



# 从0开始，自顶向下地学习 V8's build system

智能软件研究中心 邱吉

qiuji@iscas.ac.cn

2020/02/07

# 目录

01 背景介绍

02 Quick Glance : V8 build 过程

03 GN简介

04 Ninja简介

05 V8的build system全景

# 目录

01 背景介绍

02 Quick Glance : V8 build 过程

03 GN简介

04 Ninja简介

05 V8的build system全景

今天 (2020-02-06) 的内容

## V8 是什么？

- V8 是 Google 浏览器 Chrome 的 JavaScript 引擎, 开源
- JavaScript 是一种脚本语言, JavaScript 的引擎负责将这种语言编写的程序进行“解释执行” (interpreter) 或者 “即时编译成本地代码然后执行” ( Just-In-Time compile to native code )
- Google Chrome 在当前浏览器市场中占据了绝对份额优势  
( [https://en.wikipedia.org/wiki/Usage\\_share\\_of\\_web\\_browser](https://en.wikipedia.org/wiki/Usage_share_of_web_browser) )
- V8 is one of the best JavaScript Engine in the world , V8的代码在 Chromium项目中
- Chromium 是 Chrome 浏览器的开源项目名称, Chrome 浏览器基于 Chromium 的代码

## V8 的代码获取和编译—需要网络代理

### ● 参考网页

- <https://v8.dev/docs/build>
- <https://v8.dev/docs/source-code>
- [https://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot\\_tools/docs/html/depot\\_tools\\_tutorial.html](https://commondatastorage.googleapis.com/chrome-infra-docs/flat/depot_tools/docs/html/depot_tools_tutorial.html)

### ● V8可编译代码包的下载，依赖于 depot\_tools，下载depot\_tools后设置PATH到该目录下，后续主要会用到如下的组件：（列举出来是说明我们可以修改/打印/探索）

- gn (bash) → gn.py → gn(binary in V8 buildtools)
- ninja (bash) → ninja-linux64
- autoninja (bash) → autoninja.py (打印ninja命令行)

### ● 在设置好网络代理后，需要执行如下命令才能获得完整的 V8 所需构建内容：

- fetch v8
- gclient sync

### ● x64上v8 release版本的build命令：`"v8/tools/dev/gm.py x64.release"`

## 02 Quick Glance : V8 build 过程-1

### V8 的编译拆解-总体流程

执行 “`v8/tools/dev/gm.py x64.release`” 后发生了什么：

```
qiuji@ubuntu-s-1vcpu-3gb-sfo2-01:~/work/v8clean/v8$ ./tools/dev/gm.py x64.release
# mkdir -p out/x64.release ← 1
# echo > out/x64.release/args.gn << EOF
is_component_build = false
is_debug = false
target_cpu = "x64"
use_goma = false ← 2
goma_dir = "None"
v8_enable_backtrace = true
v8_enable_disassembler = true
v8_enable_object_print = true
v8_enable_verify_heap = true
EOF
# gn gen out/x64.release ← 3
Done. Made 142 targets from 83 files in 378ms
# autoninja -C out/x64.release d8 ← 4
/home/qiuji/work/depot_tools/ninja -C out/x64.release d8 -j 3 ← 5
ninja: Entering directory `out/x64.release'
[4/1403] CXX obj/torque_base/types.o ← 6
```

## 02 Quick Glance : V8 build 过程-2


### V8 的编译拆解-

- 执行命令 : `v8/tools/dev/gm.py x64.release`
- 执行主体 : `tools/dev/gm.py`
- 建立 `out/x64.release` 目录
- echo如下内容到`out/x64.release/args.gn`文件

```
is_component_build = false (编译 V8 的可执行文件, 而不是库文件)
is_debug = false (release编译, debug为false)
target_cpu = "x64" (编译出来的 V8 二进制 在x64上执行)
use_goma = false
goma_dir = "None"
v8_enable_backtrace = true
v8_enable_disassembler = true
v8_enable_object_print = true
```

