Eredmel Specification

Kavi Gupta

July 4, 2015

1 Definitions

1. "regex" refers to a modified version of the OpenJDK `java.util.regex` package (updated to support repeated capturing groups) whose documentation is packaged with Eredmel
2. "enregex" refers to an enhanced regex library that allows for parenthesis matching and in quote / out of quote determination
3. An indentation is a sequence of whitespace that defaults to a single tab. (In all examples, I will be using four spaces to represent an indentation.)
4. The indentation level of a line is the number of indentations at the beginning of the line
5. A block is a line with an indentation level of n, followed by any number of lines with indentation levels greater than n

2 Compiler Statements

While Eredmel is interpreted, there is a preprocessor that acts as a compiler and produces a `.edc` file.

2.1 Replacement Statements

```
replace <regex>
    <replacement>
```

The use of regex inputs means that replacement macroes do not need to have matched parentheses. If there are multiple lines, each line after the first will be assigned an indentation level of (Original indentation level) $-1+$ (Regex match's first line's indentation level). The Eredmel Interpreter should go through each regex match and replace it with the given replacement text.

Replacement statements may only be placed at the beginning of an Enregex source file, and are evaluated before the rest of the file is interpreted in a top-down manner.

2.2 Import Statements

```
import <filepath>
```

The Eredmel Interpeter reads the given file and loads all Replacement Statements, Native Macro Definition Statements, and Macro Definition Statements.

2.3 Native Macro Definition Statements

```
native [<ID> ]name(<argc>)
```

The Enregex Interpreter searches the directory for a file with the same name as the `.edl` file with the extension `.java`. It then compiles and loads the method with the given name and number of arguments from the file, which must have the signature:

```
public static String <name>([String <arg1name>[, String <arg2name>[<...>]]])
```

3 State Modification Statements

3.1 Macro Definition Statements

```
define [<ID> ]<enregex>
    <replacement>
```

Macro definitions are similar to replacement statements except that they take enregexes instead of regexes. Additionally, they are not expanded at the point of definition, rather at the point of being called. Therefore, they can be expanded conditionally and defined recursively.

3.2 Variable Definition Statements

```
define [<id> ]{<varname>}
    <value>
```

The `value` is immediately expanded and then executed, and the result is associated with the name `varname`.

3.3 Macro/Variable Undefenition Statements

The syntax of an undefinition statement is as follows:

```
undefine <ID>
```

A macro undefinition is obviously contingent on the macro having been defined with an ID. The interpreter will dissasociate the given macro or variable with the given ID from the table.

4 Control Flow Statements

4.1 If statements

The syntax of an `if` statement can take one of the following forms:

```
if <condition>
    <block if so>
[else
    <block if else>]


if
    <condition>
then
    <block if so>
[else
    <block if else>]
```

The block `condition` is evaluated. If it resolves to `'true'`, then `block if so` is executed. If it resolves to `'false'`, then it expands to the `block if else` if it exists or nothing if it does not. If `condition` resolves to something else, an error is raised and the Eredmel Interpreter exits.

4.2 Do statement

```
do
    <block>
```

The block `block` is executed and the macro expands to nothing. This is useful for defining macroes that contain both logic and something to expand to; logic can be placed in a `do` statement.

4.2.1 Break statement

```
break
```

If the line `break` appears in a `do` block, the evaluation is ended and the interpreter jumps to the end of the block.

5 Macro Expansion Calls

Anything that does not match one of the above patterns is assumed to be a macro expansion call. First, all variables are replaced with their values from the lookup table. If the line does not match any stored macro, then an error will be raised. Otherwise, the macro matched will be expanded and the block will be reinterpreted.

6 Interpretation

A script in Eredmel is interpreted top-down with each block being fed into the interpreter.

6.1 Block interpreter

The interpreter will first try to match the block against any of the above statements. If it does not match any of them, it

- System functions. These are covered in greater detail below. Basically, the argument to these is fully expanded and the results are fed into the system function, evaluated in Java.
- Macro Expansion Statements. These are covered in greater detail below. These are directives to the interpreter that add or subtract macroes from the environment or conditionally expand macroes.
- Valid enregex matches.