## Windows7 64位系统编译Chromium指南

1. 代理相关问题，以ssh隧道代理为例：

a.配置bitvise，设置socket5代理。指定socket代理端口号127.0.0.1:xxxx;

b.配置privoxy，设置http代理。

转发socket5：forward-socket5 127.0.0.1:xxxx .

监听端口：listen-addrenss 127.0.0.1:yyyy .

c.命令行下netsh winhttp set proxy 127.0.0.1:yyyy “<local>” 设置winhttp代理。

供gclient下载工具链时使用。

d.环境变量添加http\_proxy，值为127.0.0.1:yyyy;这是因为chromium的

Python下载脚本会用到这个环境变量设置代理。

e.使用git前可以通过git config --global http.proxy=127.0.0.1:yyyy配置代理。

ps.如果有vps，一定直接在vsp上下载chromium源码，打包成tar，在公司网络下直接拉取chromium源码简直是找死。

1. 安装64位windows7，本指南可能不适用于Windows7 32位系统，请自重。
2. 安装VisualStudio2010，设置环境变量GYP\_MSVS\_VERSION=2010
3. 安装VisualStudio2010 SP1。
4. 安装windows 8.0 sdk（不要安装8.1），并修改Windows Kits\8.0\Include\WinRT\asyncinfo.h

注释第66行的class关键字：

65 namespace ABI (namespace Windows { namespace Foundation {

66 enum /\*class\*/ AsyncStatus {

1. 安装June 2010 DirectX SDK。如果有遇到”Error Code:S1023”，则通过控制面板将

Microsoft Visual C++ 2010 x86 Redistributable

Microsoft Visual C++ 2010 x64 Redistributable

都卸载掉，重启后重新安装June 2010 DirectX SDK。

1. 下载depot\_tools，解压，并将depot\_tools根目录添加到Path环境变量。

为了让depot\_tools正常可用，可能需要配置好代理，这个尤为重要，很多人都死在这。

1. 在cmd命令行下执行命令： gclient

该命令会优先下载git、svn、python并解压到depot\_tools目录下，并设置环境变量。

1. 下载chromium源码，可以直接下载chromium的源码压缩包也可以通过gclient直接下载。

8.1、下载源码包。

下载最新的chromium源码压缩包，并且最好用7z解压：

<http://chromium-browser-source.commondatastorage.googleapis.com/chromium_tarball.html>

下载指定版本的chromium源码压缩包：

[http://chromium-browser-source.commondatastorage.googleapis.com/chromium.rXXXXX.tgz](http://chromium-browser-source.commondatastorage.googleapis.com/chromium.rxxx.tgz)

其中rXXXXX表示版本号，比如r197479表示Revision197479。所有可用的压缩包版本号列表页面是：

<http://chromium-browser-source.commondatastorage.googleapis.com/>

8.2、通过gclient下载源码。

创建存放chromium源码的路径，路径不要带空格，比如d:\dev\chromium。

打开cmd命令行，cd到chromium源码路径。

执行gclient config --git-deps <https://chromium.googlesource.com/chromium/src.git>

执行gclient sync 下载chromium源码。

gclient config url 命令在目录下创建了.gclient文件，可以编辑该文件配置更新参数。

该文件的默认内容如下：

|  |
| --- |
| solutions = [  { "name" : "src",  "url" : "https://chromium.googlesource.com/chromium/src.git",  "deps\_file" : ".DEPS.git",  "managed" : True,  "custom\_deps" : {  },  "safesync\_url": "",  },  ] |

具体参数说明详见：

<http://www.chromium.org/developers/how-tos/depottools>

8.3、无论是使用8.1还是8.2的方式获取源码包，都可通过depot\_tools进行更新。

首先需要通过查看下面的Chromium Buildbot waterfall 页面确定chromium状态：

<https://chromium-build.appspot.com/p/chromium/console>

如果源码树状态标识是OPEN，则表示源码可编译。

如果源码树状态标识是CLOSED，则表示源码编译或测试失败，最好不要更新。

其次，可以通过下面的命令行更新到指定版本：

gclient sync --revision src@####

8.4、更新LKGR版代码

如果只是想下载最新无错版：last known good revision (LKGR)，在.gclient下设置：

"safesync\_url": "https://chromium-status.appspot.com/git-lkgr",

配置了这个参数后有下面四点需要注意：

**NOTE:** Because of technical limitations you can only add a safesync\_url *after* you've done your initial sync until [this bug](http://crbug.com/109191) is fixed.