## 02 Quick Glance : V8 build 过程-3

### V8 的编译拆解-

- 执行命令：gn gen out/x64.release
- 执行主体：
  - depot-tools/gn (bash)
  - depot-tools/gn.py
  - v8/buildtools/gn(binary in v8/buildtools)
- 产生的效果：读入前一步生成的args.gn文件，以及 v8 中所有相关的 GN 配置文件，在out/x64.release 目录生成了如下ninja文件和目录：
  - build.ninja：ninja 命令启动的总体配置，枚举了顶层的build statements 和 default 动作（阅读代码推荐关注 build \*\*\*）
  - build.ninja.d：上面的 build.ninja 文件所依赖的 gn 配置文件列表，当 gn 配置文件列表中的文件有更新时，再次执行gn gen命令时，会重新生成 build.ninja
  - toolchain.ninja：是 build.ninja 的 submodule, 定义了所有与编译、链接命令相关的规则和subninja（阅读代码推荐关注 rule \*\*\* 和 subninja \*\*\*）
  - obj/\*: v8不同子模块的 subninja 文件



### V8 的编译拆解-

- 执行命令：autoninja -C out/x64.release d8
- 执行主体：
  - depot-tools/autoninja (bash)
  - depot-tools/autoninja.py
- 产生的效果：根据 build 机器的特性和 args.gn的配置选项，生成最终的ninja命令然后eval（执行）这个命令，并做一些基于ninja编译log(.ninja\_log)的后续summary statistics
- Tips
  - ninja命令行中-j N的N是在 autoninja.py 中计算并赋值到 j\_value 变量的，如果build机器的内存较小，或者核数较少，j大于2就有可能导致编译过程崩溃（无明显错误提示直接退出/killed/停滞不前），所以可以在 autoninja.py 中直接把 j\_value 改成1，强制避免启动多个job，来避免这种问题；同时，也可以使得后续对详细的编译命令和过程的观察和调试，变得更加确定化
  - 编译v8生成的可执行文件叫做d8，d8是v8的shell

## 02 Quick Glance : V8 build 过程-5

### V8 的编译拆解-

- 执行命令 : `/home/qiuji/work/depot_tools/ninja -C out/x64.release d8 -j 3`
- 执行主体 :
  - `depot-tools/ninja ( bash )`
  - `depot-tools/ninja-linux64`
- 产生的效果 : 编译所有 third-party 依赖和 v8 源码, 在 `out` 目录下生成 d8 可执行文件

