Welcome

CS1101S Discussion Group Week 8: *Language Processing & LEGO Programming*

Niu Yunpeng

niuyunpeng@u.nus.edu

October 10, 2017

Overview

1. Stateful programming
   - Mutable data
   - Loop
2. Language processing
   - Family of programming languages
   - From low-level to high-level
   - Compilation & interpretation
3. LEGO programming
   - History of OS and Linux
   - Using ev3dev
   - Robotics programming

Language Processing

Immutable

- Variable holds a value inside it.
- Cannot hold another value.

Mutable

- A new value can be assigned to the same variable.
- `<variable_name> = <new_value>`
- `x = 3;`

Language Processing

while loop

- With mutable data, we can make use of while loop.

Use while to compute fact(n)

```
var fact = 1;
var k = 1;

while(k < n) {
    fact = fact * k;
    k = k + 1;
}
```

Overview

1. Stateful programming
   - Mutable data
   - Loop
2. Language processing
   - Family of programming languages
   - From low-level to high-level
   - Compilation & interpretation
3. LEGO programming
   - History of OS and Linux
   - Using ev3dev
   - Robotics programming

Language Processing

What does a programming language do?

- A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output.
- Programming languages consist of instructions for a computer.
- Programming languages are used to create programs that implement specific algorithms.

Language Processing

History of programming languages

- *1940s*: ENIAC coding system
- *1950s*: Fortran, Lisp, Algol 58
- *1960s*: CPL, BASIC
- *1970s*: C, Pascal, Smalltalk, Prolog, Scheme, SQL
- *1980s*: C++, Erlang, Perl
- *1990s*: Haskell, Python, VB, Ruby, Lua, Java, JavaScript, PHP
- *2000s*: C#, .NET, F#, Go, Swift
- ...

Language Processing

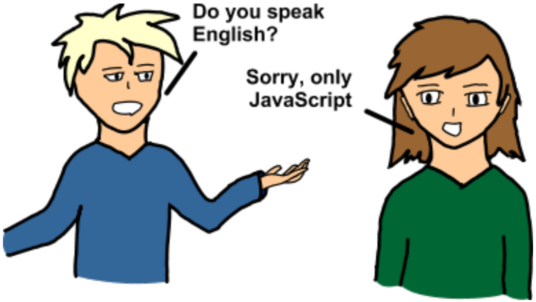
How to classify programming languages

- *According to programming paradigm*: functional, object-oriented, procedural, declarative, imperative, ...;
- *According to the way of execution*: compile, interpret;
- *According to the field of usage*: web, mobile, database, security, design, scientific calculation, ...;
- *According to typing system*: typed/untyped, static/dynamic typing, strong/weak typing, ...;
- ...

Language Processing

How does the machine understand programs?

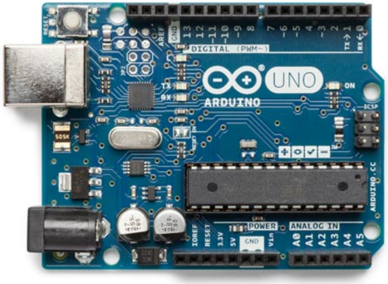
- No, computers actually does not understand the programs written by programmers.



Language Processing

What does the machine understand?

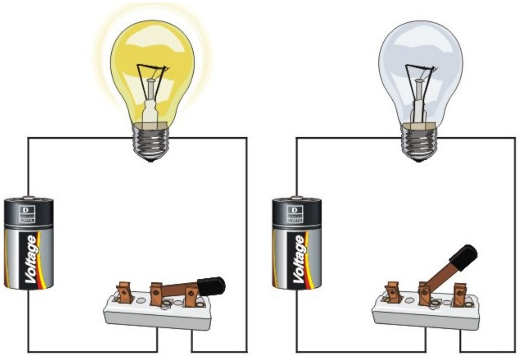
- Computers only understand byte-language (language of 0s and 1s).
- This is because computer is an electronic machine, essentially, a lot of electrical circuits.
- For each circuit, there are only 2 states: *on/off (have/no current)*.



Language Processing

What does “on/off” mean?

- They simply refer to whether the circuit has current inside, i.e., whether open circuit or not.



Language Processing

What is the work of CPU?

