# stklos-pp

## *a STklos Preprocessor*

Erick Gallesio

# Table of Contents

# Introduction

The command `stklos-pp` is a simple text preprocessor. It takes a text as input and produces a text as output. By itself, `stklos-pp` is language agnostic; that means that its input can be in any language (LaTeX, Markdown, Asciidoctor, ...). Embedded templates in the input file are expressed in ***STklos***, using double curly braces.

Here is as simple example, using the `stklos-pp` command:

```
$ echo "A simple computation: 2 + 3 = {{+ 2 3}}." | stklos-pp
A simple computation: 2 + 3 = 5.
```

As another example, consider the following text file named `fib.txt`

```
{{define fib  (lambda (n) (if (< n 2) n (+ (fib (- n 1)) (fib (- n 2))))))}}

The first values of Fibonacci function:
{{for-each (lambda(x) (printf "| ~5f | ~5f |\n" x (fib x)))
           '(0 1 2 3 4 5 6 7 8 9 10)}}
```

Running the command `stklos-pp fib.txt` will produce the following input:

```
The first values of Fibonacci function:
|     0 |     0 |
|     1 |     1 |
|     2 |     1 |
|     3 |     2 |
|     4 |     3 |
|     5 |     5 |
|     6 |     8 |
|     7 |    13 |
|     8 |    21 |
|     9 |    34 |
|    10 |    55 |
```

# Variables

## Variable Definition

`stklos-pp` permits to define variables in several ways. Variables are defined in the module `(stklos preproc env)`.

**Definition in Scheme**

A variable can be defined in a document using the standard Scheme `define` special form:

```
{{define foo 5}}
The square of foo is {{* foo foo}}.
```

**Definition from the Command Line**

A variable can also be defined from the command line using the `-D` option of the `stklos-pp` command. Note that, in this case, variables are defined as strings. You have to convert them if this type is not convenient:

```
$ stklos-pp -D"install-dir: /opt/foo" -D"mode: 0755" <<EOF
The install directory is {{install-dir}} and mode is {{mode}}.
The mode in decimal is {{string->number (string-append "#o" mode)}}
EOF
The install directory is /opt/foo and mode is 0755
The mode in decimal is 493
```

**Frontmatter**

Variables can also be defined in the header of a file. As command line variables, variables defined in the header are defined as strings.

Consider the file `fronmatter.txt`:

```
---
author: John Doe
title: I'm the title of the document
creation-date: Tue Jul 18 08:08:39 PM CEST 2023
---

Some information about this file:
  - Title: {{title}}
  - Author: {{author}}
  - File Creation Date: {{creation-date}}

Note that the '---' separators can be omitted.
```

Running the command `stklos-pp frontmatter.txt` will produce the following output.

```
Some information about this file:
  - Title: I'm the title of the document
  - Author: John Doe
  - File Creation Date: Tue Jul 18 08:08:39 PM CEST 2023

Note that the '---' separators can be omitted.
```

As said in the above example, the '---' separators can be omitted (in this case, let an empty line after the last variable definition).

# Special Variables

**stklos-pp-version**

When `stkos-pp` starts, there is only one variable defined: `stklos-pp-version`. This variable contains a function which returns the version of `stklos-pp` as a string.

```
STklos version: {{short-version}}
stklos-pp version: {{stklos-pp-version}}
```

produces

```
STklos version: 26.0
stklos-pp version: 0.3
```

**process-command**

The `process-command` can be used to specify the program to apply to the whole text when `stklos-pp` has built it. If you have set this variable to `pandoc` in the file `F.md`, running the command

```
$ stklos-pp F.md
```

will print the HTML output built by `pandoc` on the prepossessing of file `F.md`. Without this variable, you'll have to run something like

```
$ stklos-pp F.md | pandoc
```

**process-args**

This variable permits to collect the (eventual) arguments that must be passed to the command defined to `process-command`. For instance, if you use `pandoc`, you can define `process-args` to `"--from=markdown --to=latex"`, if your text is written in Markdown format and you want a result

expressed in LaTeX.

