Why in heaven there is no dep manager in C/C++?

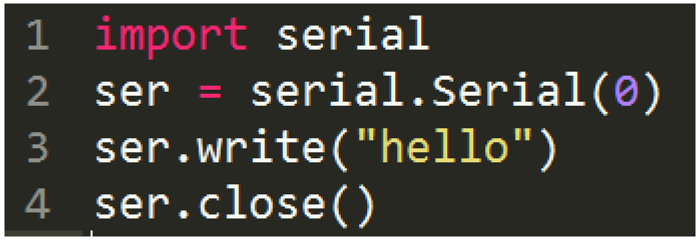
C++Now 2015

Diego Rodriguez-Losada, PhD
@diegorlosada

LETS SEE AN EXAMPLE

- Send some bytes over a serial port

```
$ pip install pyserial
```

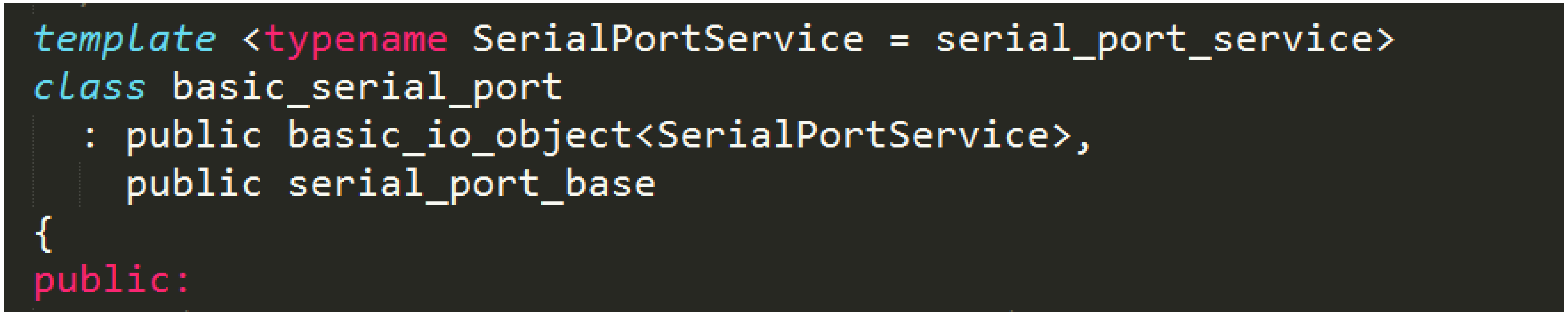


```python
import serial
ser = serial.Serial(0)
ser.write("hello")
ser.close()
```

.pypirc
setup.py
licence

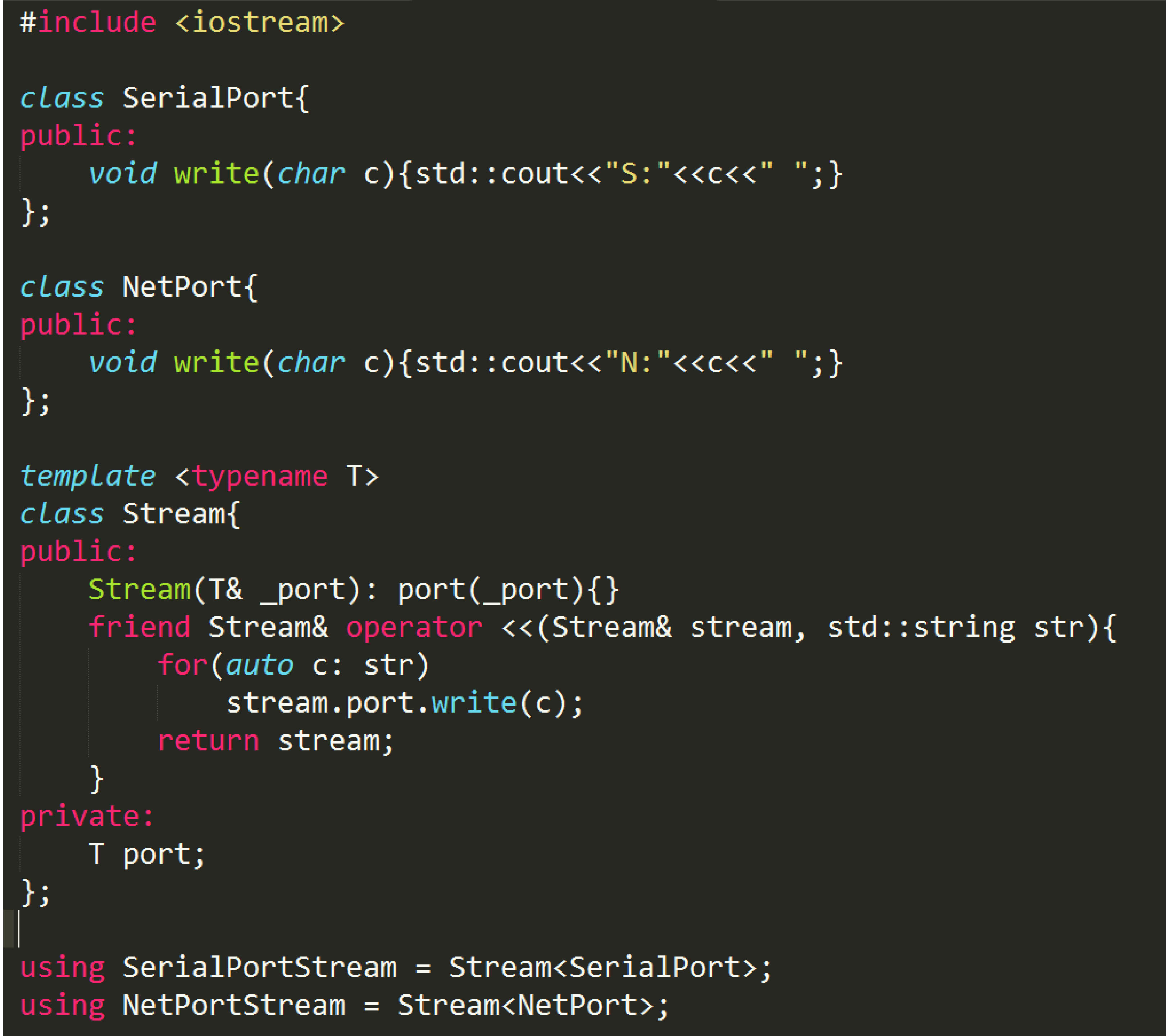
```
$ python setup.py register -r pypi
$ python setup.py sdist upload -r pypi
```

