



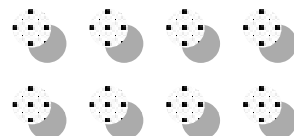
搞个 LibreOffice 移植先

陈璇

12/06/24

啥是 LibreOffice ?

- 开源软件中□为流行的 Office 套件，支持多种硬件架构 / 操作系统
- 起源于 1985 年的 StarWriter
- 2000 年，被 Sun 收购后开放源代码，名为 OpenOffice.org (简称 OOO)
- 2010 年，Apache 收购 Sun 后，很快就放弃了对 OOO 的支持。OOO 成员出走，建立了分支 LibreOffice
- **2022 年 11 月，LibreOffice 支持 riscv64 架构**

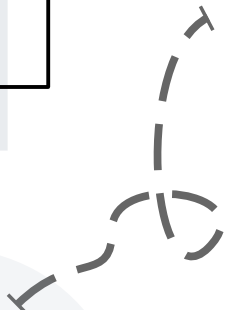




LibreOffice riscv64 port 现状

- 除了涉及到 NaN Propagation 的五个测试之外全部通过，针对失败测试，上游已合入相关代码，允许构建者选择跳过 *
- LibreOffice 已出现在 Debian Testing riscv64 ，预计将在 Debian Trixie 可用

* 在 autogen.sh 阶段使用 --disable-nan-tests 选项





如何 Port



很简单，只需要三步！



第 1 步：修改构建系统

- **Configure**
 - `configure.ac`
 - `m4/ax_boost_base.m4`
- **Makefile**
 - < 新增 > `solv/gbuild/platform/LINUX_RISCV64_GCC.mk`
 - `bridges/Library_cpp_uno.mk`
- **把其他架构的代码复制一份**
 - < 新增 > `bridges/source/cpp_uno/gcc3_linux_riscv64/*`

第 2 步：汇编替换

- 以下文件皆在 `bridges/source/cpp_uno/gcc3_linux_riscv64`
- `cpp2uno.cxx`
 - `codeSnippet()` 部分
 - `cpp2uno_call()` 部分
- `call.s`
 - 全文替换
- `uno2cpp.cxx`
 - `callVirtualFunction()` 部分

第 3 步：发新闻稿



移植的核心工作 / 架构相关的地方

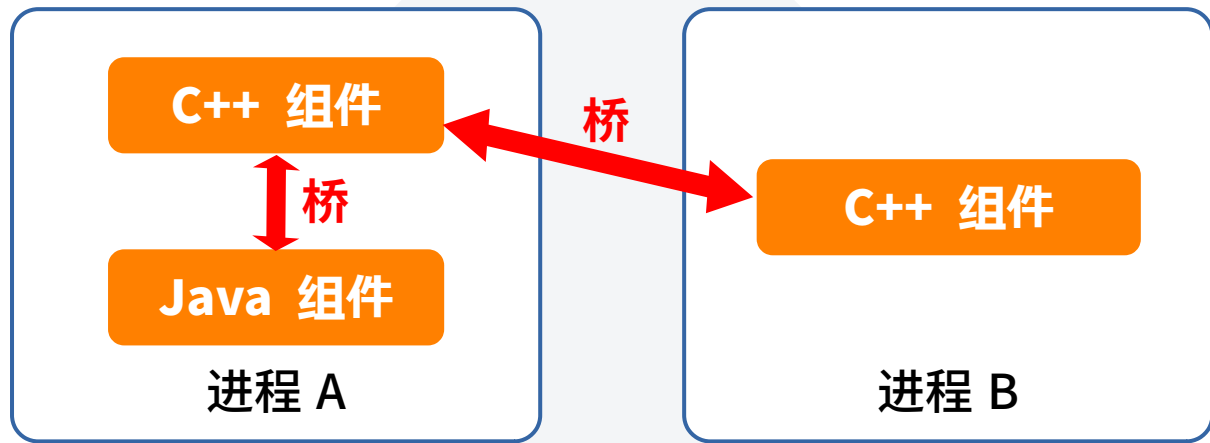
- 核心代码： UNO Bridge
- `bridges/source/cpp_uno/gcc3_linux_riscv64`

