Programming For Beginners With Common Lisp

unsigned_nerd

30 พฤษภาคม พ.ศ. 2563

สารบัญ

1 จุดเด่นของ Common Lisp คือความกระชับ 1

2 ความไม่ดังของ Common Lisp 3

3 Common Lisp Package, ASDF, System และ Quicklisp 4

4 การติดตั้งระบบเพื่อเริ่มเขียนโปรแกรมด้วยภาษา Common Lisp 5

1 จุดเด่นของ Common Lisp คือความกระชับ

การเขียนโปรแกรม คือ การเขียนชุดคำสั่งเพื่อสั่งให้คอมพิวเตอร์ทำงานตามที่เรากำหนด

เราเขียนชุดคำสั่งด้วย Programming Language

โดยทั่วไปหากเราพูดถึง Programming Language เราจะหมายถึงภาษาแบบ High-Level เช่น Common Lisp, Python, Go, Java, PHP, Perl, C, JavaScript เป็นต้น แต่ไม่ได้หมายถึงภาษา Assembly[1]

แต่ละภาษาฯมีจุดเด่นที่แตกต่างกัน

ภาษา Common Lisp เป็นภาษาที่ปัจจุบันไม่ได้รับความนิยมเลยทั้งๆที่มี Feature ที่น่าสนใจต่างๆมากมาย โดยเฉพาะอย่างยิ่ง Macro ใน Common Lisp ที่ทำให้เราสามารถเพิ่ม Syntax ต่างๆที่ต้องการเองได้ ทำให้เราสามารถเขียนโค้ดที่มีความกระชับและสั้นได้ง่าย[2] ตัวอย่างเช่น หากเราต้องการเขียนโปรแกรมที่ทำการอ่านข้อมูลแบบ Plain Text จาก Standard Input แล้วทำการใส่ Prefix String “Common Lisp is fun?: ” นำหน้าทุก

[1] ภาษา Assembly เป็น Low-Level Programming Language

[2] การเขียนโค้ดหากเขียนได้ยิ่งสั้นก็ถือว่ายิ่งดี โค้ดที่สั้นย่อมมีโอกาสเกิด Bug ได้น้อยกว่าโค้ดที่ยาว อีกทั้งยังสามารถอ่านให้เข้าใจได้ง่ายกว่าอีกด้วย ท่านคงเคยได้ยินคำกล่าวที่ว่า การเขียน Function ใดๆ ไม่ควรเขียนให้มีความยาวเกินความสูงของหน้าจอคอมฯของท่าน เพราะเมื่อมันยาวเกินไป สมองคนเราจะจำไม่ไหว

บรรทัด ก่อนที่จะพิมพ์ข้อมูลในแต่ละบรรทัดออกไปยัง Standard Output โดยปกติเราอาจเขียนโค้ดได้ดังนี้:

Listing 1: โค้ดแบบปกติ

```
(let (line)
  (loop for line = (read-line *standard-input* nil 'eof)
    until (eq line 'eof)
    do
      (format t "Common Lisp is fun? ~A~%" line)))
```

จะเห็นได้ว่ามันดูเข้าใจยากและมีรายละเอียดมาก ทั้ง loop, until, do, 'eof และ nil แต่หากสมมุติว่าเราเขียน Macro ใน Common Lisp เป็น แล้วเราเขียน Macro ดังนี้:

Listing 2: Macro for-each-$line-in & print-line

```
(defmacro for-each-$line-in (in-stream &rest body
  (let ((($line (intern (symbol-name '$line))))
    `(let (,$line)
       (loop for ,$line = (read-line ,in-stream nil 'eof)
         until (eq ,$line 'eof)
         do
           ,@body))))

(defmacro print-line (formatted-string &rest args)
  `(format t ,(concatenate 'string formatted-string "~%") ,@args))
```

จะทำให้เราสามารถเขียนโค้ดใหม่ได้ดังนี้:

Listing 3: โค้ดที่อ่านง่าย สบายตา ด้วยการใช้ Macro

```
(for-each-$line-in *standard-input*
  (print-line "Common Lisp is fun? ~A" $line))
```