SERIAL PORT IN BOOST.ASIO



```
template <typename SerialPortService = serial_port_service>
class basic_serial_port
  : public basic_io_object<SerialPortService>,
    public serial_port_base
{
public:
```

- I have to download hundreds of Mb
  - Know that ASIO can be STANDALONE
- I have to configure (include) path, cmake FindBoost/FindPkg
- I depend on Async features



```cpp
#include <iostream>

class SerialPort{
public:
    void write(char c){std::cout<<"S:"<<c<<" ";}
};

class NetPort{
public:
    void write(char c){std::cout<<"N:"<<c<<" ";}
};

template <typename T>
class Stream{
public:
    Stream(T& _port): port(_port){}
    friend Stream& operator <<(Stream& stream, std::string str){
        for(auto c: str)
            stream.port.write(c);
        return stream;
    }
private:
    T port;
};

using SerialPortStream = Stream<SerialPort>;
using NetPortStream = Stream<NetPort>;
```

PATTERNS

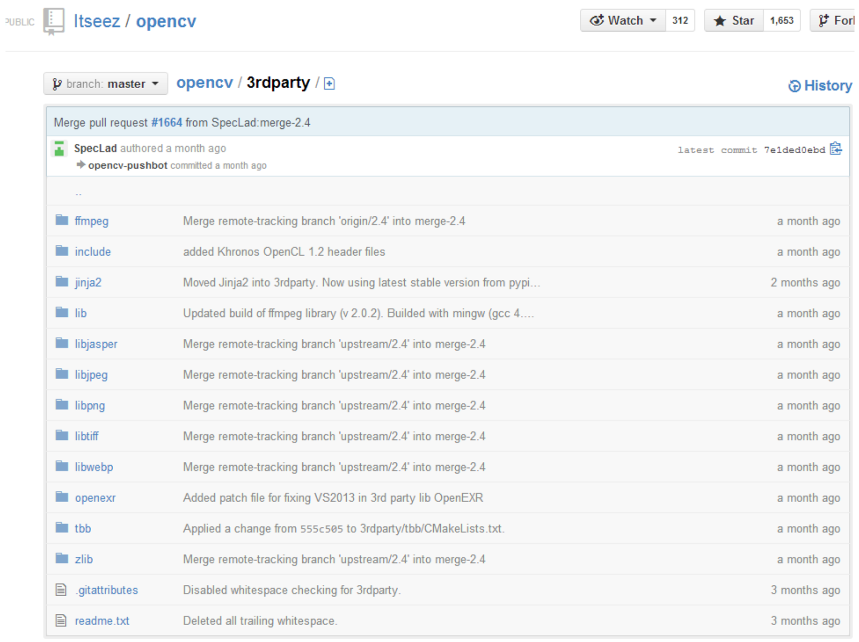
- Low coupling, separation of concerns, single responsibility
- Functionality, core value should be independent, easily isolated:
    - Easily testable, mockable
    - Easily understandable
    - Easily reusable
- P. Fultz: Asio => Asio.Net, Asio.Serial…
- Instead => HW.Serial

SERIAL IN GITHUB

- https://github.com/wjwwood/serial
- I have to:
  – Git clone
  – Build
  – Configure my project
  – Or invest more time and automate (Cmake?)