**template**

The `template` variable permits to specify the two template files that will be combined to build the output. If the value of `template` is `T`, the output of `stklos-pp` on file `F` will contain:

- the result of processing the file `T-prelude.templ`, followed by
- the result of processing the file `F`, followed by
- the result of processing the file `T-postlude.templ`, followed by

For instance, using the file `my-prelude.templ`

# Variables: Full Example

Here, we want to produce an HTML file from a Markdown named `pp-demo.md` expressed in Markdown (using GitHub extensions).

**File `pp-demo.md`**

```
---
title: A simple file in Markdown
author: John Doe
date: 22-Jul-2023 15:31
modified: 22-Jul-2023 17:57
template: mymd
process-command: pandoc
process-args: --from=gfm --to=html
---

## A First Section

###  A Subsection

Pellentesque dapibus suscipit ligula.  Donec posuere augue in quam.
Etiam vel tortor sodales tellus ultricies commodo.  Suspendisse
potenti.  Aenean in sem ac leo mollis blandit.  Donec neque quam,
dignissim in, mollis nec, sagittis eu, wisi.

* [ ] Cum sociis natoque penatibus et magnis dis parturient montes, nascetur
  ridiculus mus.
* [x] Phasellus purus.
* [ ] Vivamus id enim.

Praesent augue.  Fusce commodo.  Vestibulum convallis, lorem a tempus
semper, dui dui euismod elit, vitae placerat urna tortor vitae lacus.
Nullam libero mauris, consequat quis, varius et, dictum id, arcu.
Mauris mollis tincidunt felis.
```

This file states that the template is called `"mymd"`. Hence, we have to define the files `mymd-prelude.templ` and `my-postlude.templ`.

**File `mymd-prelude.templ`**

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta name="author" content="{{author}}" />
    <meta name="generator" content="stklos-pp (version {{stklos-pp-version}}" />
    <title>{{title}}</title>
  </head>
  <body>
    <div style="text-align:center">
      <h1>{{title}}</h1>
      <h2>{{author}}</h2>
    </div>
```

**File `mymd-postlude.templ`**

```
    <hr>
    <p style='text-align: right; font-size: 70%;'>
      <em>Document updated on {{modified}}</em>.
    </p>
  </body>
<html>
```

Running the command `stklos-pp pp-demo.md` will produce an HTML file. A version of the file is available here.

# Directives

`stklos-pp` directives are also delimited a double pair of braces. The list of available directives is given below:

## Directives `Include` and `Include*`

The `{{Include pathname}}` is replaced by the characters of the file named `pathname`. The `{{Include* pathname}}` is replaced by the result of pre-processing the characters of the file named `pathname`.

Suppose, we have a file named `"included.txt"` which contains

```
The square of 1234 is {{* 1234 1234}}.
```

Using `{{Include "included.txt"}}` will be replaced by the characters

```
The square of 1234 is {{* 1234 1234}}.
```

whereas `{{Include* "included.txt"}}` will be replaced by

```
The square of 1234 is 1522756.
```

## Directive `System`

The directive `{{System shell-command}}` is replaced by the characters of the standard output of `shell-command`. For instance,

```
BEGIN {{System "cd /tmp; echo -n ${PWD}"}} END
```

will produces the following output:

```
BEGIN /tmp END
```

## Directives `If`, `Else` and `End`

These directives permits to choose the piece of text that must be in the output given a (Scheme) boolean value:

```
{{If (odd? n)}}
This text is printed since {{n}} is odd.
{{Else}}
```

```
another message
{{End}}
```

Here, the first message is output when `n` is odd; otherwise, the second message is printed. Of course, the `{{Else}}` directive can be omitted when no output is not needed when the condition is false.