ซึ่งสั้นกว่าเดิมมาก และอ่านเข้าใจได้ง่าย

ท่านผู้อ่านที่ยังไม่ทราบว่า Macro คืออะไรอาจเห็นโค้ดนี้แล้วเข้าใจว่าสามารถใช้การเขียน Function ทำได้เหมือนกัน ซึ่งหากลองดูดีๆจะพบว่าไม่สามารถทำได้ โดยมีจุดที่น่าสังเกตดังนี้

1. มีการสร้างตัวแปรชื่อ $line ขึ้นมาจาก Macro ชื่อ for-each-$line-in โดยโค้ดที่เรียก Macro นี้สามารถนำตัวแปร $line ไปใช้ต่อภายใน Loop ได้ด้วย เราไม่สามารถทำสิ่งนี้ได้ด้วย Function[3] สิ่งนี้ศัพท์เทคนิค เรียกว่า Anaphoric Macro และ Intentional

[3]เพราะ Function ใช้ Stack ทำให้พอ Function ทำการ Return แล้ว ตัวแปรต่างๆที่ได้ Define ไว้ใน Function นั้นๆก็จะถูกคืนหายไปหมด ในขณะที่ Macro จะทำงานตอน Compile Time

Variable Capture[4]

2. จะเห็นได้ว่า Definition ของ Macro for-each-$line-in เป็นการลอกโค้ดแบบปกติใน Listing 1 มาตรงๆ ซึ่งเป็นเรื่องที่ดีมาก เป็นการแสดงให้เห็นว่าการเขียน Macro เพื่อทำให้โค้ดกระชับแบบนี้เป็นเรื่องง่าย เมื่อเราเห็นโค้ดตรงไหนอ่านยาก ไม่อยากอ่าน เราก็ย้ายมันไปอยู่ใน Macro ได้ตรงๆเลย แต่ถ้าเราใช้ Function ทำแทนจะยุ่งยากกว่ามาก โดยเฉพาะการส่ง Source Code ผ่าน Parameter ชื่อ body จะไม่สามารถทำได้

3. Definition ของ Macro ดูเข้าใจยากและซับซ้อน แต่นั่นไม่ใช่ปัญหา เป้าประสงค์ของการใช้ Macro คือการ Encapsulate ความยุ่งยากให้อยู่ใน Macro แล้วทำให้โค้ดที่เรียกใช้ (Caller) อ่านง่าย

2 ความไม่ดังของ Common Lisp

สามารถดูการจัดอันดับภาษาคอมฯที่ได้รับความนิยมล่าสุดได้ที่นี่: http://pypl.github.io/PYPL.html

ณ วันที่ 30 พฤษภาคม 2563 อันดับหนึ่งคือภาษา Python ส่วนภาษา Common Lisp ไม่ติดอันดับบน PYPL เลย

ทาง PYPL แนะนำว่าถ้าหากอยากทราบความนิยมของภาษาที่ไม่ติดอันดับให้ใช้ Google Trends ดู

เมื่อลองใช้ Google Trends เปรียบเทียบความนิยมระหว่าง 3 ภาษา ได้แก่ Python, PHP และ Common Lisp เราจะพบว่ากระแสความนิยมในภาษา Common Lisp นั้นต่ำมาก ดังจะเห็นได้จากรูปที่ 1:

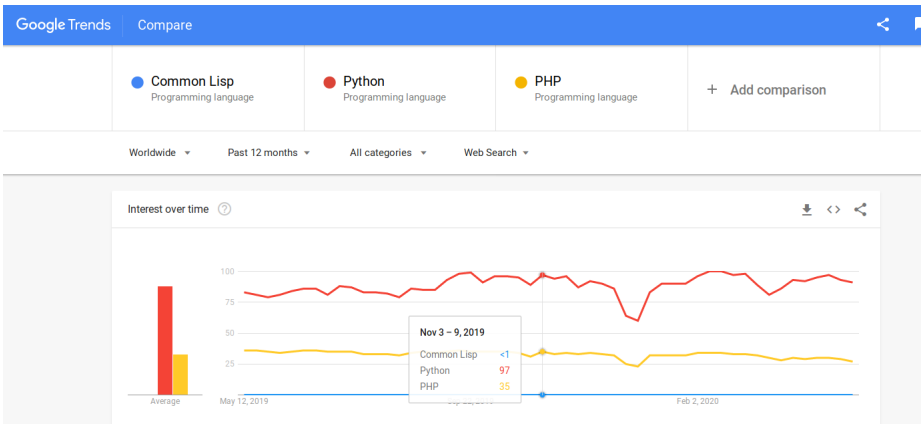
อย่างไรก็ตาม แม้ว่า Common Lisp จะไม่ได้รับความนิยมในปัจจุบัน แต่ก็ไม่ได้หมายความว่ามันจะไม่เหมาะกับการใช้เป็นเครื่องมือในการเรียนรู้การเขียนโปรแกรมสำหรับมือใหม่

เมื่อท่านได้เรียนรู้การเขียนโปรแกรมด้วยภาษา Common Lisp แล้ว ท่านจะสามารถเรียนภาษาคอมฯอื่นๆเพิ่มเติมได้ง่ายขึ้นมาก

เป็นที่ยอมรับกันโดยทั่วไปว่าในชีวิตหนึ่งๆของคนเรา เรามักต้องเรียนภาษาคอมพิวเตอร์หลายภาษากันอยู่แล้ว แล้วทำไมจึงไม่ลองเรียนภาษา Common Lisp กันเล่า!

> **รู้หรือไม่?** ภาษา Logo (เจ้าเต่าน้อย Logo) ที่หลายๆท่านอาจได้เคยเรียนในสมัยเด็กๆเป็น Dialect หนึ่งของภาษา Lisp นี่ก็ยิ่งชี้ให้เห็นว่า Common Lisp เหมาะกับการเป็นภาษาเขียนโปรแกรมแรกๆของทุกท่านขนาดไหน

[4] $line ที่อยู่ในชื่อ Function for-each-$line-in เป็นเพียงส่วนหนึ่งของชื่อ Function เท่านั้น ผู้เขียนนิยมตั้งชื่อ Macro ที่เป็น Anaphoric Macro (คือ Macro ที่มีการ Define ตัวแปรออกมาให้ Caller ใช้) โดยการเขียนชื่อตัวแปรที่ Macro สร้างขึ้นใส่ไว้ในชื่อ Function เลย (เช่นในกรณีนี้คือ $line) จะได้ดูได้ง่ายเวลาอ่านโค้ดว่าอะไรคือตัวแปรที่ Macro นั้นๆสร้างขึ้น



รูปที่ 1: Language popularity

3 Common Lisp Package, ASDF, System และ Quicklisp

เรื่อง Package, ASDF, System และ Quicklisp เป็นเรื่องหนึ่งที่ผู้เขียนมีความสับสนมากในช่วงเริ่มต้นของการเรียนภาษา Common Lisp

ความยืดหยุ่นที่มากเกินไป กับ Documentation ที่ไม่ดีนักก็ทำให้เกิดผลเสียต่อความนิยมของภาษาคอมฯหนึ่งๆได้

เรื่องนี้เกี่ยวกับระบบ Software Library ใน Common Lisp

สำหรับมือใหม่ ให้ลองทำตามนี้ก่อน จะได้ไม่สับสน แล้วเมื่อเชี่ยวชาญแล้วค่อยปรับแต่งตามใจชอบ

Software Library ต่างๆใน Common Lisp จะอยู่ในสิ่งที่เรียกว่า System

System หนึ่งๆจะประกอบไปด้วย Package ตั้งแต่ 1 อันขึ้นไป โดยจะใช้ ASDF เป็นตัวบอกว่าไฟล์ต่างๆของ Package ทั้งหลายนั้นอยู่ตรงไหน