- Each CPU has a set of basic operations that it can perform directly.
- Machine code is a set of instructions containing these opeartions only.
- CPU can execute a program only if it is converted to machine code.

Language Processing

Machine code

- There are different kinds of machine codes, like x86/x86-64 and ARM.
- x86/x86-64 is widely used on desktops and personal computers.
- ARM is widely used on mobile devices, likesmart phones, iPad, etc.

Language Processing

Assembly language

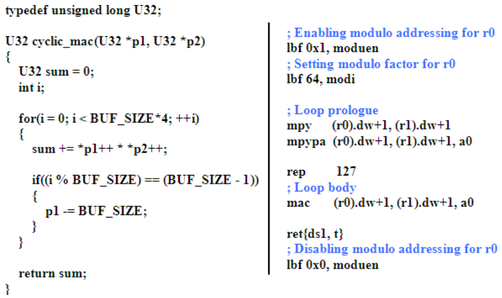
- Machine code is not human-readable.
- To make life easier, people invent **assembly languages** which use mnemonics (labels and symbols) to replace some 0s and 1s.
- Assembly code can be converted to *executable* machine code using a utility called *assembler*.

```
Machine code bytes      Assembly language statements

                        foo:
B8 22 11 00 FF          movl $0xFF001122, %eax
01 CA                   addl %ecx, %edx
31 F6                   xorl %esi, %esi
53                      pushl %ebx
8B 5C 24 04             movl 4(%esp), %ebx
8D 34 48                leal (%eax,%ecx,2), %esi
39 C3                   cmpl %eax, %ebx
72 EB                   jnae foo
C3                      retl
```

Language Processing

High-level language

- However, as you see, assembly code is still very hard to maintain.
- Therefore, people have invented more powerful languages later. They usually use some English words as syntax, like C, Java and JavaScript.
- We almost only use high-level languages nowadays.



```
typedef unsigned long U32;

U32 cyclic_mac(U32 *p1, U32 *p2)
{
   U32 sum = 0;
   int i;

   for(i = 0; i < BUF_SIZE*4; ++i)
   {
      sum += *p1++ * *p2++;

      if((i % BUF_SIZE) == (BUF_SIZE - 1))
      {
         p1 -= BUF_SIZE;
      }
   }

   return sum;
}
```

```
; Enabling modulo addressing for r0
lbf 0x1, moduen
; Setting modulo factor for r0
lbf 64, modi

; Loop prologue
mpy     (r0).dw+1, (r1).dw+1
mpypa  (r0).dw+1, (r1).dw+1, a0

rep       127
; Loop body
mac      (r0).dw+1, (r1).dw+1, a0

ret{ds1, t}
; Disabling modulo addressing for r0
lbf 0x0, moduen
```

Language Processing

The "gap" now...

- For CPU: they only understand low-level machine code;
- For programmers: they only want to write codes in high-level languages.

Language Processing

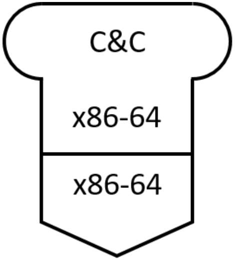
Solution

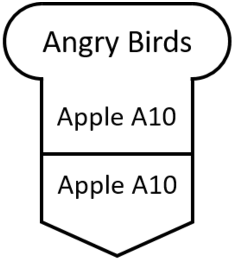
- Interpreter: a program that can execute another program written in high-level languages, like JavaScript, Python, Ruby, etc.
- Compiler: a program that translates high-level languages into low-level languages, like C/C++, Java, etc.

Language Processing

T-diagrams - direct executable

- You can directly write programs in machine code and they will be able to execute directly (although your life will be painful).

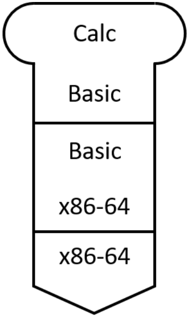


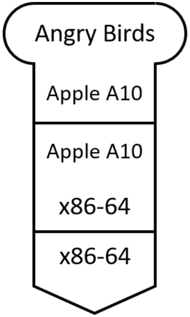


Language Processing

Use T-diagrams - interpreter

- However, in most cases, you should write programs in high-level languages and use an interpreter to execute them.



