Huan-gîng 使用 Clochur!

Welcome to use Clóċʏɾ, a toy editor, toy intepreter and a toy typesetting-engine frontend.

Author: Yoxem Chen (aka Tan, Kian-ting) <yoxem.tem98@nctu.edu.tw>

Website: https://www.github.com/Yoxem/Clochur

1. What is Clochur?

Clochur, or printed as "Clóċʏɾ" in Irish language (*"CLOW-kur"* Clóċur as Roman type, which means "typesetting"), is a toy-lisp typesetting language with a intepreter written in Python 3, and with a simple editor written in PyQt5 and QScintilla.

It generate a XML that is readable for SILE, which is a typesetting engine written in Lua, and it generate PDF with SILE.

The functions that it has (although may be buggy or needed to be tested) is:

- Macro expansion.

- call SILE command.

- count basic arithmetic expression.

- lambda function, function definition.

1. Why it's called Clochur?

The author has (unofficially) learned Irish language (for a while and uncontinuously), so use the name.

2. How is the language? It seems that it uses brackets insteads of parathesis.

The langauge is inspired by SILE and Scheme, even though it has some different characteristics. To make the code neat and consider that parathesis is used more often than bracket, so it's more suitable to use bracket for syntatical usage.

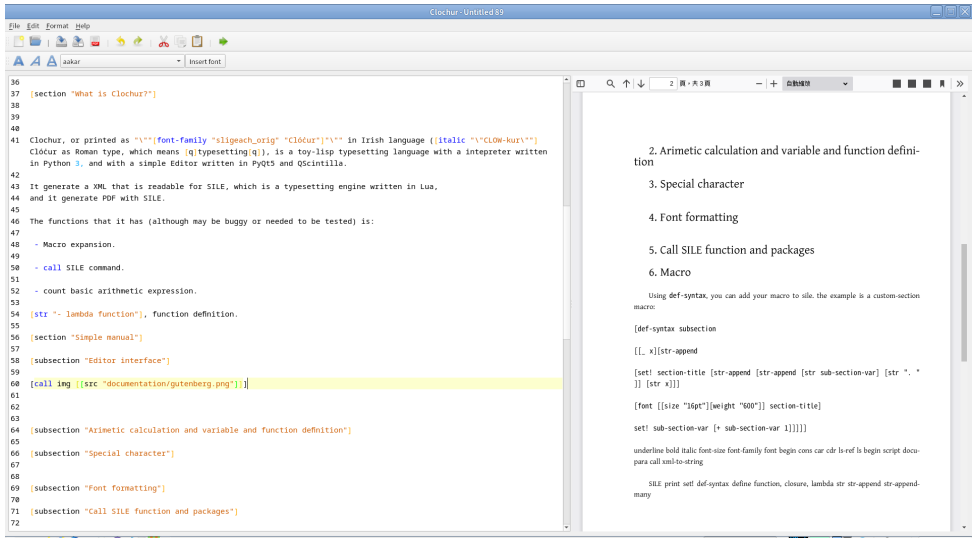
It's a toy language, so many of the function of Scheme, is not used here (for example call/cc), and there is no "let" to support local variables. However, you can call SILE function and using the packages of it with "call" and "script" respectively.

3. Does it support Taiwanese (Hokkien)/Hakka/Mandarin/Japanese/Korean or any other language that I want?

SILE supports utf-8, and Clochur will generate a XML that is readable to SILE, as long as any language that SILE can support, Clochur will support. If you find any bug, please tell me.

2. Simple manual

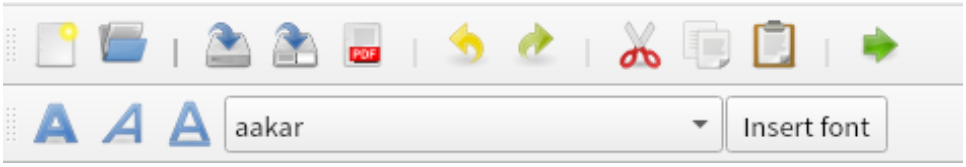
1. Editor interface



1. The interface of Clochur

Clochur is not a WYSIWYG editor, you have to type the lisp language by your self. Nevertheless, it's not a pure text-editor, it contains a PDF viewer powered by PDF.js, If you have

edited you code, you can click the green right arrow botton on the first toolbar to convert it to XML file, then generate and show PDF file from it.



1. The toolbars of Clochur

The description of the botton of the 1st toolbar is shown below (from left to right):

- Create a file (create a new window)
- Open a file
- Save a file
- Save as...
- Convert to PDF
- Redo
- Undo
- Cut
- Copy
- Paste

The description of the botton of the 2nd toolbar is shown below (from left to right):

- apply bold macro to the selected text
- apply italic macro to the selected text
- apply underline macro to the selected text
- font list
- apply the font shown in the font list to the selected text

4. Basic input, [docu], and special character

Basically, you can type the sentence that you want to type directly in sile. However, the document should be inside a macro call "docu". For example:

```
[docu Hello world!]
```

It will show

Hello world!