Universal Network Object

- UNO 是一种远程调用框架 / 对象模型技术，类似于 CORBA、COM/DCOM、Java RMI、GRPC 等
- 起源于 OpenOffice 早期，用以弥补当时类似技术的不足
- 跨语言、跨进程、跨网络
- 每个由某种语言编写的 LibreOffice 组件运行在一个 URE（UNO 运行时环境）中，并且通过桥（bridge）进行通信

Universal Network Object

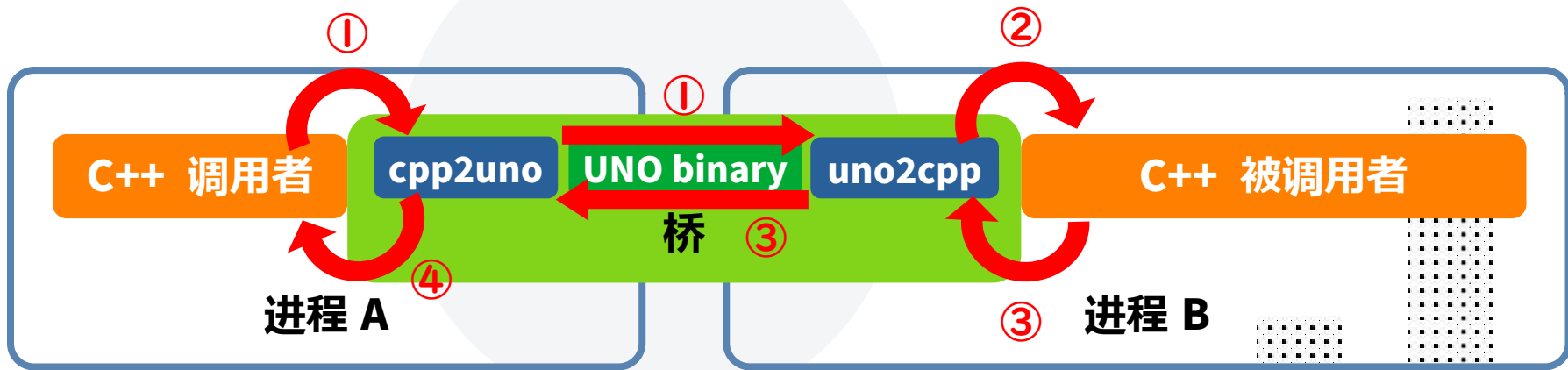
- 跨语言、跨进程、跨网络
- 每个由某种语言编写的 LibreOffice 组件运行在一个 URE（UNO 运行时环境）中，并且通过桥（bridge）进行通信