เวลาเราจะเปิด Project ใหม่ สมมุติว่าชื่อ hello-world ให้เราไปที่ Directory ชื่อ ~/common-lisp/ แล้วสร้าง directory ชื่อ hello-world/ ขึ้นมา ซึ่งจะเป็นที่ๆเราจะใส่ Source Code ของเราในนั้น

ต่อมาเรามี Project ใหม่ สมมุติว่าชื่อ goodbye-universe เราก็สร้าง Directory ชื่อ ~/common-lisp/goodbye-universe/ แล้วใส่ Source Code ของ Project นี้ไปในนั้น

สมมุติว่าเรามี Function ใน Project ชื่อ hello-world ที่สามารถนำมาใช้ใน Project ชื่อ goodbye-universe ด้วย ทางหนึ่งที่ทำได้คือ Copy & Paste Source Code ของ Function นั้นมาใส่ แต่วิธีนี้ไม่ดี

วิธีที่ดีกว่าคือ เอา Function นั้นไปสร้างเป็น Package แล้วให้ Project ชื่อ hello-world และ goodbye-universe เรียกใช้ Function นั้นร่วมกันผ่านระบบ Package

เมื่อเราจัดเอา Function และ/หรือ โค้ดส่วนอื่นๆที่สามารถใช้ร่วมกันได้ระหว่างหลาย Project มาจัดใส่ Package ต่างๆแล้ว เราก็ต้องมาคิดต่อว่าจะต้องทำอย่างไร Sbcl จึงจะรู้ว่าไฟล์ Package ต่างๆอยู่ที่ไหน ซึ่งนี่จะเป็นหน้าที่ของ ASDF

ASDF คือเครื่องมือที่จะบอก Sbcl ว่าไฟล์ต่างๆอยู่ตรงไหน

4 การติดตั้งระบบเพื่อเริ่มเขียนโปรแกรมด้วยภาษา Common Lisp

Common Lisp มีหลาย Implementation เราเลือกใช้ Sbcl ซึ่งเป็น Common Lisp Implementation ที่ได้รับความนิยมที่สุด

ผู้เขียนใช้ Debian 10 เป็น Operating System

ทำการติดตั้ง Sbcl, Quicklisp และ un-utils บนคอมฯของท่านโดยทำตามขั้นตอนในลิงค์นี้: https://github.com/unsigned-nerd/un-utils

Sbcl

ดังได้กล่าวไปก่อนหน้านี้ Sbcl เป็น Implementation หนึ่งของ Common Lisp

เป็นโปรแกรมที่ใช้คอมไพล์และรันโปรแกรมที่เราเขียนขึ้นด้วยภาษา Common Lisp

Quicklisp

Quicklisp เป็นโปรแกรมเชิงระบบที่นิยมใช้ในการติดตั้ง Systems ต่างๆจากผู้พัฒนาคนอื่น

System ใน Common Lisp ก็คือ Library ต่างๆที่เราเรียกกันในภาษาอื่นนั่นเอง

โดยทั่วไป ให้เราคิดเสียว่า หากเราต้องการดาวน์โหลด System ของคนอื่นมาใช้ โดยคิดว่าจะใช้อย่างเดียว ไม่ได้ต้องการจะแก้ไขอะไรมัน ก็ควรจะใช้ Quicklisp ในการดาวน์โหลดและติดตั้ง

แต่ถ้าเราต้องการเขียน System เอง หรือ ต้องการแก้ไข System ของผู้อื่น ก็ให้ใช้เครื่องมือชื่อ ASDF ในการจัดการ

ASDF เป็นเครื่องมือที่ใช้ในการจัดการ

un-utils

un-utils เป็น Common Lisp System ที่ทางผู้เขียนพัฒนาขึ้นมา ซึ่งมี Package ที่น่าสนใจชื่อ un-utils.simple-syntax

un-utils.simple-syntax นี้จะ

```
1 // Hello.java
2 import javax.swing.JApplet;
3 import java.awt.Graphics;
4
5 public class Hello extends JApplet {
6     public void paintComponent(Graphics g) {
7         g.drawString("Hello, world!", 65, 95);
8     }
9 }
```

```
1 ; comment
2 (setf a 'a)
```