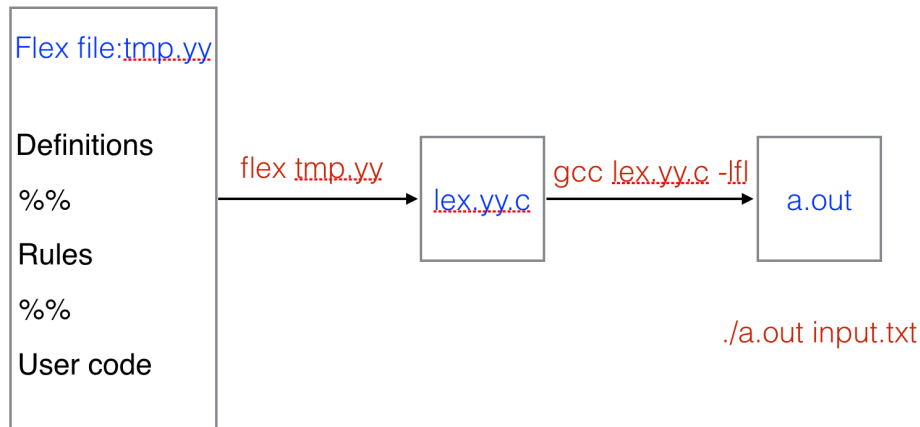


Flex

- Flex makes use of **regular expressions** to **define lexical syntax**.
- To generate a lexical analyzer, we need a .yy file (flex file). A flex file has three **three parts**: **definitions**, **rules**, and **user code**. They are separated by the double percentage sign **%%**.



- Definitions section
 - Define **macros** to be used in the rule section
 - Include C code to define **variables** used in the rule implementations
- Rules section
 - Each **lexical rule** is a **regular expression followed by C code** that expresses an action (including doing nothing) when flex finds a string matching the regular expression.
 - The **regular expression must not be indented**
 - The **code** for a rule **has to start on the same line** as the regular expression
 - Multi-line code needs to be inside a block (`{...}`), with **at least the opening curly-bracket on the same line** as the rule
 - The **rules are tested in the order** in which they **appear in the flex file**, which allows you to specify priority.
 - Text that does not match any rule is passed along to the output.
- User code section
 - Appended to the end of the C file generated by flex.
 - You can **put the main function in the user code section** of the flex file.
- If you use OSX, you probably have the flex installed on the computer already. If your computer doesn't have flex installed, you should check and make sure you have flex installed.
 - A way to check is to use command "`flex --version`"
 - You can also ssh to one of the dwarves in the robotics lab. They have flex installed.
 - Resources for flex installation can be found on the project 2 webpage.

Sample code for reading input file

- **Sample 1:** Parse an input file, print out the numbers in the input files, and display the number of lines in the file. The input file name is given via the command line.
 - To read from the command line, use the two arguments of the main function, argc and argv.
 - The sample below defines a int count and a macro DIGIT in the Definitions section. count is used to track the number of line, and DIGIT defines the digits.
 - If use macros in the Rules section, use {} to enclose the macros.
 - You can consider “yyin” as a file handler for the input file, which will be used by the yylex().
 - yylex() is a function of flex that reads input till it's exhausted.
 - yytext is a char array, which is a matched string.

```

/**
 * Read in from a specified file and
 * print out a list of all the integers in the file
 *
 * flex -o test.yy.c test.yy
 * gcc -o test test.yy.c -ll
 * ./test input.txt
 */

    int count = 0;

DIGIT    [0-9]

%%

{DIGIT}+    { printf("number: %s\n", yytext);
              /*yytext contains the matched string */
              }
\n          count++;
.           /* skip all other input */

%%
int main( int argc, char *argv[] ) {

    if (argc > 1)
        yyin = fopen( argv[1], "r" ); //where yylex reads its input

    yylex(); // a function of flex that read input till it is exhausted
    printf("There are %d lines in the file.\n", count);
    return 0;
}

```