环境：Ubuntu虚拟机

参考资料：

1、[How to download Skia](https://chromium.googlesource.com/skia/+/chrome/m41/site/user/download.md)

2、[How to build Skia](https://skia.org/docs/user/build/" \l "macos)

3、[Skia Docx](https://skia.org/docs/)

[4、Skia API](https://api.skia.org/classSkPath.html" \l "details)

<https://github.com/cisen/blog/issues/464>

[windows下编译skia](https://redrain.blog.csdn.net/article/details/111685123)

[build Skia on MacOS](https://gist.github.com/velyan/7f474a09c51d2c7b658036913edfe003)

<https://www.twblogs.net/a/5c4b0056bd9eee6e7e06b6f5/?lang=zh-cn>

[关于mac visual studio mac运行gn项目](https://v8.dev/docs/ide-setup)

Mac.Linux平台静态库编译

git clone https://skia.googlesource.com/skia.git

python2 tools/git-sync-deps

bin/gn gen out/Static

ninja -C out/Static

//如果需要生成compile\_commands.json，方便给visual studio code使用（结合cquery）

ninja -C out/Static -t compdb cxx cc > compile\_commands.json

Mac&Linux&Android 过滤后代码：

代码位置：/Users/luozhipeng/Mydata/Ubuntu/skia\_mac

# Copyright 2019 Google LLC.

# Use of this source code is governed by a BSD-style license that can be

# found in the LICENSE file.

if (!defined(is\_skia\_standalone)) {

is\_skia\_standalone = false

}

is\_skia\_dev\_build = is\_skia\_standalone && !is\_official\_build

declare\_args() {

skia\_android\_serial = ""

skia\_compile\_processors = false

skia\_enable\_api\_available\_macro = true

skia\_enable\_android\_utils = false

skia\_enable\_ccpr = true

skia\_enable\_discrete\_gpu = true

skia\_enable\_flutter\_defines = false

skia\_enable\_fontmgr\_empty = true

skia\_enable\_fontmgr\_fuchsia = false

skia\_enable\_fontmgr\_win = false

skia\_enable\_gpu = false

skia\_enable\_pdf = false

skia\_enable\_skottie = !(is\_win && is\_component\_build)

skia\_enable\_skrive = true

skia\_enable\_skvm\_jit\_when\_possible = false

skia\_enable\_svg = false

skia\_enable\_tools = false

skia\_enable\_gpu\_debug\_layers = false

skia\_enable\_winuwp = false

skia\_generate\_workarounds = false

skia\_include\_multiframe\_procs = false

skia\_lex = false

skia\_libgifcodec\_path = "third\_party/externals/libgifcodec"

skia\_pdf\_subset\_harfbuzz =

false # TODO: set skia\_pdf\_subset\_harfbuzz to skia\_use\_harfbuzz.

skia\_qt\_path = getenv("QT\_PATH")

skia\_skqp\_global\_error\_tolerance = 0

skia\_tools\_require\_resources = false

skia\_update\_fuchsia\_sdk = false

skia\_use\_angle = false

skia\_use\_dawn = false

skia\_use\_direct3d = false

skia\_use\_egl = false

skia\_use\_expat = false

skia\_use\_experimental\_xform = false

skia\_use\_ffmpeg = false

skia\_use\_fixed\_gamma\_text = is\_android

skia\_use\_fontconfig = false

skia\_use\_fonthost\_mac = is\_mac || is\_ios

skia\_use\_freetype = false

skia\_use\_harfbuzz = false

skia\_use\_gl = false

skia\_use\_icu = false

skia\_use\_libheif = false

skia\_use\_libjpeg\_turbo\_decode = true

skia\_use\_libjpeg\_turbo\_encode = true

skia\_use\_libpng\_decode = true

skia\_use\_libpng\_encode = true

skia\_use\_libwebp\_decode = false

skia\_use\_libwebp\_encode = false

skia\_use\_lua = is\_skia\_dev\_build && !is\_ios

skia\_use\_metal = false

skia\_use\_ndk\_images = is\_android && defined(ndk\_api) && ndk\_api >= 30

skia\_use\_opencl = false

skia\_use\_piex = !is\_win

skia\_use\_sfml = false

skia\_use\_webgl = false

skia\_use\_wuffs = false

skia\_use\_x11 = false

skia\_use\_xps = false

skia\_use\_zlib = false

skia\_vtune\_path = ""

if (is\_ios) {

skia\_ios\_identity = ".\*Google.\*"

skia\_ios\_profile = "Google Development"