THE OPENCV CASE

- Great SW



PUBLIC Itseez / opencv — Watch 312 · Star 1,653 · Fork

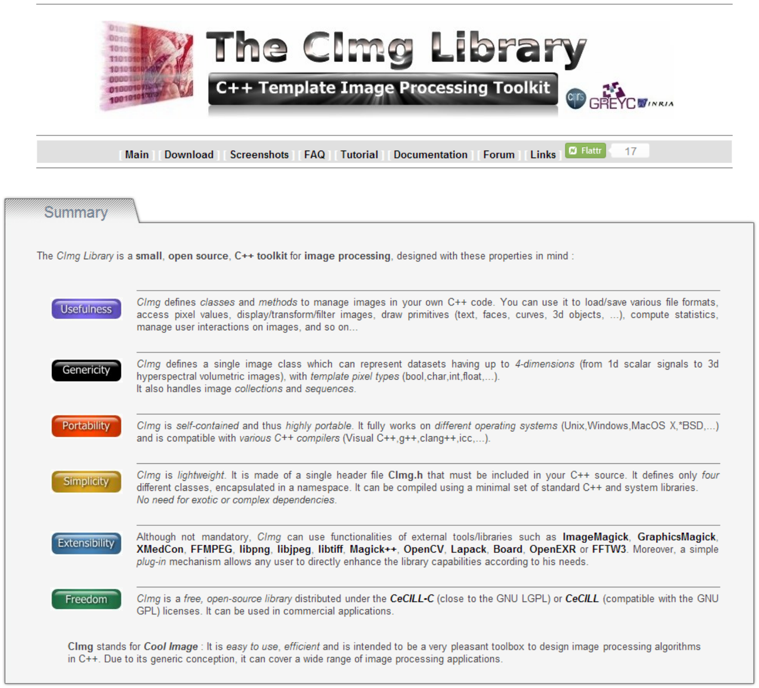
branch: master — opencv / 3rdparty / — History

Merge pull request #1664 from SpecLad:merge-2.4

SpecLad authored a month ago — latest commit 7e1ded0ebd

opencv-pushbot committed a month ago

..		
ffmpeg	Merge remote-tracking branch 'origin/2.4' into merge-2.4	a month ago
include	added Khronos OpenCL 1.2 header files	a month ago
jinja2	Moved Jinja2 into 3rdparty. Now using latest stable version from pypi…	2 months ago
lib	Updated build of ffmpeg library (v 2.0.2). Builded with mingw (gcc 4….	a month ago
libjasper	Merge remote-tracking branch 'upstream/2.4' into merge-2.4	a month ago
libjpeg	Merge remote-tracking branch 'upstream/2.4' into merge-2.4	a month ago
libpng	Merge remote-tracking branch 'upstream/2.4' into merge-2.4	a month ago
libtiff	Merge remote-tracking branch 'upstream/2.4' into merge-2.4	a month ago
libwebp	Merge remote-tracking branch 'upstream/2.4' into merge-2.4	a month ago
openexr	Added patch file for fixing VS2013 in 3rd party lib OpenEXR	a month ago
tbb	Applied a change from 555c505 to 3rdparty/tbb/CMakeLists.txt.	a month ago
zlib	Merge remote-tracking branch 'upstream/2.4' into merge-2.4	a month ago
.gitattributes	Disabled whitespace checking for 3rdparty.	3 months ago
readme.txt	Deleted all trailing whitespace.	3 months ago



The CImg Library

C++ Template Image Processing Toolkit

Main | Download | Screenshots | FAQ | Tutorial | Documentation | Forum | Links | Flattr 17

Summary

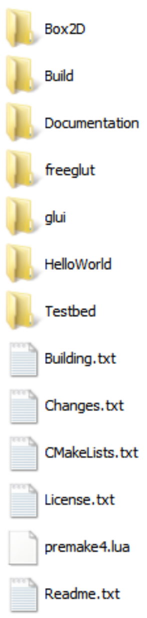
The *CImg Library* is a **small**, **open source**, **C++ toolkit** for **image processing**, designed with these properties in mind :

