COSCUP 2022
開源人年會

Scriptl

(A meta language that translated to JavaScript/Python/Dart)

陳鍾誠

2022 年 7 月 31 日

話說

- 我在金門大學資工系教程式

最近每年都固定教下列課程

1. 網頁設計 / 網站設計進階
2. 計算機結構 / 系統程式
3. 演算法 / 人工智慧

這些課程使用不同的程式語言

1. 網頁設計 / 網站設計進階（JS）
2. 計算機結構 / 系統程式（C）
3. 演算法 / 人工智慧 （Python）

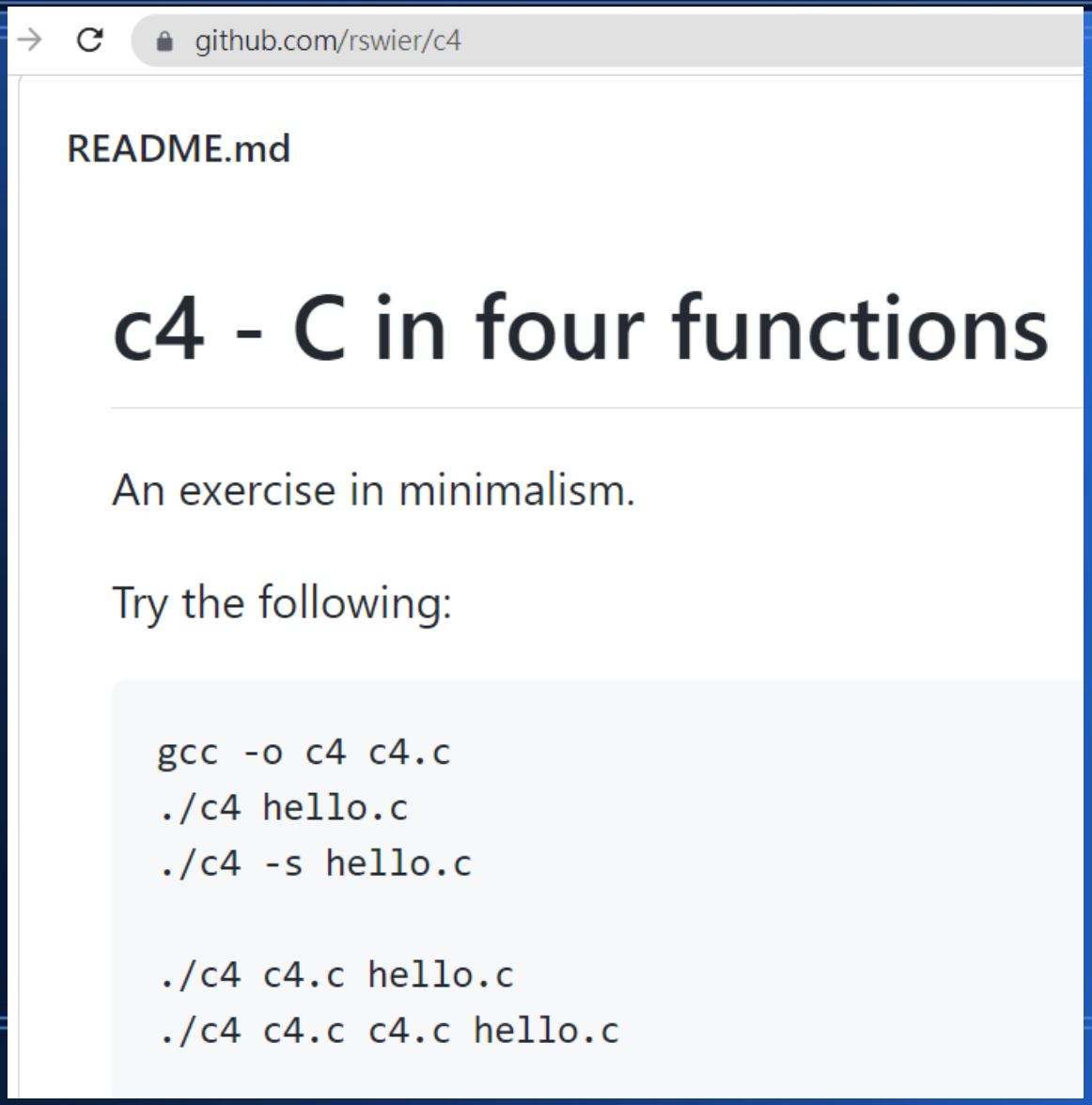
在系統程式這門課中

- 除了 C/Linux 程式設計外
- 主要的焦點放在
  - 虛擬機
  - 編譯器
  - 作業系統

在我教編譯器的過程中

- 除了教簡易的遞迴下降剖析法之外
- 也在 github 上找了一個 c4 專案
  - c4 是只有 500 行的小型 C 語言編譯器
    還附了一個虛擬機
  - 然後還有人幫他加上了 JIT 即時編譯技術

