

Electron build

Code url

<https://github.com/revyos/debian-electron>

<https://github.com/electron/electron>

1. Electron Debianized

Packaging Electron as .deb dependency for applications e.g. VS Code, just like what Arch Linux did. Currently version 22 and 23 is packaged.

Since Electron (together with Chromium) is so huge, and I don't really want to bother using git-buildpackage or something on such package. Also, some preparation done by either Chromium or Electron *needs* to be in a Git repository. So I've settled on only uploading debian/ directory, and creating tarball using scripts from projects' Git repositories.

Coincidentally, this is also how openSUSE packaged [their Electron](#).

2. Building

1. Install yarn:

```
1 # apt install yarnpkg
```

1. Fetch necessary repos (Chromium, Electron):

```
1 $ debian/rules fetch-repos
```

2. Create source directory structure from repos, with pruning and cleaning:

```
1 $ debian/rules create-source
```

3. (Optional) Pack source directory. The result should be a ~737M tarball:

```
1 $ debian/rules pack-tarball
```

4. Start building:

```
1 $ ln -s electron-<major>-<version>.tar.xz electron-<major>_<version>.orig.tar.xz
2 $ cd electron-<major>-<version>/
3 $ cp -r ../debian .
4 $ dpkg-buildpackage -us -uc # or
5 $ sbuild ...
```

Building on riscv64

This package also includes unofficial support for riscv64, as first done in [Arch Linux RISC-V](#).

Note that since Google doesn't build depot_tools for riscv64 (and fortunately it is only used in preparation), you need to:

1. Follow the step 1-3 above
2. Move the tarball to the riscv64 machine and extract it
3. Finish step 4

3. Sidenote

Why put all the code in `src/` directory and not directly in the root?

Some Electron scripts run before packing actually depend on this file structure, and I didn't bother move them into a different name. Also, this is what Arch Linux did in their build scripts (although they do not need something like .orig.tar)

As a side effect, I find this hierarchy easy to let me work on Debian-specific stuff, since there are so many directories and files inside Chromium source's root.

4. need attention

1. Before the first step

```
1 git clone --depth 1 https://github.com/chromium/chromium \
2 --branch $RULES_chromium_ver
```

```

GITLAB-RUNNER
> .cache
> .config
> .gsutil
> .local
> .ssh
> .vpython-root
> .yarn
> 3.0.4-visionfive2
> 7210-visionfive2
> 7220-visionfive2
> 7221-visionfive2
> 7453-visionfive2
> 7459-visionfive2
> builds
> buildsdcad
> buildsdk
> debian-electron
  > debian
    > patches
    > scripts
      $ create-source.sh
      $ fetch-repos.sh
      $ pack-tarball.sh
      $ unbundle.py
    > source
  > changelog
  > control
  > copyright
  > electron.install.in
  > electron.links.in
  > files-excluded.txt
  > README.source

debian-electron > debian > scripts > $ fetch-repos.sh
1  #!/bin/bash
2
3  _dir=repos
4  if [ ! -d $_dir ]; then
5  |   mkdir $_dir
6  | fi
7  cd $_dir
8
9  git clone https://salsa.debian.org/chromium-team/chromium.git \
10  |   debian-chromium \
11  |   --branch debian/$RULES_deb_chromium_ver
12
13
14  git clone https://gitlab.archlinux.org/archlinux/packaging/packages/electron$RULES_electron_ver_major.git \
15  |   archlinux-electron$RULES_electron_ver_major
16
17
18  git clone https://chromium.googlesource.com/chromium/tools/depot_tools.git
19
20  git clone https://github.com/electron/electron.git \
21  |   --branch v$RULES_electron_ver
22
23  git clone --depth 1 https://github.com/chromium/chromium \
24  |   --branch $RULES_chromium_ver
25
26  # export RULES_electron_ver_major = 23
27  # export RULES_pkgname = electron-23
28  # export RULES_chromium_ver = 110.0.5481.208
29  # export RULES_deb_chromium_ver = 111.0.5563.110-1_deb11u1
30  # export RULES_electron_ver = 23.3.11