Usefulness	*CImg* defines *classes* and *methods* to manage images in your own C++ code. You can use it to load/save various file formats, access pixel values, display/transform/filter images, draw primitives (text, faces, curves, 3d objects, ...), compute statistics, manage user interactions on images, and so on...
Genericity	*CImg* defines a single image class which can represent datasets having up to *4-dimensions* (from 1d scalar signals to 3d hyperspectral volumetric images), with *template pixel types* (bool,char,int,float,...). It also handles image *collections* and *sequences*.
Portability	*CImg* is *self-contained* and thus *highly portable*. It fully works on *different operating systems* (Unix,Windows,MacOS X,*BSD,...) and is compatible with *various C++ compilers* (Visual C++,g++,clang++,icc,...).
Simplicity	*CImg* is *lightweight*. It is made of a single header file **CImg.h** that must be included in your C++ source. It defines only *four* different classes, encapsulated in a namespace. It can be compiled using a minimal set of standard C++ and system libraries. *No need for exotic or complex dependencies.*
Extensibility	Although not mandatory, *CImg* can use functionalities of external tools/libraries such as **ImageMagick**, **GraphicsMagick**, **XMedCon**, **FFMPEG**, **libpng**, **libjpeg**, **libtiff**, **Magick++**, **OpenCV**, **Lapack**, **Board**, **OpenEXR** or **FFTW3**. Moreover, a simple *plug-in* mechanism allows any user to directly enhance the library capabilities according to his needs.
Freedom	*CImg* is a *free, open-source library* distributed under the ***CeCILL-C*** (close to the GNU LGPL) or ***CeCILL*** (compatible with the GNU GPL) licenses. It can be used in commercial applications.

CImg stands for ***Cool Image*** : It is *easy to use*, *efficient* and is intended to be a very pleasant toolbox to design image processing algorithms in C++. Due to its generic conception, it can cover a wide range of image processing applications.

BOX2D, GTEST, CATCH

- Erin Catto



Box2D
Build
Documentation
freeglut
glui
HelloWorld
Testbed
Building.txt
Changes.txt
CMakeLists.txt
License.txt
premake4.lua
Readme.txt

STATEMENT I

- **The design, size and modularization of current C and C++ SW packages would be completely different if we had a deps manager.**

- Library in a week: Not many dependencies!

WAIT! WE ALREADY HAVE …

- Linux
  - Apt, yum, pacman
- Windows
  - NuGet
  - Chocolatey
- Mac
  - Homebrew, macports

STATE OF THE ART

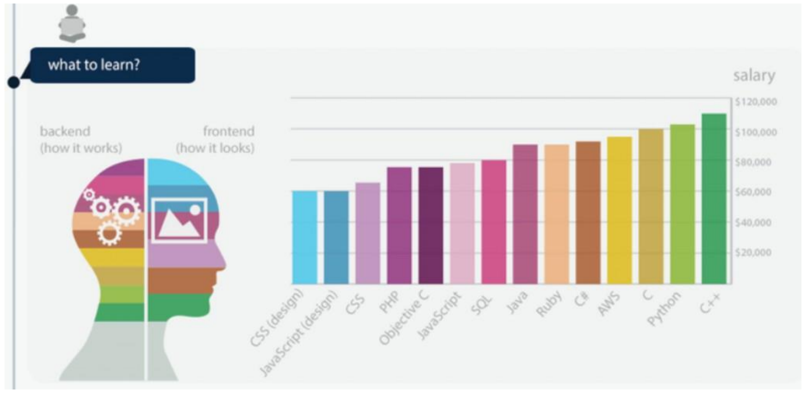
- Python taking over numeric/maths/simulation/engineering, why?
  - Numpy, numba, simple, fast enough
  - PyPy
- NPM: raised $8M
  - 2-3% of programmers