需要学习

ninja

->next week<-

## 02 Quick Glance : V8 build 过程-6

### V8 的编译拆解- 6

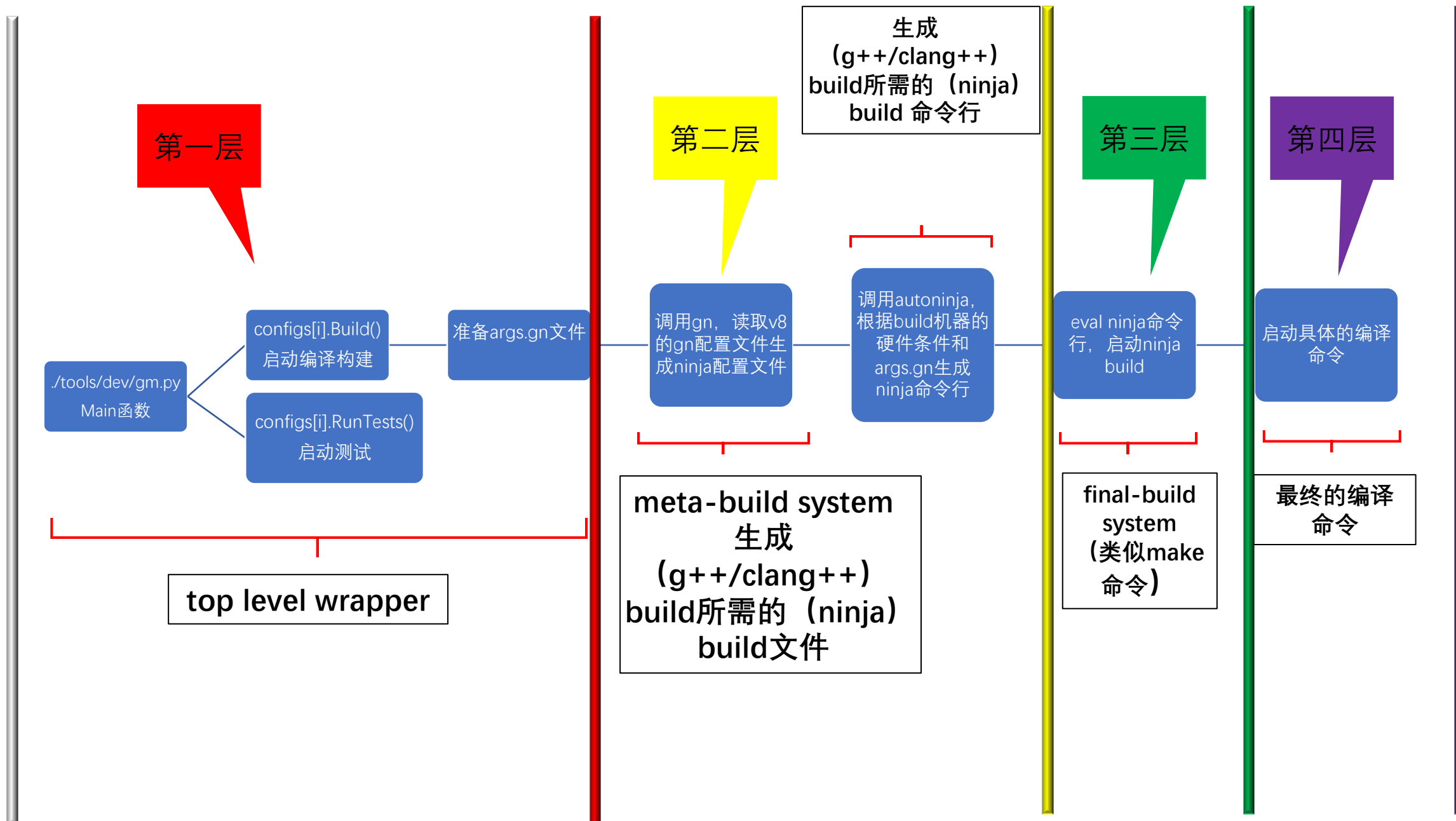
- 执行的具体命令可以通过在v8目录下手动执行如下命令，并通过 “2>&1 |tee log.txt” 来查看 /home/qiuji/work/depot\_tools/ninja -v -C out/x64.release d8 -j 3

```
[1/1524] g++ -MMD -MF obj/torque_base/global-context.o.d -DUSE_UDEV -DUSE_AURA=1 -DUSE_GLIB=1 -DUSE_NSS_CERTS=1 -DUSE_X11=1 -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE -D_STDC_CONSTANT_MACROS -D_STDC_FORMAT_MACROS -DCOMPONENT_BUILD -D_LICPP_ABI_UNSTABLE -D_LICPP_ABI_VERSION=Cr -D_LICPP_ENABLE_NODISCARD -D_LICPP_DEBUG=0 -DCR_LIBCXX_REVISION=375504 -D_DEBUG -DDYNAMIC_ANNOTATIONS_ENABLED=1 -DENABLE_DISASSEMBLER -DV8_TYPED_ARRAY_MAX_SIZE_IN_HEAP=64 -DENABLE_MINOR_MC -DOBJECT_PRINT -DVERIFY_HEAP -DV8_TRACE_MAPS -DV8_ENABLE_ALLOCATION_TIMEOUT -DV8_ENABLE_FORCE_SLOW_PATH -DV8_ENABLE_DOUBLE_CONST_STORE_CHECK -DV8_INTL_SUPPORT -DENABLE_HANDLE_ZAPPING -DV8_SNAPSHOT_NATIVE_CODE_COUNTERS -DV8_USE_EXTERNAL_STARTUP_DATA -DV8_CONCURRENT_MARKING -DV8_ARRAY_BUFFER_EXTENSION -DV8_ENABLE_LAZY_SOURCE_POSITIONS -DV8_CHECK_MICROTASKS_SCOPES_CONSISTENCY -DV8_EMBEDDED_BUILTINS -DV8_WIN64_UNWINDING_INFO -DV8_ENABLE_REGEXP_INTERPRETER_THREADED_DISPATCH -DV8_ENABLE_CHECKS -DV8_COMPRESS_POINTERS -DV8_31BIT_SMIS_ON_64BIT_ARCH -DV8_DEPRECATION_WARNINGS -DV8_IMMINENT_DEPRECATION_WARNINGS -DV8_TARGET_ARCH_ARM64 -DV8_HAVE_TARGET_OS -DV8_TARGET_OS_LINUX -DDEBUG -DENABLE_SLOW_DCHECKS -DBUILDING_V8_SHARED -DV8_ENABLE_CHECKS -DV8_COMPRESS_POINTERS -DV8_31BIT_SMIS_ON_64BIT_ARCH -DV8_DEPRECATION_WARNINGS -DV8_IMMINENT_DEPRECATION_WARNINGS -DBUILDING_V8_SHARED -DV8_ENABLE_CHECKS -DV8_COMPRESS_POINTERS -DV8_31BIT_SMIS_ON_64BIT_ARCH -DV8_DEPRECATION_WARNINGS -DV8_IMMINENT_DEPRECATION_WARNINGS -DUSING_V8_BASE_SHARED -I./.. -Igen -I./.. -Igen -I./.. -Igen -fno-strict-aliasing --param=ssp-buffer-size=4 -fstack-protector -funwind-tables -fPIC -pipe -B./../third_party/binutils/Linux_x64/Release/bin -pthread -m64 -march=x86-64 -Wno-builtin-macro-redefined -D__DATE__ -D__TIME__ -D__TIMESTAMP__ -Wall -Werror -Wno-unused-local-typedefs -Wno-maybe-uninitialized -Wno-deprecated-declarations -Wno-comments -Wno-packed-not-aligned -Wno-missing-field-initializers -Wno-unused-parameter -fno-omit-frame-pointer -g2 -gsplit-dwarf -fno-builtin-abs -Wno-strict-overflow -Wno-return-type-visibility=default -O3 -fno-ident -fdata-sections -ffunction-sections -std=gnu++14 -Wno-narrowing -Wno-class-memaccess -nostdinc++ -isystem./../buildtools/third_party/libc++/trunk/include -isystem./../buildtools/third_party/libc++abi/trunk/include -fexceptions -frtti -c ./../src/torque/global-context.cc -o obj/torque_base/global-context.o
```