```

2. Before the two step

```

1  # ( cd electron && yarnpkg install --frozen-lockfile )
2  echo "*****start*****"
3  (cd electron && unset NODE_OPTIONS && yarnpkg install --frozen-lockfile --ignor
4  echo "*****end*****"

```

```
> .vpython-root
> .yarn
> 3.0.4-visionfive2
> 7210-visionfive2
> 7220-visionfive2
> 7221-visionfive2
> 7453-visionfive2
> 7459-visionfive2
> builds
> buildsdcard
> buildsdk
> debian-electron
  > debian
    > patches
    > scripts
      $ create-source.sh
        $ fetch-repos.sh
        $ pack-tarball.sh
        $ unbundle.py
      > source
      > changelog
      > control
      > copyright
      > electron.install.in
      > electron.links.in
      > files-excluded.txt
      > README.source
      > rules
    > electron
    > electron-23-23.3.11

44 python3 build/util/lastchange.py -m GPU_LISTS_VERSION \
45 --revision-id-only --header gpu/config/gpu_lists_version.h
46 python3 build/util/lastchange.py -m SKIA_COMMIT_HASH \
47 -s third_party/skia --header skia/ext/skia_commit_hash.h
48 python3 build/util/lastchange.py \
49 -s third_party/dawn --revision gpu/webgpu/DAWN_VERSION
50 python3 tools/update_pgo_profiles.py --target=linux update \
51 --gs-url-base=chromium-optimization-profiles/pgo_profiles
52 download_from_google_storage --no_resume --extract --no_auth \
53 --bucket chromium-nodejs -s third_party/node/node_modules.tar.gz.sha1
54
55 export PATH=$_oldpath
56 unset DEPOT_TOOLS_UPDATE
57
58 # apply electron patches beforehand since they need git to apply
59 ( cd .. && \
60   src/electron/script/apply_all_patches.py \
61   src/electron/patches/config.json )
62
63 # Same as yarn install
64 # ( cd electron && yarnpkg install --frozen-lockfile )
65 echo "*****start*****"
66 (cd electron && unset NODE_OPTIONS && yarnpkg install --frozen-lockfile --ignore-engines)
67 echo "*****end*****"
68
69 set +e
70
71
72
73 # remove unused files (mainly to shrink size)
74 readarray -t _files_excluded < ../debian/files-excluded.txt
75 for _f in "${_files_excluded[@]"; do
76   if [ -n "$_f" ] && [ "$_f" != "#*" ]; then
77     for _g in $(bash -O dotglob -O globstar -c "echo $_f"); do
78       rm -rf $_g
```

3. before the four step

```
1 def RunNode(cmd_parts, stdout=None):
2     cmd = [GetBinaryPath()] + cmd_parts
3     process = subprocess.Popen(
4         cmd, cwd=os.getcwd(), stdout=subprocess.PIPE, stderr=subprocess.PIPE,
5         universal_newlines=True)
6     while True:
7         stdout, stderr = process.communicate()
8         returncode = process.returncode
9         if returncode == 0:
10             print("success")
11         if returncode != 0:
12             errors = stderr.split('\n')
13             for errorline in errors :
14                 if 'FAILED' in errorline:
15                     print('error %s %s' % (cmd, errline))
16             print(returncode)
17             #raise RuntimeError('Command \'%s\' failed\n%s' %(' '.join(cmd), err))
18         if stdout == '' and process.poll() != None:
19             break;
20     # stdout, stderr = process.communicate()
21
22     # if process.returncode != 0:
23     #     # Handle cases where stderr is empty, even though the command failed, for
24     #     # example https://github.com/microsoft/TypeScript/issues/615
25     #     err = stderr if len(stderr) > 0 else stdout
```

```

26     # raise RuntimeError('Command \'%s\' failed\n%s' % (' '.join(cmd), err))
27
28     # return stdout
29