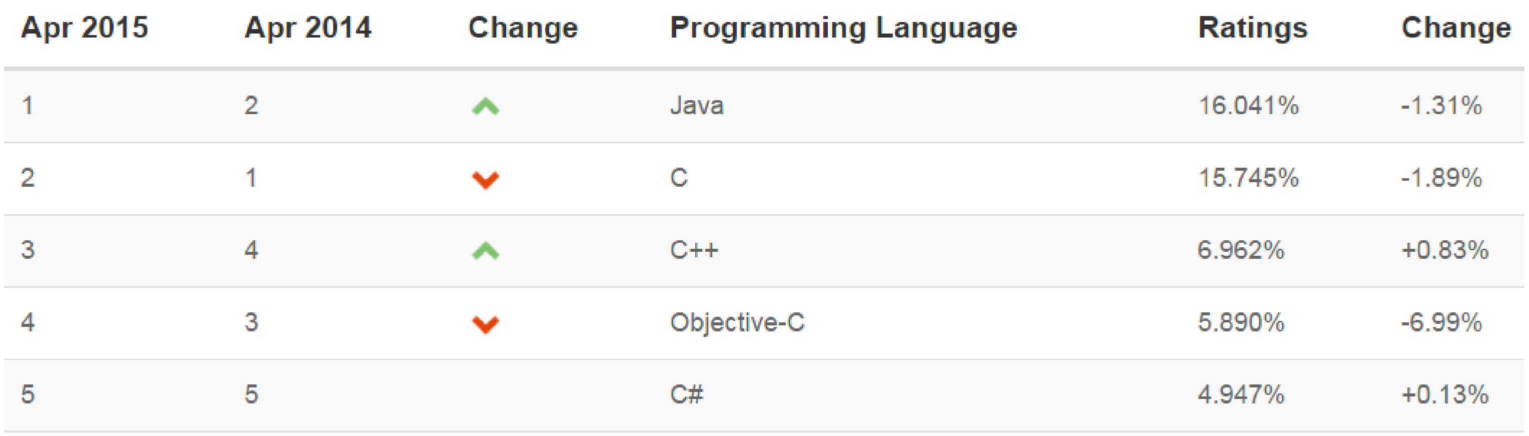
NATIVE LANGUAGES

- GO: By google, good at concurrency and server tasks
- But…
  - Go get considered harmful
  - Poor language

NOTHING TO BE AFRAID OF

- 3.9M devs C and C++
- Most paid
- C++11 2nd most loved

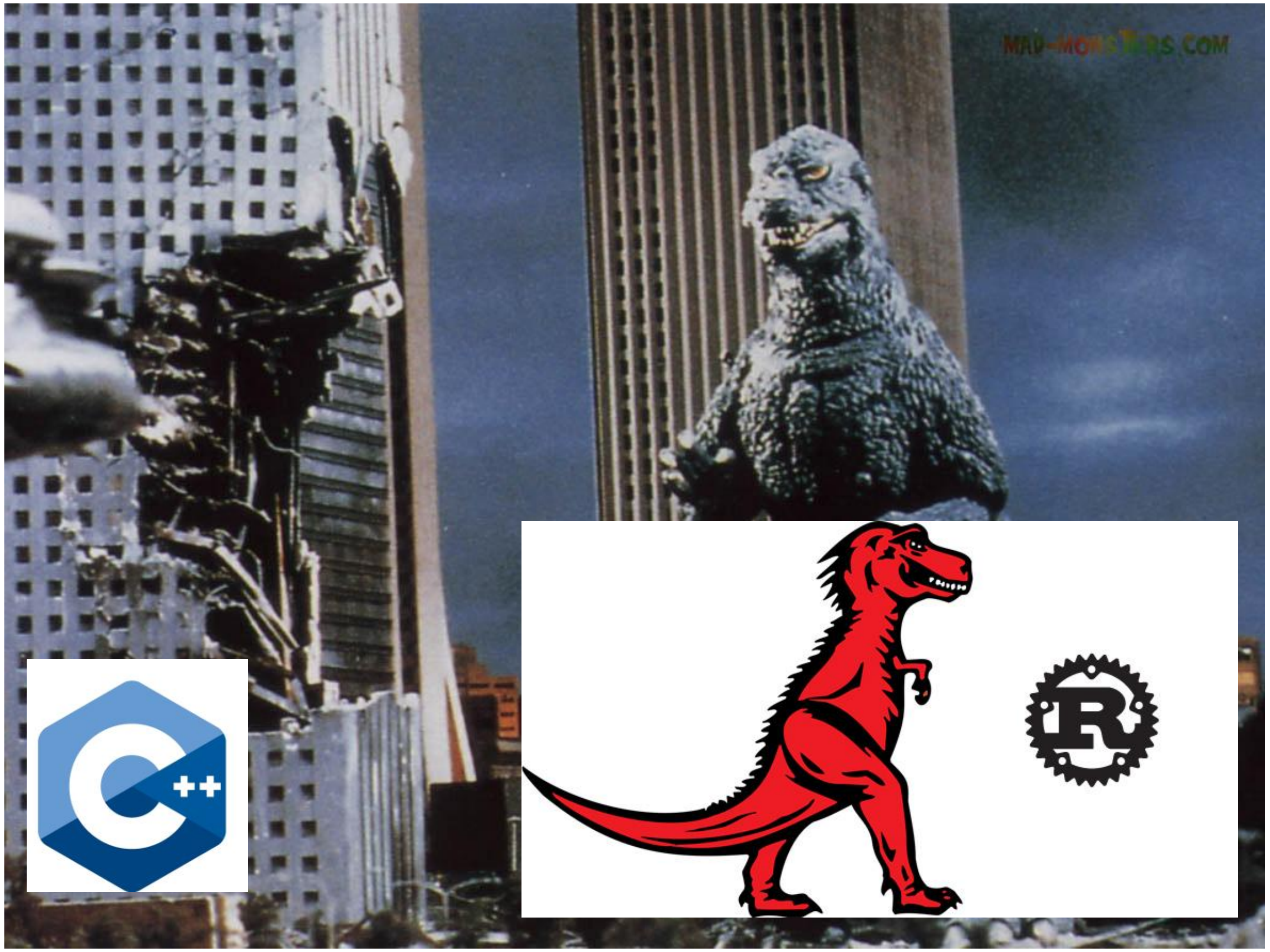




Apr 2015	Apr 2014	Change	Programming Language	Ratings	Change
1	2	▲	Java	16.041%	-1.31%
2	1	▼	C	15.745%	-1.89%
3	4	▲	C++	6.962%	+0.83%
4	3	▼	Objective-C	5.890%	-6.99%
5	5		C#	4.947%	+0.13%

1. http://www.businessinsider.com/the-programming-and-engineering-skills-with-the-highest-salaries-2015-3
2. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
3. http://techcrunch.com/2015/04/14/popular-javascript-package-manager-npm-raises-8m-launches-private-modules/

NOTHING TO BE AFRAID OF?