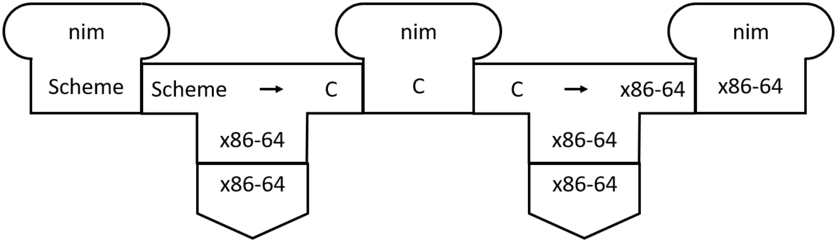


(Hardware emulation)

Language Processing

Use T-diagrams - compiler

- For some other languages, they need a compiler to translate them to low-level languages to be able to execute.
- The translation may be done in multiple steps.



(Two-stage compilation)

Language Processing

Cross platform

- Machine code may be different for different CPUs (x86/64, ARM).
- That means, the same program cannot be used across different paltforms (devices running on different hardware).
- Is it possible for the same program to run anywhere?

Language Processing

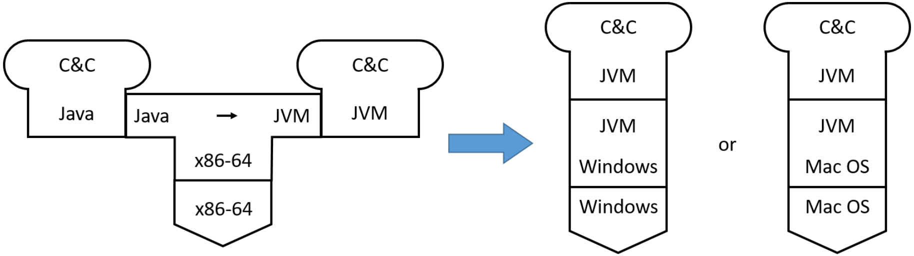
Solution - virtual machine (VM)

- We implement the same virtual machine (VM) for all platforms.
- Therefore, other programs will be able to run anywhere as long as they are converted into the “machine code” of this VM.

Language Processing

Use T-diagrams - VM

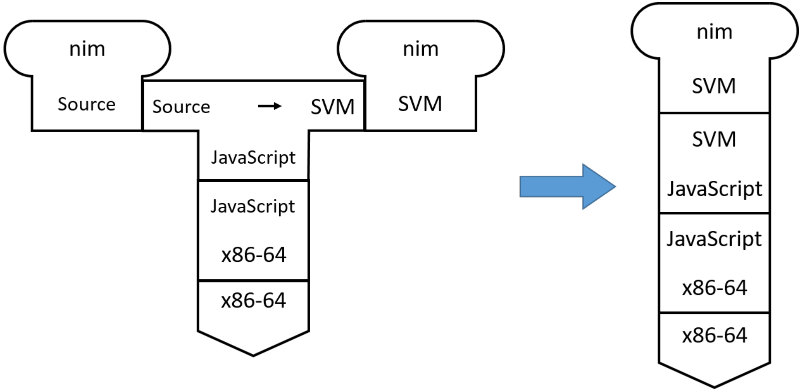
- A very famous example: Java Virtual Machine (JVM)



Language Processing

Use T-diagrams - VM

- Not that famous example: Source Virtual Machine (SVM)



Language Processing

Recommended modules at SoC

- CS2104 Programming Language Concepts
- CS4212 Compiler Design
- CS4248 Natural Language Processing
- CS6202 Advanced Topics in Programming Languages

Caution

- Conceptual-oriented;
- Abstract and theoretical.

Overview

1. Stateful programming
   - Mutable data
   - Loop
2. Language processing
   - Family of programming languages
   - From low-level to high-level
   - Compilation & interpretation
3. LEGO programming
   - History of OS and Linux
   - Using ev3dev
   - Robotics programming

Mission 13 Administration

Mission 13 Grading

- Done on this Wednesday and Thursday by Yunpeng.
- Available slots: Wednesday 20:00 - 22:00, Thursday 14:00 - 18:00.
- Venue: outside SR1, COM1, NUS

Caution

- At least one team member should be on the spot to demo.
- Submit your programs on Source Academy after demo (write down the team name and names of all teammates, including yourself).

