Alana Arjune

Brandon Bradley

USER’S GUIDE FOR PL/0

Table of Contents:

<u>USING EUSTIS</u>

When using Eustis on your Virtual Machine you must always use the terminal, or what's commonly known on your host computer as the command prompt. By now you will have already typed up your code which is ready to be uploaded into Eustis. The most important part is using the **scp** command to download files from your computer to the virtual machine. For example, the file that you want to upload is called main.c, in the terminal you would type in

```
scp main.c <yourNID>@eustis.eecs.ucf.edu:
```

This instantly puts the main.c file into Eustis. You can do the same thing for any other type of file. (i.e. text files, header files) Now what do you do with your uploaded files?

HOW TO COMPILE AND RUN IN EUSTIS

GCC stands for GNU Compiler Collection and the name gives away its use. We use the gcc command to compile our source code via the terminal. When using the gcc command we basically create an executable file, with this we can run the program. To compile we simply use this pattern

```
gcc main.c –o program1
```

main.c comes from the file that you have already uploaded in the section above.

program1 is where you now have your executable file. With this we can now run.

Running the program is easier than uploading or compiling the program as you don't have to remember any acronyms. You must simply remember "**./**" and the directory where you put the executable file. Using the same example from above, we can run the program by using this pattern:

```
./program1
```

If your program was outputting to the terminal then the program's output will show up there. If, however, your program specifically outputs a file then you can simply type **ls** into the terminal and the output file will show up in your Eustis account.

<u>HOW TO USE PL/0 COMPILER</u>

By now your compiler executable file should be running via Eustis. The source code should take in an input file and output different files (i.e. lexemelist.txt, lexemetable.txt, etc.). When you run the compiler it should also take in a flag **–l** which prints out the list of lexemes. To use this flag you simply

```
./program1 –l
```

This will print out the lexeme list. Otherwise running it normally without the **–l** will simply output the files you defined in the program and you can look them up using the **ls** command.

HOW TO USE THE PL/0 LANGUAGE

PL/0 language is an odd one. Like any other language it has grammar and patterns. The grammar of PL/0 is something called EBNF and can be shown here:

EBNF of PL/0:

```
program ::= block "." .
block ::= const-declaration  var-declaration  procedure-declaration statement.
constdeclaration ::= ["const" ident "=" number {"," ident "=" number} ";"].
var-declaration  ::= [ "int "ident {"," ident} “;"].
procedure-declaration ::= { "procedure" ident ";" block ";" }
statement   ::= [ ident ":=" expression
                | "call" ident
                | "begin" statement { ";" statement } "end"
                | "if" condition "then" statement ["else" statement]
                | "while" condition "do" statement
                | "read"
                | "write"
                | e ] .
condition ::= "odd" expression
                | expression  rel-op  expression.
rel-op ::= "="|“!="|"<"|"<="|">"|">=“.
expression ::= [ "+"|"-"] term { ("+"|"-") term}.
term ::= factor {("*"|"/") factor}.
factor ::= ident | number | "(" expression ")“.
number ::= digit {digit}.
ident ::= letter {letter | digit}.
digit ;;= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9“.
letter ::= "a" | "b" | … | "y" | "z" | "A" | "B" | ... | "Y" | "Z".
```

Based on Wirth’s definition for EBNF we have the following rule:
[ ] means an optional item.
{ } means repeat 0 or more times.
Terminal symbols are enclosed in quote marks.
A period is used to indicate the end of the definition of a syntactic class.

EBNF is a bit weird to look at and try to comprehend but it will make sense with a look into the essentials of the language. Some things of note are that PL/0 handles only integers and the virtual machine that executes the PL/0 code must be a stack based machine.

Writing PL/0 takes a while to get used to because its similar to most languages like C and Java but there are specific rules that must be followed when writing it. Some topics we will look into are constants, variables, and the read/write operators.

Constants:

Constants are like variables (which we will get into shortly) but represent a more permanent status. It is basically an expression with a fixed value and to declare it you use the const ketword.

const a = 5;

const b = 6; x = 3; z = 0;

As shown above you can declare one at a time or have multiple declarations at a time. To declare multiple constants at a time you can just use a comma to separate each const name.

Variables:

Variables are similar to constants in that you can declare a variable and assign a value to it. BUT! Variables can be changed and modified. Variables also have a different way to be written.

int x;

x := 1;

As you can see, when initializing x, we must use the **:=** symbol to make x equal to 1. We declared x as an integer by using the **int** keyword which shows that x is a variable that can be modified.

Read/Write:

The read operator is the scanf or Scanner in PL/0. it’s what you use to read in an input from the keyboard. The write operator is what you use to print out a value. This value comes from a stack where when the write operator is called, that value is popped off the stack.

Other things to note are the control structures like if then statements, loops, and procedures.

If Then Statements:

Like any other if then statement there are conditions and statements:

```
if condition then
        statement1.0
else
        statement2.0
```

Here we have an if then else statement. If the condition is true then we will do statement 1.0. If the condition is not true then we go the to else ident and do statement2.0.

Loops:

```
int x, z;
z := 0;
x := 1;
while x <= 10 do
        z = z + x;
        x = x + 1;
```

In this piece of code we first have two variable declarations and their initialization. Then we have a while loop. While x is less than 10 we do the two statements within the while loop.

Procedures:

There isn’t much to procedures. They’re basically the methods and functions of PL/0 where you can execute instructions when that procedure is called on and modify variables.