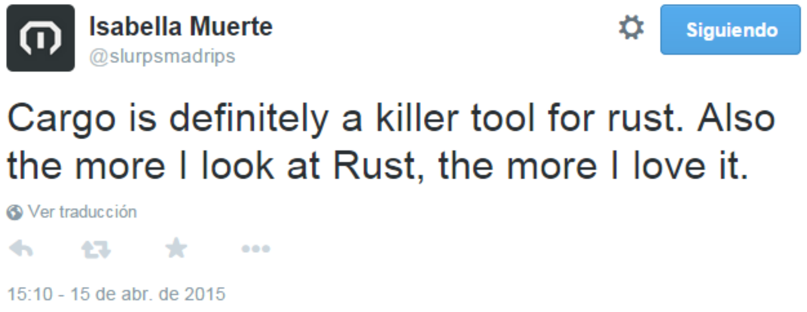


YOU ARE NOT AVG C++ CODERS

- Many C++ coders don’t know that you exist
  - They don’t care about functional programming, TMP or ranges.
- They are in C++98, full of MyClass* p = new MyClass()

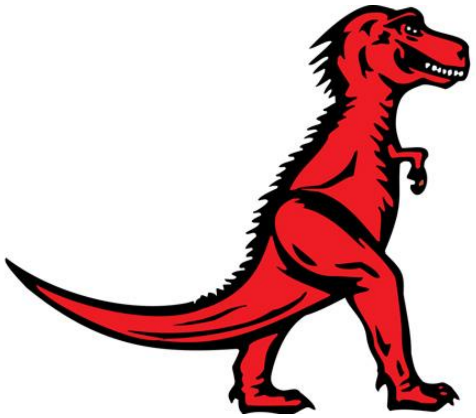
BEAST COME IN CARGO

- Safer, less time debugging
- Good performance (LLVM) 90% of performance 90% cases
- Companies will realize: decrease times, cheaper costs


Isabella Muerte
@slurpsmadrips

Cargo is definitely a killer tool for rust. Also the more I look at Rust, the more I love it.

15:10 - 15 de abr. de 2015



OK, JUST FOOD FOR THOUGHT

- Criticizing the Rust Language, and Why C/C++ Will Never Die
  - http://www.viva64.com/en/b/0324/

STATEMENT II

- A multiplatform C and C++ deps manager would be more beneficial than any other new language feature
  - Except: modules, but when they will be widely used 2020?

REQUISITES OF A DEP MANAGER

- Same in all OS/platforms
- Manage deps per-project
  - Dep override, conflict resolution
- Simple and inmediate to publish
  - With private artifacts
- From source, but also binaries
- Metrics, statistics:
  - Number of downloads/uses per library (which boost libs are obsolete/not used?)

CAN BE (TECHNICALLY) DONE?

- Ryppl
- CPMCPP
- Hunter
- FIPS
- Needs a company?
  - Not developed in C++!
  - Difficult to get python coders.