我很喜歡像C4這樣具體而微的小專案



於是我 fork 後寫了文件

github.com/ccc-c/c4/wiki

簡介 -- C4 編譯器

C4 是 Robert Swierczek 寫的一個小型 C 語言編譯器，全部 527 行的原始碼都在 c4.c 裏。

C4 編譯完成後，會產生一種《堆疊機機器碼》放在記憶體內，然後 虛擬機 會立刻執行該機器碼。

以下是 C4 編譯器的用法，C4 可以進行《自我編譯》：

```
gcc -o c4 c4.c  (you may need the -m32 option on 64bit machines)
./c4 hello.c
./c4 -s hello.c

./c4 c4.c hello.c
./c4 c4.c c4.c hello.c
```

C4 在 Windows / Linux / MAC 中都可以執行，使用的完全是標準 C 語言語法!

Pages 4

C4 編譯器

- 簡介
- 使用方式
- 支援語法
- 虛擬機
- 源碼說明

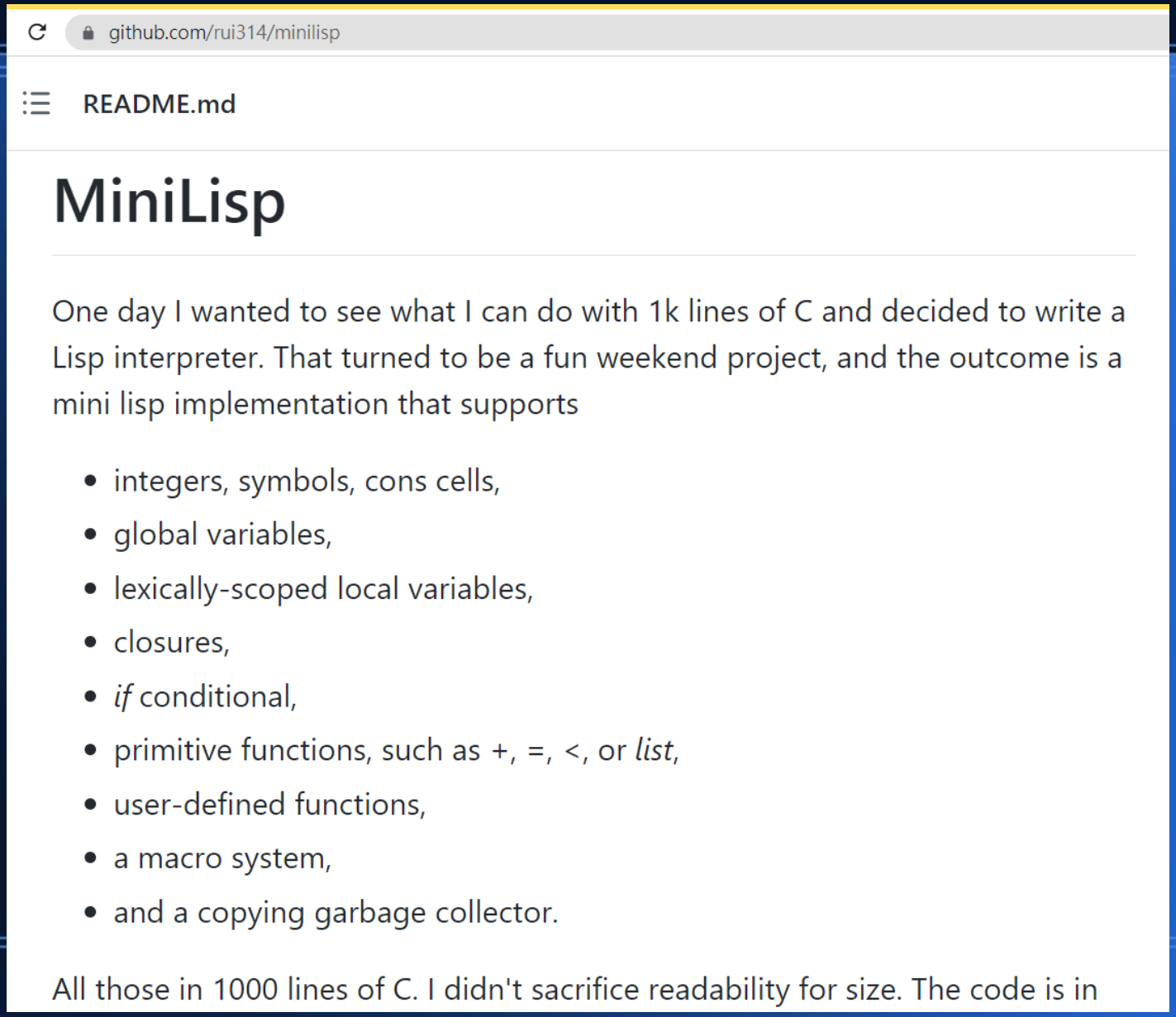
Clone this wiki locally

https://github.com/ccc-c/c4.wiki.g

後來我又看到minilisp這個專案

- 用1000行實作了一個lisp解譯器

還包含了垃圾蒐集 gc 機制



於是我又 fork 並加了註解

github.com/ccc-c/minilisp/blob/master/ccc/minilisp.c

```
622         Symbols = Nil;
623         void *root = NULL;
624         DEFINE2(env, expr);
625         *env = make_env(root, &Nil, &Nil); // 創建 root 環境
626         define_constants(root, env);  // 定義常數
627         define_primitives(root, env); // 定義基本運算
628
629         // The main loop // 主迴圈
630         for (;;) {
631             *expr = read_expr(root); // 讀取一個運算式 (...)
632             if (!*expr) // 沒有運算式了, 離開!
633                 return 0;
634             if (*expr == Cparen)
635                 error("Stray close parenthesis"); // 括號位置錯誤 )
636             if (*expr == Dot) // 點 . 位置錯誤
637                 error("Stray dot");
638             print(eval(root, env, expr)); // 執行該運算式
639             printf("\n");
640         }
641     }
```

於是我想

- 能不能自己寫一個具有高階虛擬機的編譯器
- 這個虛擬機要能支援 closure
