ABI compatibility
is not a **MAJOR** problem

MINOR
PATCH

Javier G. Sogo

jgsogo@gmail.com
javierg@jfrog.com
@jgsogo

C++ Italia 2019

June 15th, 2019
Milan, IT

ABI compatibility
is not a **MAJOR** problem

MINOR
PATCH

Javier G. Sogo

jgsogo@gmail.com
javierg@jfrog.com
@jgsogo

C++ Italia 2019

June 15th, 2019
Milan, IT

HOST

PATRON

Community Crumbs

SPONSORS

what is this talk about?

- **ABI**, not API

> **API (Application Programming Interface):** *A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.*
>
> oxforddictionaries.com

It's a **contract**: language/compiler + documentation

what is this talk about?

API is defined by

- Names
- Signatures
- Declarations location
- Functionality (docs)

std::apply

Defined in header <tuple>

```
template <class F, class Tuple>
constexpr decltype(auto) apply(F&& f, Tuple&& t);
```

(since C++17)

Invoke the *Callable* object f with a tuple of arguments.

Parameters

f - *Callable* object to be invoked

t - tuple whose elements to be used as arguments to f

Return value

The value returned by f.

cppreference.com

what is this talk about?

- **ABI**, not API

> **ABI (Application Binary Interface):** *an interface between two binary program modules. An ABI defines how data structures or computational routines are accessed in machine code, which is a low-level, hardware-dependent format.*
>
> en.wikipedia.org

Not covered by the C++ Standard (implementation detail)

what is this talk about?

ABI can be affected by

Infrastructure (DevOps, CI, Sysadmin)

- Calling convention
- Exception handling
- Name mangling
- C++ runtime

Pablo Ruíz Picasso. *Guernica* (fragment). 1937

what is this talk about?

Pablo Ruíz Picasso. *Guernica* (fragment). 1937

ABI can be affected by

Code (Developer)

- Binary representation of types
- vtable layout
- Inheritance, namespaces, overloading,...

what is this talk about?

- **ABI Compatibility**

A library is **binary compatible** if another module linked dynamically with a former version of that library continues running with newer versions without the need of recompiling.

Source compatibility: a program need to be recompiled against a new version of the library, but no further changes are needed.

CONAN C/C++ Package Manager

FOSS (MIT), including in-house server

Decentralized/distributed, git-like

Build system agnostic

~ 2900 stars

cpplang/#conan

@conan-io

Barbarians

- Diego Rodríguez-Losada
- Luis Martínez de Bartolomé
- Daniel Manzaneque
- Javier García Sogo
- Uilian Ries
- Konstantine Ivlev
- Carlos Zoido
- …and, **we are hiring!**

what is this talk about?

Pablo Ruíz Picasso. *Guernica*. 1937
Museo Nacional Centro de Arte Reina Sofía, Madrid, Spain

ABI compatibility

is ~~not~~ a **MAJOR** problem

HARD

COMPLEX

ABI compatibility, should I care?

- If you build an **application**, and
  - All your code is in the same repo, or
  - You compile always everything from sources, and
  - Do not allow plugins

- If you build a **library**
  - you DO care about ABI compatibility
  - …unless you are writing a header-only

ABI compatibility, should I care?

- If you build an **application**, and
  - All your code is in the same repo, or
  - You compile always everything from sources, and
  - Do not allow plugins

😥
😢
😭

- If you build a **library**
  - you DO care about ABI compatibility
  - …~~unless~~ also if you are writing a header-only


Challenger #1

The runtime

javierg@jfrog.com | @jgsogo | JFrog Ltd.

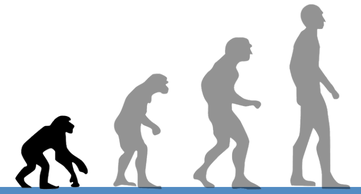
Challenger #1 - The runtime 

```c
int main( int argc, const char* argv[] )
{
    printf( "Scenario #1" );

    struct dirent **namelist;
    int n;

    n = scandir(".", &namelist, NULL, alphasort);

    if (n < 0)
        perror("scandir");
    else {
        while (n--) {
            printf("%s\n", namelist[n]->d_name);
            free(namelist[n]);
        }
        free(namelist);
    }
}
```

C plain application (C89 standard)

c-plain-app
gcc main.c -o c-plain-app

$ ./c-plain-app



Challenger #1 - The runtime 

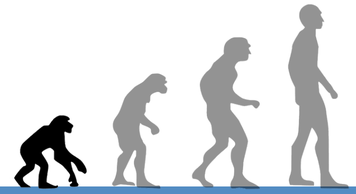
C standard library (libc)

- Many implementations: glibc, musl, uClibc, dietlibc,...