```

```

EXPLORER
SIXUE
> gradle_wrapper
> grpc
> grpc-java
> gvr-android-keyboard
> gvr-android-sdk
> hamcrest
> harfbuzz-ng
> highway
> hunspell
> hunspell_dictionaries
> hyphenation-patterns
> iaccessible2
> icjjpeg
> icu
> icu4j
> ijar
> inspector_protocol
> ipc
> isimpledom
> jacoco
> javalang
> jinja2
> js_code_coverage
> jstemplate
> junit
> khronos
> lcov
> leveledb
> libaddressinput
> libaom
> libavif
> libbrotli
> libFuzzer
> libgavl
> libipp
> libjingle_xmpp
> liblouis
> libphonenumbers
> libprotobuf-mutator
> libsecret
> libsrtp
> libsync

lastchange.py
create-source.sh M
node.py
fetch-repos.sh M
node.py

6 from os import path as os_path
7 import platform
8 import subprocess
9 import sys
10 import os
11
12
13 def GetBinaryPath():
14     darwin_name = ('node-darwin-arm64' if platform.machine() == 'arm64' else
15                   'node-darwin-x64')
16     return os_path.join(os_path.dirname(__file__), *{
17         'Darwin': ('mac', darwin_name, 'bin', 'node'),
18         'Linux': ('linux', 'node-linux-x64', 'bin', 'node'),
19         'Windows': ('win', 'node.exe'),
20     }[platform.system()])
21
22
23 def RunNode(cmd_parts, stdout=None):
24     cmd = [GetBinaryPath()] + cmd_parts
25     process = subprocess.Popen(
26         cmd, cwd=os.getcwd(), stdout=subprocess.PIPE, stderr=subprocess.PIPE,
27         universal_newlines=True)
28     while True:
29         stdout, stderr = process.communicate()
30         returncode = process.returncode
31         if returncode == 0:
32             print("success")
33         if returncode != 0:
34             errors = stderr.split('\n')
35             for errorline in errors:
36                 if 'FAILED' in errorline:
37                     print('error %s %s' % (cmd, errorline))
38             print(returncode)
39             #raise RuntimeError('Command \'%s\' failed\n%s' % (' '.join(cmd), err))
40         if stdout == '' and process.poll() != None:
41             break;
42         # stdout, stderr = process.communicate()
43
44         # if process.returncode != 0:
45         #     # Handle cases where stderr is empty, even though the command failed, for
46         #     # example https://github.com/microsoft/TypeScript/issues/615
47         #     err = stderr if len(stderr) > 0 else stdout
48         #     raise RuntimeError('Command \'%s\' failed\n%s' % (' '.join(cmd), err))
49
50     # return stdout
51
52 if __name__ == '__main__':
53     RunNode(sys.argv[1:])
54

```

```

1 apt install dpkg-dev debhelper clang-15 clang-format-15 lld-15 generate-ninja
ninja-build openjdk-11-jdk-headless npm nodejs elfutils brotli pkg-config
gperf libopus-dev liblcms2-dev libjsoncpp-dev libopenjp2-7-dev libxslt1-dev
libwoff-dev libflac-dev libopenh264-dev libre2-dev libevent-dev libdouble-
conversion-dev libxnvctrl-dev libsnappy-dev libminizip-dev libxshmfence-dev
libjpeg-dev libicu-dev rapidjson-dev libcups2-dev zlib1g-dev libglib2.0-dev
libdbus-1-dev libgdk-pixbuf-2.0-dev libnss3-dev libpulse-dev libxkbcommon-dev
libatspi2.0-dev libatk1.0-dev libdrm-dev libfreetype-dev libatk-bridge2.0-dev
libgtk-3-dev libgbm-dev libpipewire-0.3-dev mesa-common-dev libnotify-dev

```

```
libx11-xcb-dev libcurl4-openssl-dev libpci-dev libasound2-dev libkrb5-dev  
libnghttp2-dev libc-ares-dev libva-dev  
2 dpkg-buildpackage -us -uc -b
```