LEGO Programming

Operating system (OS)

Maybe you are familiar with these operating systems:

- Windows
- macOS
- Android
- iOS
- ...

LEGO Programming

Operating system (OS)

But what about them:

- Unix
- Linux
- Ubuntu/Debian/CentOS...

LEGO Programming

Starting from Unix

- Unix is a pioneer OS that was first developed in 1969 at at the Bell Labs research center by Ken Thompson and Dennis Ritchie, also called *AT&T Unix*.

LEGO Programming

After that...

- Many other OSs have been inspired by Unix philosophy:
    - a set of simple tools (to each perform a limited, well-defined function)
    - a unified file system (as the main means of communication)
    - a shell scripting and command language (to combine the tools to perform complex workfows)
    - modular design.
- These OSs are called Unix-like systems, which is a family of multi-tasking, multi-user computer operating systems.

LEGO Programming

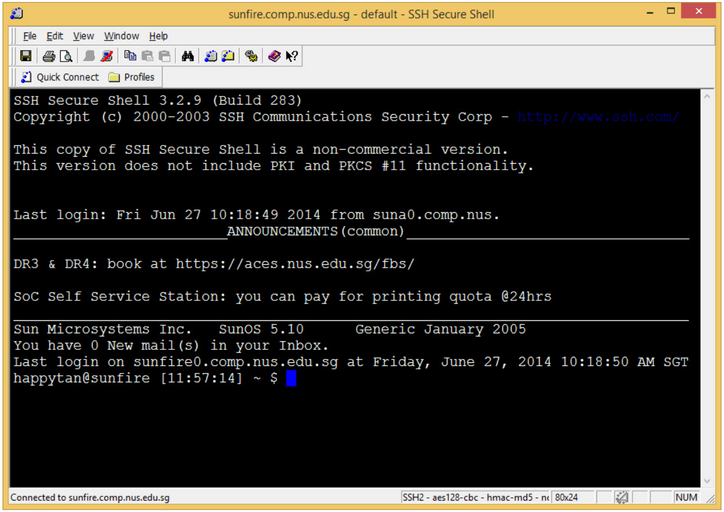
Growing up fast

- Nowadays, Unix-like OS is in fact almost everywhere.
- You may still be not aware that *macOS*, *Linux* and *Android* are all based on *AT&T Unix* and members of the Unix-like family.

Everywhere

- Due to its high performance and reliability, more than 90% of the super- computers around the world is using Unix.
- Our SoC server, **SunFire** is using *Solaris*, a Unix-like OS developed by *Sun Microsystems*.

LEGO Programming


sunfire.comp.nus.edu.sg - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles

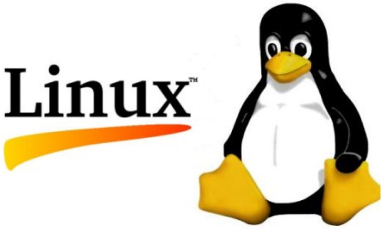
```
SSH Secure Shell 3.2.9 (Build 283)
Copyright (c) 2000-2003 SSH Communications Security Corp - http://www.ssh.com/

This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.


Last login: Fri Jun 27 10:18:49 2014 from suna0.comp.nus.
_________________________ANNOUNCEMENTS(common)______________________________

DR3 & DR4: book at https://aces.nus.edu.sg/fbs/

SoC Self Service Station: you can pay for printing quota @24hrs
____________________________________________________________________________
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
You have 0 New mail(s) in your Inbox.
Last login on sunfire0.comp.nus.edu.sg at Friday, June 27, 2014 10:18:50 AM SGT
happytan@sunfire [11:57:14] ~ $
```

Connected to sunfire.comp.nus.edu.sg
SSH2 - aes128-cbc - hmac-md5 - nc
80x24
NUM

LEGO Programming

From Unix to Linux

- Linux was developed by Linus Torvalds in 1991.
- At that time, Linus was still an undergraduate student at University of Helsinki. He was frustrated by the OS used at school then, called Minix. So, he decided to develop a better one by himself.
- If you found any system (like the printers) at Soc very hard to use, you should know why the school makes it to be like that now.