UNO Bridge

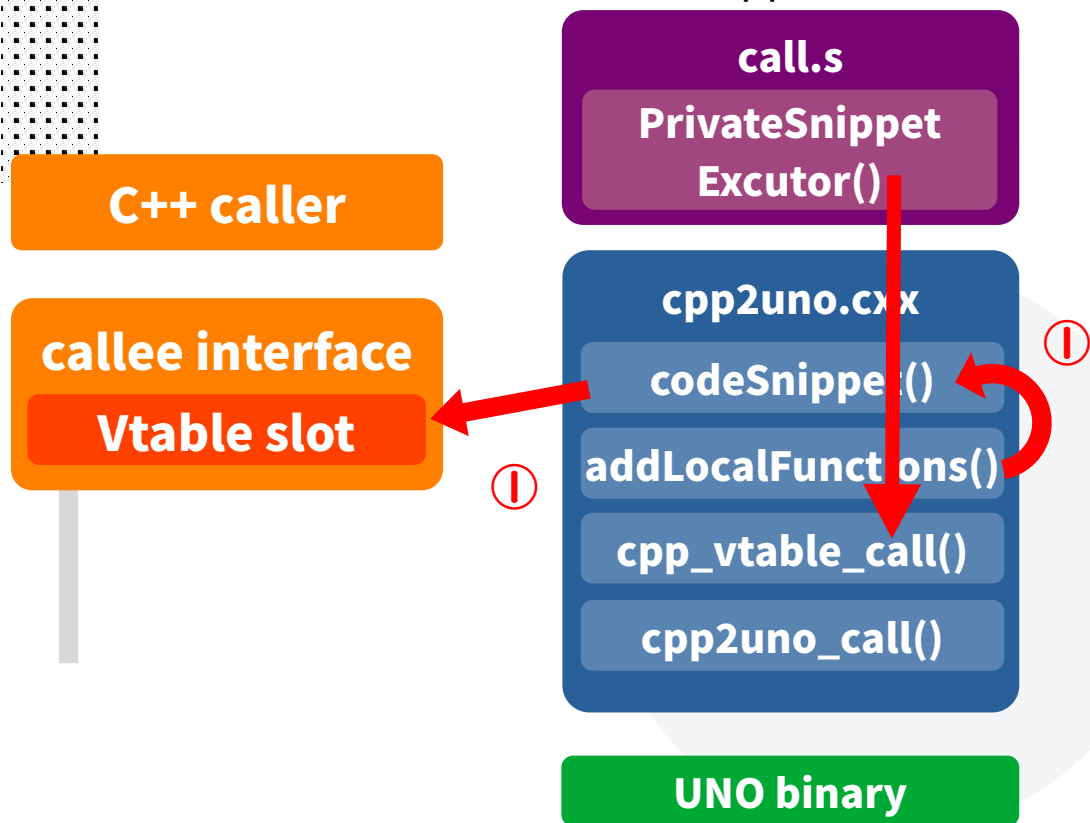
一个 C++ 到 C++ 的例子

1. cpp2uno 把 caller 传来的参数转为 UNO binary（装箱），传递到 callee 进程
2. uno2cpp 把 UNO binary 转换成 C++ 参数（拆箱），调用 callee
3. uno2cpp 把 callee 传来的返回值转换成 UNO binary（装箱），传递到 caller 进程
4. cpp2uno 把 UNO binary 解包（拆箱），传回 caller



UNO Bridge

cpp2uno: caller – uno binary



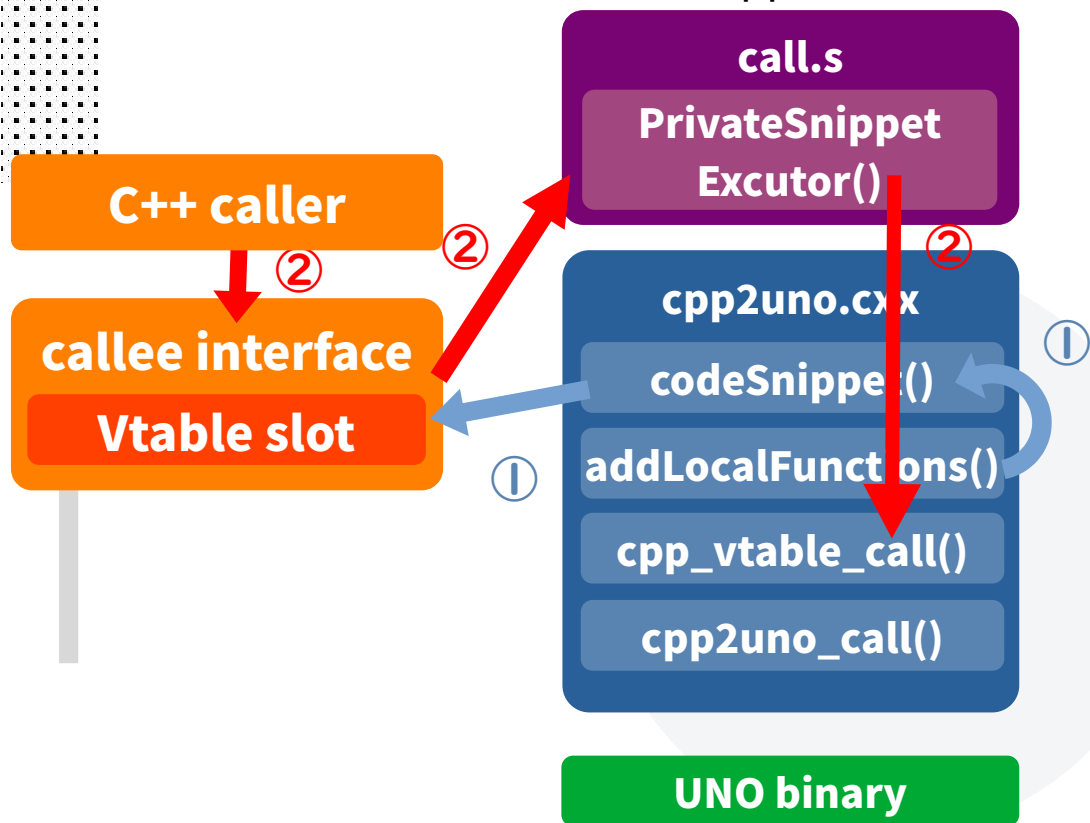
① caller 从 service manager 获取 callee 接口, `addLocalFunctions()` 通过 `codeSnippet()` 生成跳板代码, 插入 caller 对应 callee 的 **Vtable slot**