ABI and versioning comparison	musl	uClibc	dietlibc	glibc
Stable ABI	yes	no	unofficially	yes
LSB-compatible ABI	incomplete	no	no	yes
Backwards compatibility	yes	no	unofficially	yes
Forwards compatibility	yes	no	unofficially	no
Atomic upgrades	yes	no	no	no
Symbol versioning	no	no	no	yes

- “Provided” by the running distro

```
bash: ./c-plain-app: /lib/ld-musl-x86_64.so.1: bad ELF interpreter: No such file or directory
bash: ./c-plain-app: No such file or directory
./c-plain-app: /lib64/libc.so.6: version `GLIBC_2.15' not found (required by ./c-plain-app)
```



Challenger #1 - The runtime 

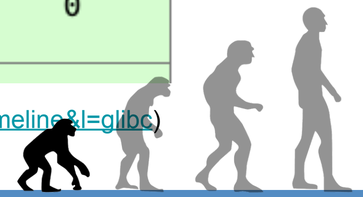
glibc: GNU C library

- Backwards compatible
- Compile your executables against a really old distro (centos6)
- Be careful even if it works

Version	Date	Soname	Change Log	Backward Compat.	Added Symbols	Removed Symbols
2.29	2019-01-31	0/1/2/6	changelog	99.96%	2 new	0
2.28	2018-08-01	0/1/2/6	changelog	99.92%	27 new	0
2.27	2018-02-01	0/1/2/6	changelog	98.49%	15 new	0
2.26	2017-08-02	0/1/2/6	changelog	99.54%	33 new	5 removed
2.25	2017-02-05	0/1/2/6	changelog	99.70%	22 new	0
2.24	2016-08-02	0/1/2/6	changelog	99.76%	3 new	0
2.23	2016-02-18	0/1/2/6	changelog	99.96%	3 new	0

!

Table from ABI Laboratory (https://abi-laboratory.pro/?view=timeline&l=glibc)



javierg@jfrog.com | @jgsogo | JFrog Ltd.

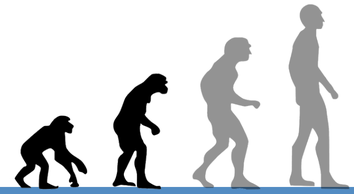
Challenger #2

The tools

Challenger #2 - Tools 

Shit happens!

- GCC 4.7.0 vs GCC 4.7.1
  - a data member was added to `std::list`
  - `std::pair`'s move constructor was not trivial
- GCC has some defaults
  - C++ dialect `-fabi-version`
  - Dual ABI: `_GLIBCXX_USE_CXX11_ABI`
  - ...



Challenger #2 - Tools 

`_GLIBCXX_USE_CXX11_ABI` and GCC > 5.x

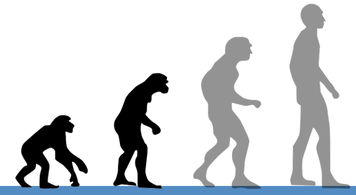
```cpp
#define _GLIBCXX_USE_CXX11_ABI 1

#include <string>

void function(std::string input) {}
```

GCC < 5.0 function(std::basic_string<char, std::char_traits<char>, std::allocator<char> >)

GCC > 5.x function(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >)

```cpp
#define _GLIBCXX_USE_CXX11_ABI 0
```

GCC > 5.x function(std::basic_string<char, std::char_traits<char>, std::allocator<char> >)



Challenger #2 - Tools 

`_GLIBCXX_USE_CXX11_ABI` and GCC > 5.x

```cpp
#define _GLIBCXX_USE_CXX11_ABI 1

#include <string>

void function(std::string input) {}
```

Force C++11 ABI

```
$> cat /etc/os-release
NAME="Ubuntu"
VERSION="14.04.6 LTS, Trusty Tahr"
```

Old distro (no dual ABI)

```
$> g++ --version
g++ (Ubuntu 6.5.0-2ubuntu1~14.04.1) 6.5.0 20181026
```

New compiler