- 而且在對變數指定型態後，能跑得和 C/C++ 一樣快

但是

- 這個嘗試一開始失敗了！
- 我沒辦法做得很好
- 彈性和速度我無法兼顧

後來我想

- 那就先照顧彈性就好
- 專心處理動態 Script 語言
- 先別管速度了！

於是我制定了一組語法

Script1 -- EBNF

```
program = stmts
stmts = stmt*
block = { stmts }
stmt = block                        |
       import str as id             |
       function                     |
       class                        |
       while expr stmt              |
       if expr stmt (else stmt)?    |
       for id in expr stmt          |
       try stmt catch expr stmt     |
       throw expr                   |
       (return|?) expr              |
       continue                     |
       break                        |
       assign
term = (await|new)? pid ( [expr] | . id | args )*
factor = (!-~) factor | Num | ( expr ) | term
map = { (str:expr)* }
item = Str | array | lambda | map | factor
bexpr = item (op2 expr)?
expr = bexpr (? expr : expr)
type = (id | str)?
field=id(:type)?
assign = (term|field) (= expr)?
param = field=expr
params = param*
term = (await|new)? pid ( [expr] | . id | args )*
function = async? fn(:id)? id(params) block
class = class id extends id { field* function* }
array = [ expr* ]
map = { (str:expr)* }
pid = (@|$)? id
num : [0-9]+(.[0-9]*)?
str : '.*'
id  : [a-zA-Z_][a-zA-Z_0-9]*
```

寫了一些範例程式

```
e2c:={'dog':'狗', 'cat':'貓', 'a':'一隻', 'the':'這隻', 'chase':'追', 'bite':'吃'}

fn:List translate(ewords) {
  cwords:=[]
  for e in ewords {
    cwords = push(cwords, e2c[e])
  }
  return cwords
}

c := translate(['a','dog','chase','a','cat'])
log(c)
```