**NOTE:** If you use lkgr, gclient sync will ignore svn revisions you want to sync to using the -r or -H arguments. You need to comment out lkgr from your .gclient first.

**NOTE:** If you use lkgr when running gclient sync, note that rebasing a branch and sticking with lkgr can be a little tricky. Instead of rebasing on origin/master, rebase on to origin/lkgr.

**NOTE:** If you change to lkgr with any open branches, they will get confused and have merge conflicts. If you must change while branches are open, it is a good idea to move all your changes using git cherry-pick to fresh branches based off of the lkgr version of master.

1. 配置MSBuild。

打开下面的目录（将USERNAME替换成当前账户名），如果不存在则创建：

C:\Users\USERNAME\AppData\Local\Microsoft\MSBuild\v4.0

=》创建Microsoft.Cpp.Win32.user.props文件，添加如下内容：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Project DefaultTargets="Build" ToolsVersion="4.0"  xmlns="http://schemas.microsoft.com/developer/msbuild/2003">  <PropertyGroup>  <IncludePath>$(DXSDK\_DIR)\include;$(IncludePath)</IncludePath>  <LibraryPath>$(DXSDK\_DIR)\lib\x86;$(LibraryPath)</LibraryPath>  </PropertyGroup>  </Project> |

=》创建Microsoft.Cpp.x64.user.props文件，添加如下内容：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <Project DefaultTargets="Build" ToolsVersion="4.0"  xmlns="http://schemas.microsoft.com/developer/msbuild/2003">  <PropertyGroup>  <IncludePath>$(DXSDK\_DIR)\include;$(IncludePath)</IncludePath>  <LibraryPath>$(DXSDK\_DIR)\lib\x64;$(LibraryPath)</LibraryPath>  </PropertyGroup>  </Project> |

1. 打开chromium目录，添加chromium.gyp\_env文件，并添加内容：

|  |
| --- |
| {'GYP\_DEFINES': 'component=shared\_library'} |

添加这个配置原因如下：

“On Windows if using the components build, building in debug mode will generally link faster. This is because in debug mode, the linker works incrementally. In release mode, a full link is performed each time.”

1. cmd命令行cd到chromium目录，执行gclient runhooks --force，生成visual studio 2010的sln文件和vcproject文件。

如果生成过程中遇到类似下面的错误：

Warning: unrecognized setting VCLinkerTool/...

则下载下面的文件替换src\tools\gyp\pylib\gyp\MSVSSettings.py

<http://code.google.com/p/gyp/source/browse/trunk/pylib/gyp/MSVSSettings.py?spec=svn1650&r=1650>

1. 编辑src\chrome\chrome.vcxproj，注释掉依赖的test项目。
2. 编辑src\chrome\chrome\_main\_dll.vcxproj，找到下面的代码：

|  |
| --- |
| <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">true</LinkIncremental> |

修改为：

|  |
| --- |
| <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">false</LinkIncremental> |

1. 编译chrome。

第1种方式：使用vs2010编译：打开chrome.sln找到chrome.vcxproj编译之。打开chrome.sln解决方案，比过程比较慢，因为VS IntelliSense和Visual Assist（如果安装了的话）会分析代码，既耗时，有卡，强烈建议关闭VS IntelliSense，下面是加速编译的事项。

第2种方式：打开vs2010的命令行，cd到chrome.vcxproj所在目录，使用下面的命令编译：

msbuild chrome.vcxproj /p:configuration=debug /p:platform=win32 /t:build /m:4

15、经过N小时编译完毕，在src\build\Debug\目录下生成chrome.exe，双击运行，成功。