- 执行主体：
  - 具体的编译器、链接器和其他 build tools，例如 g++/clang++ 等
  - depot-tools/ninja-linux64
- 产生的效果：
  - 生成若干obj文件和.a文件
  - 最终效果：在out/x64.release目录下生成 d8 可执行文件

具体的  
porting和  
debug需要深  
入这些命令

## 02 Summary : V8 build system框架和层次



## ● GN

- 全部文档：<https://gn.googlesource.com/gn/+master/docs/>

- quick 上手：

[https://docs.google.com/presentation/d/15Zwb53JcncHfEwHpnG\\_PolbbzQ3GQi\\_cpuYwbpcbZo/edit#slide=id.g1199fa62d0\\_3\\_5](https://docs.google.com/presentation/d/15Zwb53JcncHfEwHpnG_PolbbzQ3GQi_cpuYwbpcbZo/edit#slide=id.g1199fa62d0_3_5)

## ● Ninja

- 详细文档：<https://ninja-build.org/manual.html>

- quick上手：

[https://www.slideshare.net/cwdoh/ninja-build-simple-guide-for-beginners?from\\_action=save](https://www.slideshare.net/cwdoh/ninja-build-simple-guide-for-beginners?from_action=save)

Q&A

# 谢 谢

欢迎交流合作

2020/02/07