接著，因為偷懶

- 所以在虛擬機未完成前
- 我先將 sl 轉成 JavaScript
  並用 deno 去執行

於是我創造了 Scriptl 這個語言

- 並透過 js/deno 走了捷徑，創造了一套執行環境

run

```
$ ./s1 -m prog/mt.s1 -o out/mt.js
$ deno run out/mt.js
[ "一隻", "狗", "追", "一隻", "貓" ]
```

後來我想

- 既然可以轉 JavaScript
  那應該也能轉 Python
  甚至是轉為 Dart

像是這樣

```
$ ./s1 -m hello.s1 -o hello.js
$ deno run hello.js
hello script1
```

```
$ ./s1 -m hello.s1 -o hello.py
$ python hello.py
hello script1
```

```
$ ./s1 -m hello.s1 -o hello.dart
$ dart hello.dart
hello script1
```

於是我的學生劉立行

- 把轉 Python 的部分做完了

而我自己

- 又把轉 Dart 的部分做完了

這樣

- Scriptl 的程式就能轉 JS/Python/Dart 了

舉例而言

- 我寫了一個 mt.sl 的程式

```
e2c:={'dog':'狗', 'cat':'貓', 'a':'一隻', 'the':'這隻', 'chase':'追', 'bite':'吃'}

fn:List translate(ewords) {
    cwords:=[]
    for e in ewords {
        cwords = push(cwords, e2c[e])
    }
    return cwords
}

c := translate(['a','dog','chase','a','cat'])
log(c)
```

於是利用 s1 這個轉換器

- 做以下的轉換動作，產生的檔案就能被對應的環境執行

```
$ ./s1 -m prog/mt.s1 -o out/mt.js
$ deno run -A out/mt.js
[ "一隻", "狗", "追", "一隻", "貓" ]
$ ./s1 -m prog/mt.s1 -o out/mt.py
$ python out/mt.py
['一隻', '狗', '追', '一隻', '貓']
$ ./s1 -m prog/mt.s1 -o out/mt.dart
$ dart run out/mt.dart
[一隻, 狗, 追, 一隻, 貓]
```

轉換出來的程式碼

- 會被標註原始檔來源
- 每一行後面會標上對應的行號

```
// source file: prog/mt.s1
import '../sys/s1.js'

let e2c={'dog':'狗','cat':'貓','a':'一隻','the':'這隻','chase':'追','bite':'吃'}; // (1)
export function translate(ewords)                    // (3)
{                                                    // (3)
    let cwords=[];                                   // (4)
    for (let e of ewords)                            // (5)
    {                                                // (5)
        cwords=push(cwords,e2c[e]);                  // (6)
    };                                               // (5)
```

JavaScript 版

```
// source file: prog/mt.s1
import '../sys/s1.js'

let e2c={'dog':'狗','cat':'貓','a':'一隻','the':'這隻','chase':'追','bite':'吃'}; // (1)
export function translate(ewords)                    // (3)
{                                                    // (3)
    let cwords=[];                                   // (4)
    for (let e of ewords)                            // (5)
    {                                                // (5)
        cwords=push(cwords,e2c[e]);                  // (6)
    };                                               // (5)
                                                     // (5)
    return cwords;                                   // (8)
}                                                    // (3)
let c=translate(['a','dog','chase','a','cat']);      // (11)
log(c);                                              // (12)
if (typeof main == 'function') main()
```

Python 版

```
# source file: prog/mt.s1
import sys
import os
sys.path.append('sys')
from s1 import *

e2c={'dog':'狗','cat':'貓','a':'一隻','the':'這隻','chase':'追','bite':'吃'} # (1)
def translate(ewords):                                                      # (3)
    cwords=[]                                                               # (4)
    for e in ewords:                                                        # (5)
        cwords=push(cwords,e2c[e])                                          # (6)
                                                                            # (5)
                                                                            # (5)
    return cwords                                                           # (8)
                                                                            # (3)
c=translate(['a','dog','chase','a','cat'])                                  # (11)
log(c)                                                                      # (12)
```

Dart 版

```dart
// source file: prog/mt.s1
import 'package:script1/script1.dart';
void main() {
dynamic e2c={'dog':'狗','cat':'貓','a':'一隻','the':'這隻','chase':'追','bite':'吃'}; // (1)
List translate([ewords])                                    // (3)
{                                                           // (3)
    dynamic cwords=[];                                      // (4)
    for (var e in ewords)                                   // (5)
    {                                                       // (5)
        cwords=push(cwords,e2c[e]);                         // (6)
    };                                                      // (5)
                                                            // (5)
    return cwords;                                          // (8)
}                                                           // (3)
dynamic c=translate(['a','dog','chase','a','cat']); // (11)
log(c);                                                     // (12)
}
```