However, if it's a macro name (eg. docu, font, font-family, ...), you have to put them between 2 quotation marks". eg:

```
[docu "docu"]
```

It will show

docu

To modify the preferece of [docu], you should use [docu-para] before using it. Eg.

```
[docu-para [[papersize "B5"]]] [docu ...]
```

can change the paper size to B5

To type brackets [&], you should type \[& \] respectively; and to type blackslash \, you should type \\. To type quotation mark ", you should type \". A word or a sentence between 2 quotation marks (") will be shown without the quotation marks. eg: `"It will be there."`

will be shown as:

It will be there.

To make the quotation mark shown, you should enclose it between quotation mark " and using \". `"\"It will be there.\""` will be shown as:

"It will be there."

5. Arimetic calculation and variable and function definition

It can using `+ - * /` to calculate plus, minus, multiplication, and division. and print the result in the pdf. eg.

Arimetic calculation

```
The result of 3 * 6 = [* 3 6].
```

will output

The result of 3 * 6 = 18.

Define variable and change variable value

To define a variable, you should use `define`:

```
[define x [* 5 3]]
```

To change the value of a defined variable, you should use `set!`:

```
[set! x 10]
```

Define function and using lambda function

To define a function, you should using `define` and a `lambda` function. It supports currying:

```
[define add1 [lambda [x] [+ x 1]]]
[add1 7] % return 8
[[lambda [x] [+ x 2]] 9] % return 11
[define sqrt-sum [lambda [y] [lambda [x] [+ [* x x] [* y y]]]]] % currying
[call smallskip] % add line breaker
[sqrt-sum 3][call smallskip] % return closure and force to break line
[[sqrt-sum 3] 4] % return 25
[define sum [lambda [x y] [+ x y]]][call smallskip] % define multi-var function and break line
[sum 9 10]
```

```
8
11
```

<Clojure object of Clochur. vars=['x'], env=[{'y': 3}, {'section-var': 3, 'sub-section-var': 6, 'image-var': 1, 'section-title': '5. Arimetic calculation and variable and function definition', 'im-

age-desc': '1. The toolbars of Clochur', 'add1': <Clojure object of Clochur. vars=['x'], env=[{...}], body=[+ x 1]>, 'sqrt-sum': <Clojure object of Clochur. vars=['y'], env=[{...}], body=[lambda [x] [+ [* x x] [* y y]]]>}], body=[+ [* x x] [* y y]]>

25

19 `if`, `<`, `>`, `=`, `<=`, `>=` can be used to create recursive function. eg.

```
[define a [lambda [x] [if [= x 1] 2 [str-append x [a [- x 1]]]]]][call smallskip]
[a 5]
```

It returns 54321

6. Font formatting

Bacically, you can use macro `font` to set the font style. the attribute value should be quoted. eg.

```
[font [[family "Noto Serif CJK TC"]] "漢字測試。"][call smallskip] %Set the font-family to write CJK characters
%set font family and font weight from 100 (thiner), 200, ..., 900 (bolder)
[font [[family "FreeSerif"][weight "900"]] "font weight test"]
% set font style to italic and insert expression as it's argument
[font [[style "italic"]] [str [+ 9 9]]]
% set font size
[font [[size "30pt"]] "text size test"
```

will get

漢字測試。

font weight test

18

You can use the macros [bold {text}], [italic {text}], [font-family {font-family} {text}] and [font-size {font-size} {text}] to get these effects.

Using underline macro, can underline a word. eg:

[underline "underline test"]

will get

underline test

7. Call SILE function and packages

You can call sile function with call, the usage is:

[call function {[[para1 val1]{[para2 val2]}...]]} argument]

Those inside braces {} is optional.

To use SILE package, you can use macro[script source-origin].

For example, to add a url, you can add:

https://www.example.com

8. Macro

Macro is not a function, its syntax is changed before being evaluating.

Using def-syntax, you can add your macro to sile. the example is a custom-section macro:

```
[define custom-section-title ""][define custom-sub-section-var 0]
[def-syntax custom-subsection
[[_ x][str-append-many
[set! custom-section-title [str-append [str-append [str sub-section-var] [str
". "]] [str x]]]
[font [[size "16pt"][weight "600"]] custom-section-title]
[set! custom-sub-section-var [+ custom-sub-section-var 1]]]]]
```

[custom-subsection "123"] will be:

1. 123
9. Other function and macro

String appending

You can use [str-append str1 str2] to append a string to another, and you can use [str-append-many str1 str2 {str3 ...}] to append 2 or more strings. eg.

12

123

it can use to combine many commands, because the output is the combination of commands.

Print and begin

[print str] print the str to the terminal, and [begin exp1 {exp2 ...}] executes the commands and return the return value of the last command.

List

[ls item1 {item2 ...}] create a list. eg:

[ls 1 2 3 4] returns

<List object of Clochur. list=[1, 2, 3, 4]>

car get the 1st element, and cdr get the rest of the list. [list-ref list index] get the ith-index (start from 0) of list, and [cons item list] combine a item to a list. eg.

```
[define list [ls "a" "b" "c"]]
[car list] % returns "a"
[cdr list] % returns ["b" "c"]
[ls-ref list 1] % returns "b"
[cons 5 list] % returns "b"
```

will get

a <List object of Clochur. list=['"b"', '"c"']> b <List object of Clochur. list=[5, '"a"', '"b"', '"c"']>