer_print_token(t, yylineno, yytext);
    } while (t != YYEOF);
}
```

Figure 1: The provided function `lexer_output` that prints tokens from the input file. This code can be copied from the file `pl0_lexer_user_code.c`, which is included with the `hw2-tests.zip` file.

```
# $Id: hw2-errtest1.pl0,v 1.2 2023/10/06 08:18:45 leavens Exp $
what if we try some errors?
```

Figure 2: The lexer test file `hw2-errtest1.pl0`.

More extensive tests are found in the files named `hw2-test*.pl0` and `hw2-errtest*.pl0`, where `*` is replaced by a number (or letter). The expected output of each test is found in a file named the same as the test input but with the suffix `.out`. For example, the expected output of `hw2-test3.pl0` is in the file `hw2-test3.out`.

You can check your own lexer by running the tests using the Unix command on Eustis:

```
make check-outputs
```

Running the above command will generate files with the suffix `.myo`; for example your output from test `hw2-test3.pl0` will be put into `hw2-test3.myo`.

```
Tokens from file hw2-errtest1.pl0
Number Line  Text
258    2    "what"
275    2    "if"
258    2    "we"
258    2    "try"
258    2    "some"
258    2    "errors"
hw2-errtest1.pl0:2: invalid character: '?' ('\077')
```

Figure 3: Expected lexer output on stdout and stderr from running the lexer on `hw2-errtest1.pl0`, which produced the file shown (which is `hw2-errtest1.out`).

4.3 Do Not Change the Provided Files

You must not change any of the provided `.h` or `.c` files. You must use the provided files in your program.

4.4 Errors that Must be Detected

Your code must detect the following errors:

1. A number's value that is too large to be contained in a **`int`** variable (use `INT_MAX` from the standard header file `limits.h` to determine if the number's value is outside the range that the implementation of C allows).
2. A character in the input is not one of the characters permitted by the lexical grammar (i.e., the character cannot be part of a token and is not one of the recognized whitespace characters). However, note that any character is allowed inside a comment.

Lexical error messages should be sent to `stderr`; they should start with a file name, a colon, and the line number where the error occurred, followed by a colon and a space. Use the provided function `yyerror` (found in `pl0\_lexer_User_code.c`, which should be copied into your `pl0_lexer.l` file) to produce such error messages.

There are examples of programs with lexical errors in the files named `hw2-errtest*.pl0`, where `*` is replaced by a number (or letter). The expected output of each test is found in a file named the same as the test input but with the suffix `.out`. For example, the expected output of `hw2-errtest3.pl0` is in the file `hw2-test3.out`.

4.5 Checking Your Work

You can check your own lexer by running the tests using the Unix command on Eustis, which uses the `Makefile` from the `hw2-tests.zip` file in the course homeworks directory.

```
make check-outputs
```

Running the above command will generate files with the suffix `.myo`; for example your output from test `hw2-errtest3.pl0` will be put into `hw2-errtest3.myo`.

A Hints

You need to write the following files:

pl0_lexer.l, which is the specificaiton of the lexer using the flex tool [2]. You can start by creating that file by copying the provided parts into the file, putting the contents of `pl0_lexer_definitions_top.l` at the top and the contents of `pl0_lexer_user_code.c` at the bottom. This process can be automated by using the make command an the target `start-flex-file` to put the top and bottom together. Then you need to fill in any definitions you want and the regular expressions and actions in the rules section.

lexer_main.c, which is a main program that calls `lexer_init` and `lexer_output`.

We are providing several files for this homework, all of which are in the `hw2-tests.zip` file in the files section's hw2 directory on webcourses. These files include the following.

- A `Makefile` with targets `create-outputs` and `submission.zip` among others.

- `pl0_lexer_definitions_top.l` and `pl_lexer_user_code.c`, which provide the top and bottom sections of the file `pl0_lexer.l` that you need to write. You can use the make target `start-flex-file` once to put these together and begin editing `pl0_lexer.l`
- `utilities.h` and `utilities.c`, which provide the `bail_with_error` function you can use to write error messages about situations that are not lexical errors to `stderr` and then exit.
- `file_location.h` and `file_location.c`, which provide a type (`file_location` that stores a file name and line number).

References

[1] Euripides Montagne. *Systems Software: Essential Concepts*. Cognella Academic Publishing, 2021.

[2] Vern Paxson, Will Estes, and John Millaway. The flex manual. `https://westes.github.io/flex/manual/`, Oct 2016. For Flex version 2.6.2.