Table of Contents

简介

编译安装

由于笔者日常工作环境中很少使用其他系统平台，暂时abtool只支持针对macOS系统的编译运行，感兴趣的朋友可以自行适配其他系统平台，涉及平台差异的部分主要是以下几个外部链接库：

- libreadline.tbd
- libfbxsdk.a
  - Foundation.framework
  - libiconv.tbd
  - libxml2.tbd
  - libz.tbd

abtool源码基于C++14标准库实现，理论上处理好以上几个外部库的依赖问题即可成功编译。

首次编译abtool

如果您不是第一次使用abtool，也就是说您手上已经有了一份abtool工具，那么可以跳过该步骤，直接进行下一步操作。

1. Xcode编译

   打开Xcode，使用组合键 `⌘+B` 即可进行源码编译，之后编译可以在终端环境或者shell脚本里面随意使用，这是生成abtool工具的最简单的方式，需要注意的是请确保目标目录 `/usr/local/bin` 已被预先创建。

2. CMake编译

   执行如下脚本，即可在 `build/bin` 目录得到abtool命令行工具文件。

```
# 当前cd目录为工程根目录
mkdir build
cd build
cmake ..
cmake --build .
```

生成TypeTree数据

TypeTree记录了资源对象数据的序列化信息，收集TypeTree的目的是为了把Unity的类型信息集成到abtool工具里面，只有这样abtool才有可能实现它的功能。

为了让大家快速体验整个工具编译过程，笔者在工程doc目录准备了 `QuickStart.unitypackage` 资源包，现在您只需要新建一个新的Unity工程，然后导入所有资源，通过Unity菜单 `abtool/Build Asset Bundles` 即可快速生成包含了TypeTree信息的ab文件，该资源包包含了能够让abtool源码正常编译的最小集合，准确来说是，资源里面包含了以下编译必须的资源对象类型：

- GameObject
- RectTransform

- Transform
- TextAsset
- Texture2D
- Cubemap
- Material
- Shader
- SpriteRenderer
- SkinnedMeshRenderer
- MeshRenderer
- ParticleSystemRenderer
- LineRenderer
- TrailRenderer
- MeshFilter
- Animator
- Mesh

如果您现有的项目资源已经覆盖了以上资源类型，那么可以放心使用abtool收集相应的TypeTree数据。然而QuickStart资源包只是覆盖了最小集合的资源类型，如果需要最大限度发挥abtool的效力，笔者强烈建议您扫描尽可能多的ab文件，从而可以收集到尽可能多的Unity类型数据，这样会让您定位资源问题更加得心应手，相信您在后续的日常使用中会深刻明白这一点。

```
# doc/resources目录存储了QuickStart资源编译的iOS/Android双平台的ab文件
cd doc/resources
# 由于QuickStart生成的ab文件后缀为ab，所以可以通过'*.ab'进行文件匹配
# 请根据实际项目的ab文件后缀做适当修改
find . -iname '*.ab' | xargs abtool savetree -a types.tte
```

上述脚本通过 `find` 命令查找所有的ab文件，并把这些文件通过 `xargs` 透传给abtool工具去处理，最终会在当前目录生成 `types.tte` 文件（当然也可以通过 `savetree` 的 `-a` 参数指定其他保存目录），这就是我们需要的Unity类型数据。

生成对象序列化代码

`types.tte` 是个二进制文件，把它转换成C++代码才能最终为abtool所用，通过下面这行命令可以轻松完成这个任务，整个过程就好比使用 `protoc` 编译 `*.proto` 文件一样。

```
# 当前cd目录为工程根目录
abtool gtt -a doc/resources/types.tte -o abtool/assetbundles/unity
```

由于上面的脚本是在工程根目录执行，并且代码的输出目录为 `abtool/assetbundles/unity`，所以当脚本执行完成后工程的代码就得到了更新。

最终编译abtool

通过上一步骤我们修改了Unity资源对象的序列化代码，所以还需要再次编译，这样我们就最终得到了功能完备的abtool，通过后续的章节可以逐渐窥探它强大的威力。

> 什么情况下需要重新编译abtool?

1. 升级了Unity版本
2. 修改了Unity源码里面涉及资源对象的序列化的代码
3. 修改了 `AssetBundleArchive` 容器存储结构
4. 修改了 `SerializedFile` 存储结构
5. 如果需要abtool正常处理所有 `MonoBehaviour` 组件数据，那么您需要定期编译abtool，不过我们大部分情况下并不关心这部分数据。

贴图

贴图

贴图

贴图

贴图

贴图

贴图

贴图

贴图

贴图

贴图

贴图

贴图

贴图

贴图