LEGO Programming

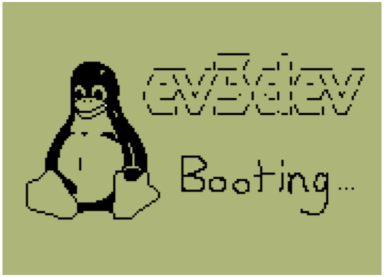
Linux's history

- However, the original Linux should be called *Linux kernel* because it usually performs as a minimum setup instead of full installation.
- Thus, Linux is usually packaged in a form known as *Linux distribution* (or *distro* for short) for both desktop and server usage.
- Some famous *Linux distros* are CentOS, Debian and Ubuntu.

LEGO Programming

Your LEGO ev3 now

- By copying the given image to the SD card, you install ***ev3dev*** for your robot.
- ***ev3dev*** is a variant of *Debian* (a famous Linux distro), which can run on several kinds of LEGO robots.
- Theoretically, you can do any legal Linux operation on ***ev3dev***.



LEGO Programming

To access your ev3dev

- Your ***ev3dev*** is not like your normal laptop OS. It is an embedded system, without monitor, keyboard or mouse.
- However, it does have CPU and memory. So, it can do any task like your normal laptop. But, you need to access it in a different way.

LEGO Programming

To access your ev3dev - use SSH

- SSH is short for *Secure Shell*, a secured method to access from local computer to a remote computer.
- For Windows: use Putty/Pietty/Kitty, OpenSSH, Xshell, etc.
- For mac and Linux: use system built-in Terminal.

LEGO Programming

Common commands in Linux

- `cd <file_name>`: changes to that selected directory;
- `cd ..`: go back to the parent directory;
- `pwd`: print the absolute path of the current directory;
- `ls`: list all files and sub-directory in the current directory;
  You may want to supply `-a` to include hidden files and `-l` to see the long format (include permission, size, timestamp, etc).
- `rm <file_name>`: remove the selected file;
- `chmod <code> <file_name>`: change the selected file's permission;
- `vim <file_name>`: use vim to edit a file.

LEGO Programming

Using vim in command-line

- Vim is a simple but powerful text editor in all platforms;
- Vim has two modes: command mode (where you can navigate and manipulate the file, press `<ESC>` to enter) and insert mode (where you edit the file, press `<i>` to enter));
- To save and exit: enter command mode, press `:wq<ENTER>`;
- You may want to modify `.vimrc` to change the vim setting (notice that common settings of this file can be found online).

LEGO Programming

Robotics programming

- Robotics programming is exciting because this may be the first time that your program can really make something real move (not on the monitor anymore).
- However, this is not going to be easy. You need to consider more.

LEGO Programming

Advice

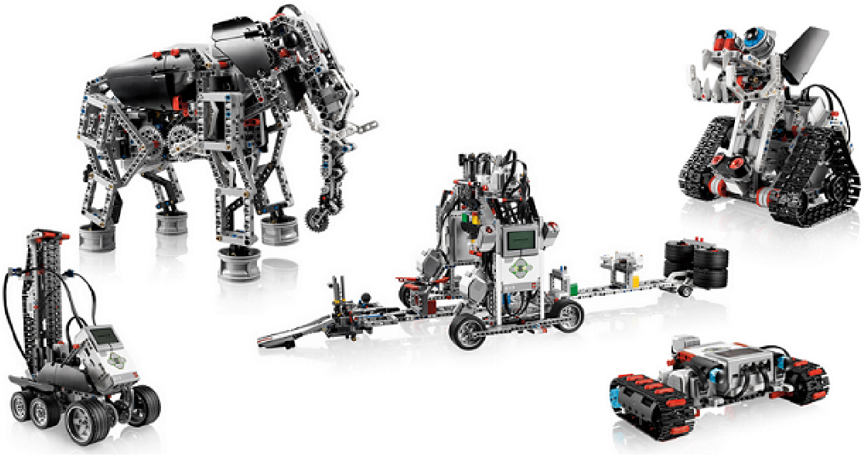
- Remember your math. Try to do some accurate calculation;
- Remember your physics. Gravity, friction, acceleration, ...;
- Remember your programming. Harder to debug this time.

LEGO Programming

A few hints

- Do modular design: each part do independent work;
- Develop your own “callback function”: keep doing checks for some conditions, whenever true, the corresponding function will be called;
- The power of the motor may change gradually as you rely on battery.

LEGO Programming



LEGO Programming

Happy developing!

End

The End