這就是 Scriptl 語言

- 還有 sl 轉換工具的現況！

我們發現

- 如果只有語法統一

  函式庫卻很分歧

  那還是不太好用

於是幫 sl 加上了一組系統函數

```
export * from './js/base.js'
export * from './js/type.js'
export * from './js/str.js'
export * from './js/array.js
export * from './js/datetime
export * from './js/math.js'
export * from './js/console.
export * from './js/json.js'
export * from './js/fs.js'
export * from './js/regexp.j
import { hash } from './js/h
import { Sqlite } from './js
import { Server } from './js
global.hash = hash
global.Sqlite = Sqlite
global.Server = Server
```

JavaScript

github.com/script-one/script1/bl

```
def map(a, f):
    return list(pymap(

def filter(a, f):
    return list(pyfilt

def reduce(a, f, init)
    return list(pyredu

# console

import logging

def log(*args):
    print(join(args))
```

Python

github.com/script-one/script1/blob/master/sys/dart/script1/lib/script1.dar

```
/// Support for doing something awesome.
///
/// More dartdocs go here.
library script1;

// export 'src/script1_base.dart';
export 'src/base.dart';
export 'src/type.dart';
export 'src/str.dart';
export 'src/array.dart';
export 'src/datetime.dart';
export 'src/math.dart';
export 'src/console.dart';
export 'src/json.dart';
export 'src/fs.dart';
export 'src/regexp.dart';
export 'dart:math';
```

Dart

這樣

- 就不用在 JavaScript 裡寫 console.log
- 在 Python 和 Dart 裡寫 print 了
- 一律統一用 log(...)

就連 for 迴圈

- 也使用統一的形式
  - for i in list
  - for i in map
  - for i in range(from, to, step)

當然

- 我們還是可以使用 deno/python/dart 原有的函式庫，甚至是第三方的擴充函式庫
- 只是這樣就不能在其他語言的環境中執行了

接著

- 我們也讓 scriptl 可以引用自己所寫的函式庫
- 這些函式庫分別被轉成 .js/.py/.dart 就能橫跨三個語言環境去被引用執行

像是這幾個用 scriptl 寫的函式庫

```
fn:double rUnif(min=0, max=1) {
    return min+random()*(max-min)
}

fn:int rInt(min, max) {
    return floor(rUnif(min, max))
}

fn rChoose(a) {
    return a[rInt(0, len(a))]
}
```

```
fn map2(a,b, f) {
  alen := len(a)
  c := array(alen)
  for i in range(0, alen) {
    c[i] = f(a[i], b[i])
  }
  return c
}

fn dot(a,b) {
  alen := len(a)
  r := 0
  for i in range(0, alen) {
    r = r + a[i]*b[i]
  }
  return r
}
```

就被以下程式給引用了

```
import './lib/rnd.s1' as R
import './lib/file.s1' as F
import './lib/guid.s1' as G
import './lib/calculus.s1' as C
import './lib/vector.s1' as V
import './lib/obj.s1' as O

async fn main() {
    log('rUnif(5,10)=', R.rUnif(5,10))
    await F.copyText('data/test.txt', 'data/test2.txt')
    log('rInt()=', R.rInt(10, 20))
    log('guid()=', G.guid())
    log('diff(sin, PI/4)=', C.df(sin, math.PI/4))
    log('dot([1,2,3], [1,1,1])=', V.dot([1,2,3],[1,1,1]))
    obj := {'name':'snoopy', 'age':3}
    obj2 := O.copy(obj)
    log('obj2=', obj2)
}
```

但是當你想單獨用某語言的強項

- 像是 deno/ js 寫網站很好用
- 我們就可以引入 oak server ，這是其他語言所沒有的

基於 oak 我們可以寫個 server 物件 然後用 scriptl 去呼叫

```
import * as oak from 'https://deno.land/x/oak/mod.ts'

export class Server extends oak.Applicatic
    constructor() {
        super()
    }
}
```