callee 已被替换为 unobridge 的一部分, 整个调用工作由 unobridge 接管

但 caller 仍然以为自己调用的是 callee
可怜的 caller

UNO Bridge

cpp2uno: caller – uno binary



② 当 caller 调用 callee 时，跳板代码跳转到

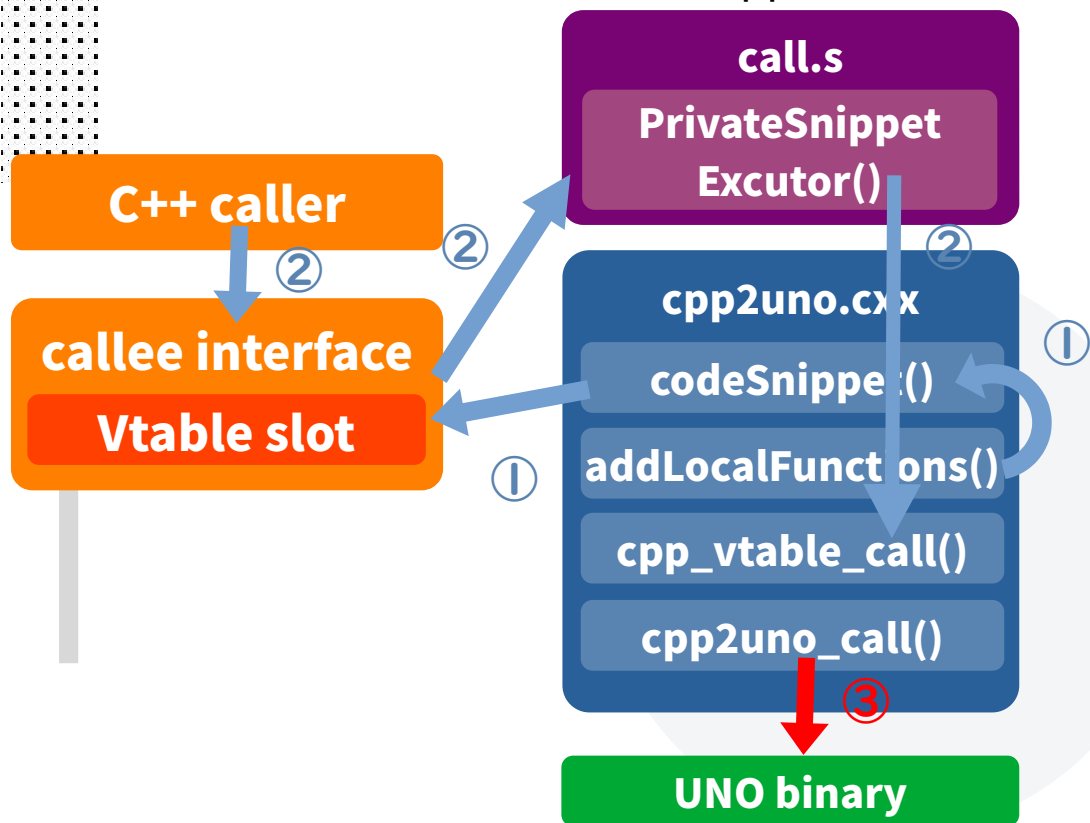
PrivateSnippetExcutor()，将所有函数参数寄存器压栈后跳入

cpp_vtable_call()

保护参数，并让 **cpp2uno** 接管调用

UNO Bridge

cpp2uno: caller – uno binary



③ `cpp_vtable_call()` 与 `cpp2uno_call()` 将 **c++** 参数转为 **uno binary** 参数，并进行远程调用

