Windows7 64 位系统编译 Chromium 指南

0、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 {
```

5、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。

6、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。

7、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。

8、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

fanfeilong@gmail.com

下载指定版本的 chromium 源码压缩包：
http://chromium-browser-source.commondatastorage.googleapis.com/chromium.rXXXXX.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 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.

9、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>
```

10、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.”

11、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

12、编辑 src\chrome\chrome.vcxproj，注释掉依赖的 test 项目。

13、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>
```

14、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。

fanfeilong@gmail.com