INCONVENIENCE: THE BUILD SYSTEM

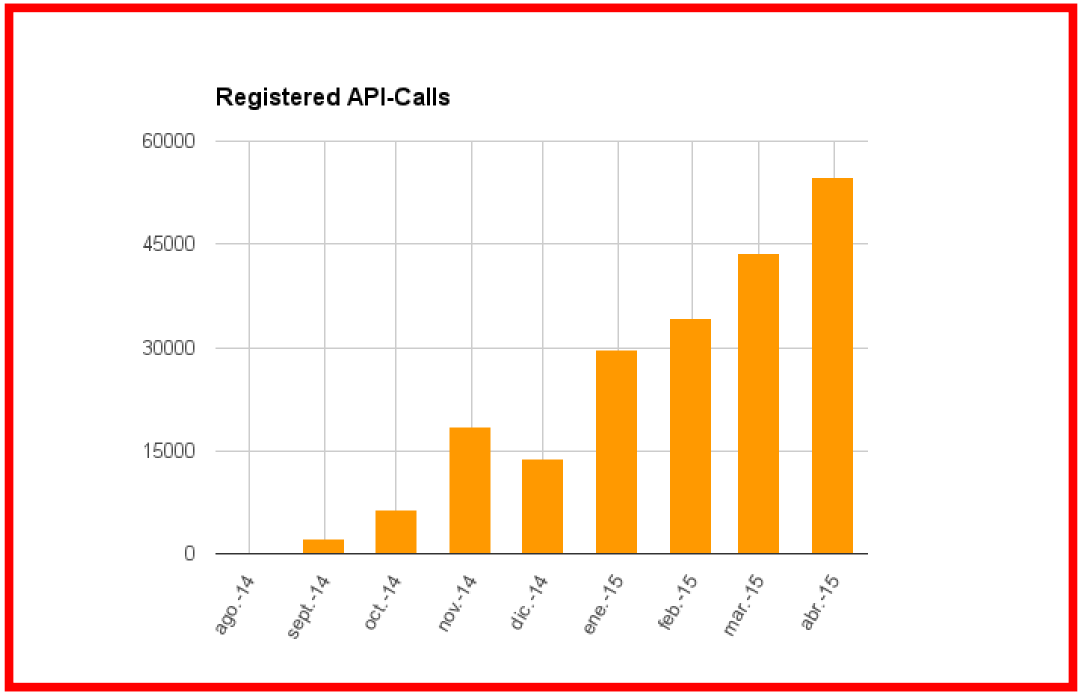
CMake
Cross-platform Make

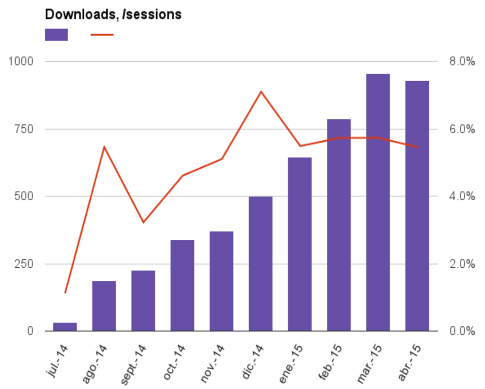
SCONS

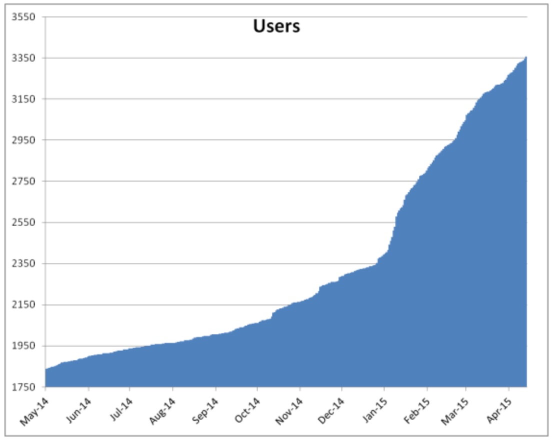
Bazel

DO WE REALLY WANT IT?