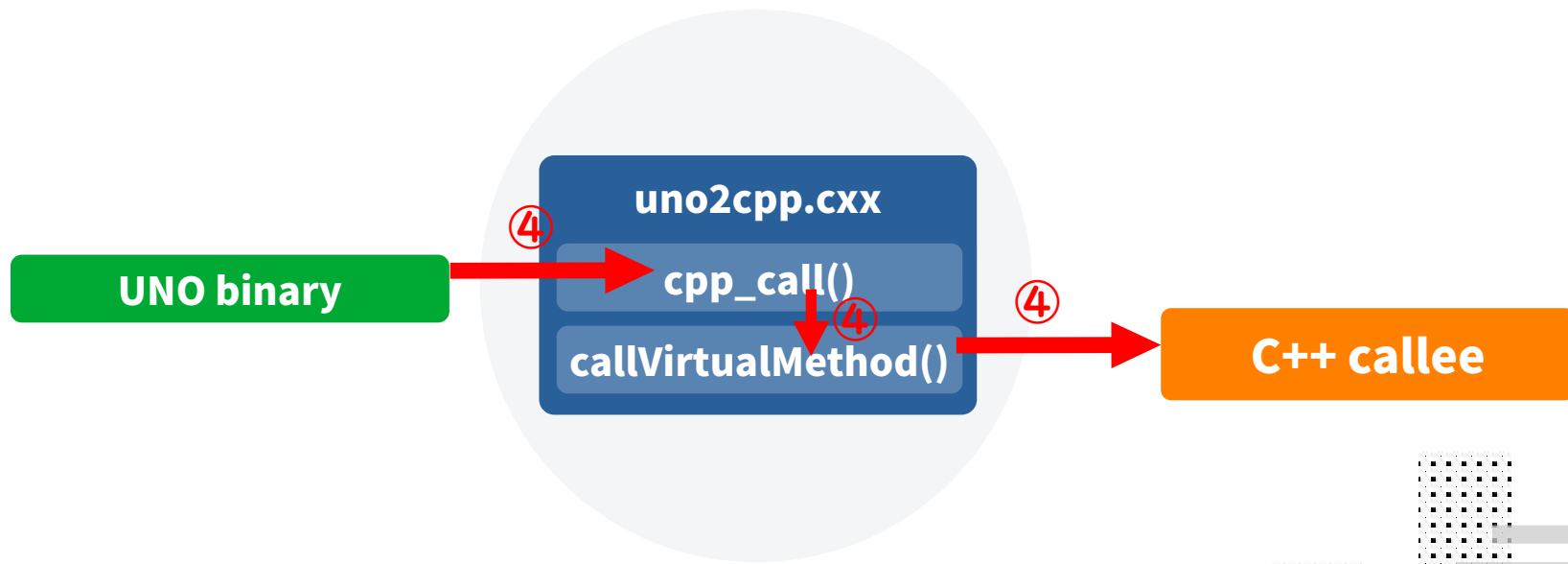
所谓「转换」，主要有两方面

- 1) 远程调用需要尽量压缩传递内容，所以要剔除冗余信息
- 2) UNO 对接口类型有特殊的定义，需要单独处理

UNO Bridge

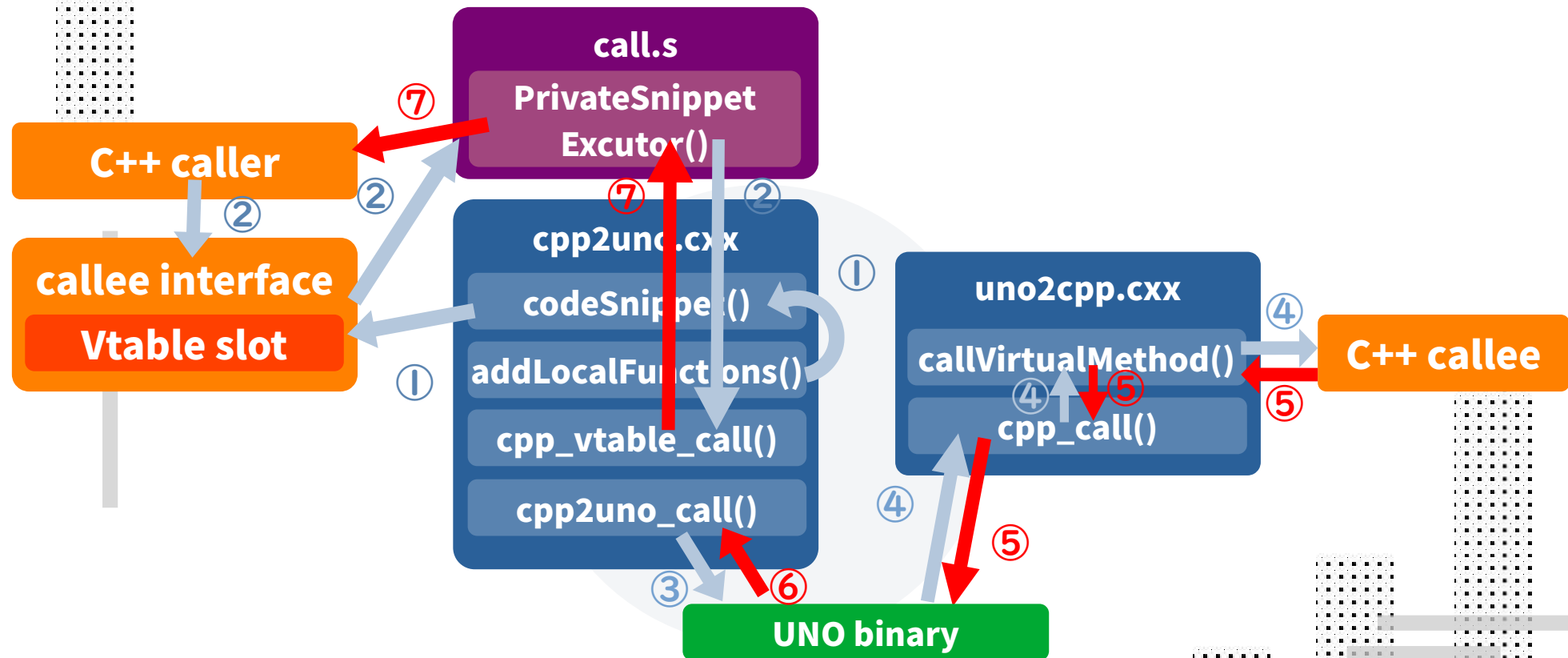
uno2cpp: uno binary – callee – uno binary

④ `cpp_call()` 将数据拆箱，转换成 C++ 参数，通过 `callVirtualMethod()` 调用 C++ callee



UNO Bridge

整体结构



Port 需要前置知识

- 1) 熟知该架构的 ABI (application binary interface)
- 2) 对 UNO Bridge 的大致了解
- 3) memset、指针等内存操作
- 4) 熟练使用 Debugger 调试代码，包括但不限于 C、Java、Python

坑 1

不同类型 **bridge** 对数据类型实现有差异

问题： 整数类型在 py-uno 一端是统一按照 Py_ssize_t 类型（64bit）来处理的，那么来自 cpp-uno 的数据加载到 py-uno 环境中时脏内存可能会污染高位

解决方案： cpp-uno 装箱整数时要做清理，要视情况将高位清零或符号拓展，见 abi.cxx 的 extIntBits() 函数

坑 2

结构体的拆装箱

问题： java-uno 与 py-uno 是统一代码，不可调整。那么在传递诸如结构体等数据结构时 cpp-uno 应当怎样拆箱、装箱？

解决方案： 照着 ABI 手册，对着 bridgetest 里的每个样例一个个糊吧

调试建议

- 1) 核心测试——请在进行 CUT/JUT 等单元测试之前确保该测试通过

`workdir/CustomTarget/testtools/bridgetest`

- 2) 官方调试建议

`https://wiki.documentfoundation.org/Development/How_to_debug`

- 3) 对于 Java 组件，建议使用 x86 上的 IDE 进行远程调试

`https://github.com/Sakura286/libreoffice-riscv-port-memo/blob/main/doc/Debug_Java.md`

- 4) UI Test 耗时较长，可以选择执行某个 .py 文件，甚至自己手写一个缩略版本进行测试

`https://github.com/Sakura286/libreoffice-riscv64-performance-testing/blob/main/doc/UI_Test_memo.md`

`https://github.com/Sakura286/libreoffice-riscv-port-memo/blob/main/doc/Debug_UI_Test.md`

参考信息

1) Debian Trixie 关于 LibreOffice 移植的现状

<https://buildd.debian.org/status/package.php?p=libreoffice&suite=trixie>

2) UNO 开发文档 - 文档基金会

https://wiki.documentfoundation.org/Documentation/DevGuide/Advanced_UNO

3) UNO intro - Libreoffice Blog

<https://niocs.github.io/L0Book/extensions/part1.html>

4) Universal Network Objects - inside-libreoffice

<https://chris-sherlock.gitbook.io/inside-libreoffice/universal-network-objects>

感谢收看

sakura286@outlook.com