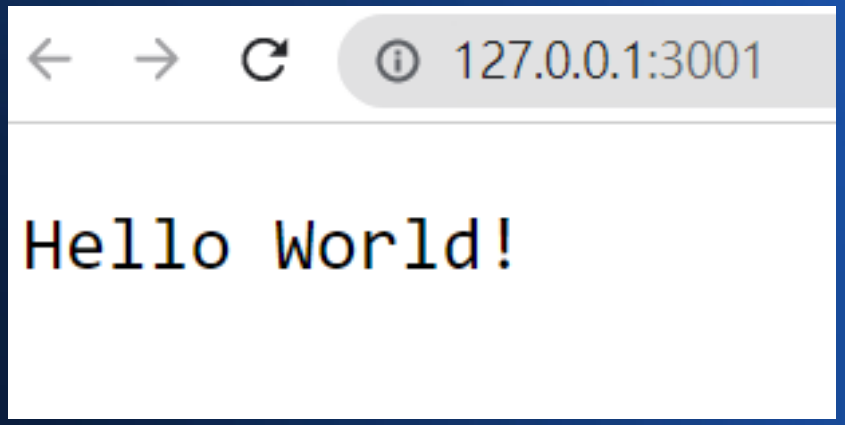
JavaScript

```
app := new Server()

fn hello(ctx) {
  ctx.response.body = 'Hello World!'
}

app.use(hello)

log('start at : http://127.0.0.1:3001')
await app.listen({ 'port': 3001 })
```

Script1

接著用 s1 轉換為 js 由 deno 執行

```
$ ./s1 -m prog/_server.s1 -o out/_server.js
$ deno run -A out/_server.js
start at : http://127.0.0.1:3001
```



這大概就是 sl 的現況了

-

不過

- 我們也還在嘗試一些新的可能性

像是把 scriptl 轉成虛擬機的組合語言

- 然後用 vm1 虛擬機執行

```
i:=1
s:=0
n:=10
while (i<=n) {
    s=s+i
    i=i+1
}
log('sum(10)=', s)
```

```
// source file: prog/while2.s1

// i:=1\n
var i    // 0
push     // 2
float 1  // 3
store    // 5
// s:=0\n
var s    // 6
push     // 8
float 0  // 9
store    // 11
// n:=10\n
var n    // 12
push     // 14
float 10     // 15
store    // 17
// while (i<=n) {\n    
get i    // 18
```

```
$ ./s1 -m prog/while2.s1 -o out/while2.ir
$ ./vm1 out/while2.ir
sum(10)=55
```

這種轉組合語言的功能

- 主要是為了系統程式的教學而設計的
- 讓學生可以觀察編譯器是如何運作的

另外我們也還在嘗試

- 讓 sl 可以將 scriptl 程式轉換成 C++
- 這樣或許就有機會讓 scriptl 跑得飛快！

因為現在的C++

- 支援了像auto這樣的自動推論型態語法
- 也支援lambda 與 class 等scriptl語言所需要的程式結構
- 下一版的sl或許會加入轉C++的功能。

雖然 S1 轉 C++ 還沒開始做

- 但我稍微設計了幾個系統函數

```
void log(auto a, string b="", string c="", string d="",
    cout << a << b << c << d << e << f << g << h;
}

vector<int> range(int from, int to, int step=1) {
    vector<int> r;
    for (int i = from; i < to; i+=step)
        r.push_back(i);
    return r;
}
```

在 C++ 新語法的加持下

- s1 轉 c++ 似乎愈來愈可行！

```
$ g++ --std=c++14 -fconcepts s1_test.cpp -o s1_test
$ ./s1_test
hello world!
2
3
4
```

不過轉換為 C++ 能否順利

- 現在還言之過早

但我自己會是 scriptl 的早期使用者

- 接下來上課時的範例程式，包含《網站設計 / 系統程式 / 演算法 / 人工智慧》
- 我都會盡量用 sl 寫，然後直接轉成 js / python / dart / c++ 給同學們閱讀。
- 這樣我就不用每個程式都寫成好幾個版本，在不同的課程使用不同的語言了。

如果有一天

- sl 足夠成熟，能讓大家輕易地用 scriptl 寫程式，然後輕鬆的佈署到 deno / python / dart+flutter / c++ 的環境上
- 那或許 scriptl 這個語言，還有 sl 工具就會有價值。

希望未來有一天

- Scriptl 也能成為一個受歡迎的程式語言！

報告完畢

Q&A　時間