}

if (is\_mac) {

skia\_gl\_standard = "gl"

} else if (is\_ios) {

skia\_gl\_standard = "gles"

} else {

skia\_gl\_standard = ""

}

if (is\_android) {

skia\_use\_vulkan = defined(ndk\_api) && ndk\_api >= 24

} else if (is\_fuchsia) {

skia\_use\_vulkan = true

} else {

skia\_use\_vulkan = false

}

skia\_build\_fuzzers = is\_clang && is\_linux && target\_cpu == "x64"

skia\_use\_libfuzzer\_defaults = true

}

declare\_args() {

skia\_compile\_sksl\_tests = skia\_compile\_processors

skia\_enable\_fontmgr\_android = skia\_use\_expat && skia\_use\_freetype

skia\_enable\_fontmgr\_custom\_directory = skia\_use\_freetype && !is\_fuchsia

skia\_enable\_fontmgr\_custom\_embedded = skia\_use\_freetype && !is\_fuchsia

skia\_enable\_fontmgr\_custom\_empty = skia\_use\_freetype

skia\_enable\_fontmgr\_fontconfig = skia\_use\_freetype && skia\_use\_fontconfig

skia\_enable\_fontmgr\_win\_gdi = is\_win && !skia\_enable\_winuwp

skia\_enable\_fontmgr\_FontConfigInterface =

skia\_use\_freetype && skia\_use\_fontconfig

skia\_enable\_spirv\_validation = is\_skia\_dev\_build && is\_debug && !skia\_use\_dawn

skia\_use\_dng\_sdk = false

skia\_use\_libgifcodec = !skia\_use\_wuffs

skia\_use\_sfntly = skia\_use\_icu

skia\_enable\_vulkan\_debug\_layers = skia\_enable\_gpu\_debug\_layers

skia\_enable\_direct3d\_debug\_layer = skia\_enable\_gpu\_debug\_layers

skia\_use\_vma = skia\_use\_vulkan

}

declare\_args() {

# skia\_fontmgr\_factory should define SkFontMgr::Factory()

if (skia\_enable\_fontmgr\_empty) {

skia\_fontmgr\_factory = ":fontmgr\_empty\_factory"

} else if (is\_android && skia\_enable\_fontmgr\_android) {

skia\_fontmgr\_factory = ":fontmgr\_android\_factory"

} else if (is\_win && skia\_enable\_fontmgr\_win) {

skia\_fontmgr\_factory = ":fontmgr\_win\_factory"

} else if ((is\_mac || is\_ios) && skia\_use\_fonthost\_mac) {

skia\_fontmgr\_factory = ":fontmgr\_mac\_ct\_factory"

} else if (skia\_enable\_fontmgr\_fontconfig) {

skia\_fontmgr\_factory = ":fontmgr\_fontconfig\_factory"

} else if (skia\_enable\_fontmgr\_custom\_directory) {

skia\_fontmgr\_factory = ":fontmgr\_custom\_directory\_factory"

} else if (skia\_enable\_fontmgr\_custom\_embedded) {

skia\_fontmgr\_factory = ":fontmgr\_custom\_embedded\_factory"

} else if (skia\_enable\_fontmgr\_custom\_empty) {

skia\_fontmgr\_factory = ":fontmgr\_custom\_empty\_factory"

} else {

#"src/ports/SkFontMgr\_FontConfigInterface\_factory.cpp" #WontFix

#"src/ports/SkFontMgr\_win\_gdi\_factory.cpp" # WontFix

skia\_fontmgr\_factory = ":fontmgr\_empty\_factory"

}

}

# Our tools require static linking (they use non-exported symbols), and the GPU backend.

skia\_enable\_tools = skia\_enable\_tools && !is\_component\_build && skia\_enable\_gpu

# Skia's targets may be built inside other gn build systems.

# Skia builds other project's build targets inside its build.

# This is easier if the built-in target types remain generic.

# Place Skia target specific configs in skia\_target\_default\_configs.

# These will be applied by the 'skia\_\*' templates.

# In the Skia build these configs apply many warnings as errors.

# Other projects may optionally set these configs as they see fit.

template("skia\_target") {

target(invoker.\_skia\_target\_type, target\_name) {

# set\_defaults(invoker.\_skia\_target\_type) might not exist or set configs

if (!defined(configs)) {

configs = []