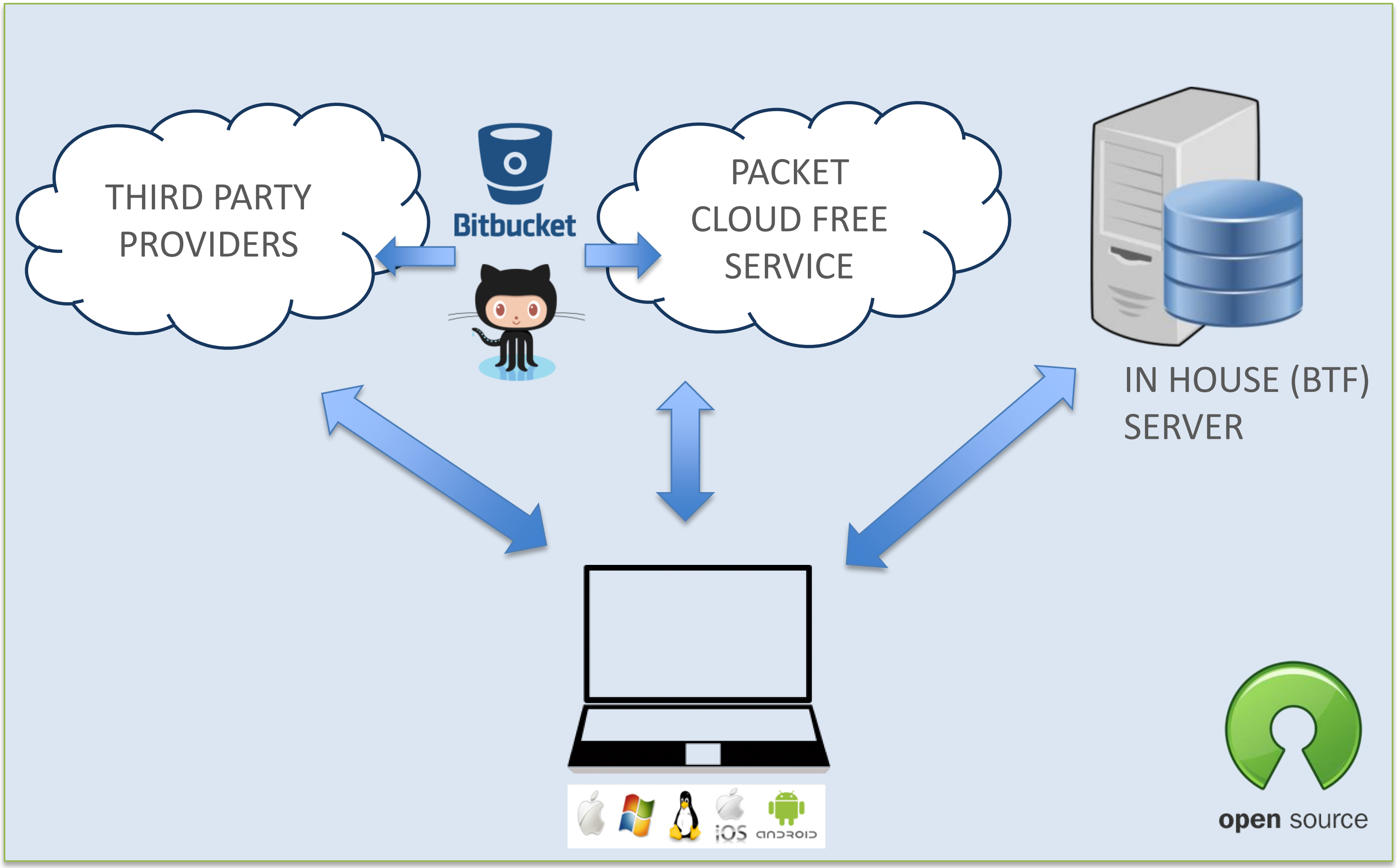
- Biicode metrics
- Too “corporate” language?







PACKET: MULTI OS & DISTRIBUTED & OSS

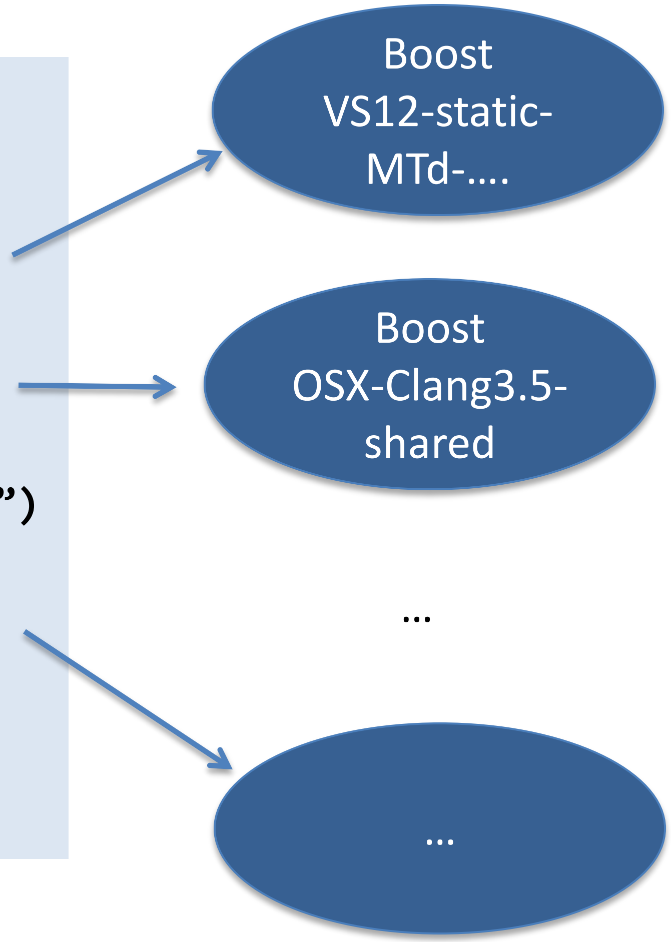


PACKET: RELEASE & BINARIES MANAGEMENT

```
class BoostPkt(BasePkt):
    name= “boost”
    version = “2.0”
    git: https://github.com/...
    options = {“static”: True}

    def reqs(self):
        if …:
            self.requires(“somelib…”)

    def build(self):
        if settings.os == “Windows”
            and options.static:
    …
```



PACKET: DEPENDENCIES

- Contribute with your own pre-compiled binaries.
- Transitive deps, version management, deps overriding, deps conflict resolution, conditional dependencies
- **Something that can be used to release/depend on boost with 0 lockin**

THANK YOU!

C++ now

biicode
C/C++ DEPENDENCY MANAGER

diego@biicode.com

@biicode