}

# Explicit configs instead of set\_defaults("skia\_target")

# Allows template("skia\_\*") below to avoid the configs dance.

if (defined(skia\_target\_default\_configs)) {

configs += skia\_target\_default\_configs

}

# "\*" clobbers the current scope; append to existing configs

forward\_variables\_from(invoker, "\*", [ "configs" ])

if (defined(invoker.configs)) {

configs += invoker.configs

}

}

}

template("skia\_executable") {

skia\_target(target\_name) {

assert(!defined(configs), "No set\_defaults(skia\_target)")

\_skia\_target\_type = "executable"

forward\_variables\_from(invoker, "\*")

}

}

template("skia\_source\_set") {

skia\_target(target\_name) {

assert(!defined(configs), "No set\_defaults(skia\_target)")

\_skia\_target\_type = "source\_set"

forward\_variables\_from(invoker, "\*")

}

}

template("skia\_static\_library") {

skia\_target(target\_name) {

assert(!defined(configs), "No set\_defaults(skia\_target)")

\_skia\_target\_type = "static\_library"

forward\_variables\_from(invoker, "\*")

}

}

template("skia\_shared\_library") {

skia\_target(target\_name) {

assert(!defined(configs), "No set\_defaults(skia\_target)")

\_skia\_target\_type = "shared\_library"

forward\_variables\_from(invoker, "\*")

}

}

template("skia\_component") {

skia\_target(target\_name) {

assert(!defined(configs), "No set\_defaults(skia\_target)")

\_skia\_target\_type = "component"

forward\_variables\_from(invoker, "\*")

}

}

linux环境中编译：

─$ export PATH=$PATH:/mnt/hgfs/Ubuntu/Git\_PDFIum/Resources/depot\_tools

─$ python2 tools/git-sync-deps

─$ bin/gn gen out/linux --args='is\_official\_build=false is\_component\_build=false'

─$ ninja -C out/linux

Mac环境中编译：

─$ export PATH=$PATH:/Users/luozhipeng/Mydata/Ubuntu/Git\_PDFIum/Resources/depot\_tools

─$ python2 tools/git-sync-deps

─$ bin/gn gen out/mac --args='is\_official\_build=false is\_component\_build=false'

─$ ninja -C out/mac

Android环境中编译（在Mac中进行交叉编译）：

[将 NDK 与其他构建系统配合使用](https://developer.android.com/ndk/guides/other_build_systems)

1、编写arm64、armeabi-v7a、x86、x86\_64四个平台的args.gn

#arm64:

is\_debug = false

target\_os = "android"

target\_cpu = "arm64"

ndk\_api = 21

ndk = "/Users/luozhipeng/Mydata/Kdan/Android\_Tools/ndk/android-ndk-r21e"

is\_official\_build=false

is\_component\_build = false

#armeabi-v7a:

is\_debug = false

target\_os = "android"

target\_cpu = "arm"

arm\_version = 7

ndk\_api = 21

ndk = "/mnt/hgfs/Android\_Tools/ndk/android-ndk-r21e"

is\_official\_build=false

is\_component\_build = false

#x86:

is\_debug = false

target\_os = "android"

target\_cpu = "x86"

ndk\_api = 21

ndk = "/mnt/hgfs/Android\_Tools/ndk/android-ndk-r21e"

is\_official\_build=false

is\_component\_build = false

#x86\_64:

is\_debug = false

target\_os = "android"

target\_cpu = "x64"

ndk\_api = 21

ndk = "/mnt/hgfs/Android\_Tools/ndk/android-ndk-r21e"

is\_official\_build=false

is\_component\_build = false

参照PDFium，创建set\_ninja\_env\_for\_arm.sh脚本

export PATH=$PATH:/Users/luozhipeng/Mydata/Ubuntu/Git\_PDFIum/Resources/depot\_tools

bin/gn gen out/armeabi-v7a/

ninja -j 8 -C out/armeabi-v7a

echo "Done!"

Decrepted

关键点，相关模块屏蔽【Android平台的屏蔽另外一套】：



