# 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:

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

4. Start building:

```
$ ln -s electron-<major>-<version>.tar.xz electron-<major>_<version>.orig.tar.xz
$ cd electron-<major>-<version>/
$ cp -r ../debian .
$ dpkg-buildpackage -us -uc  # or
$ 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?

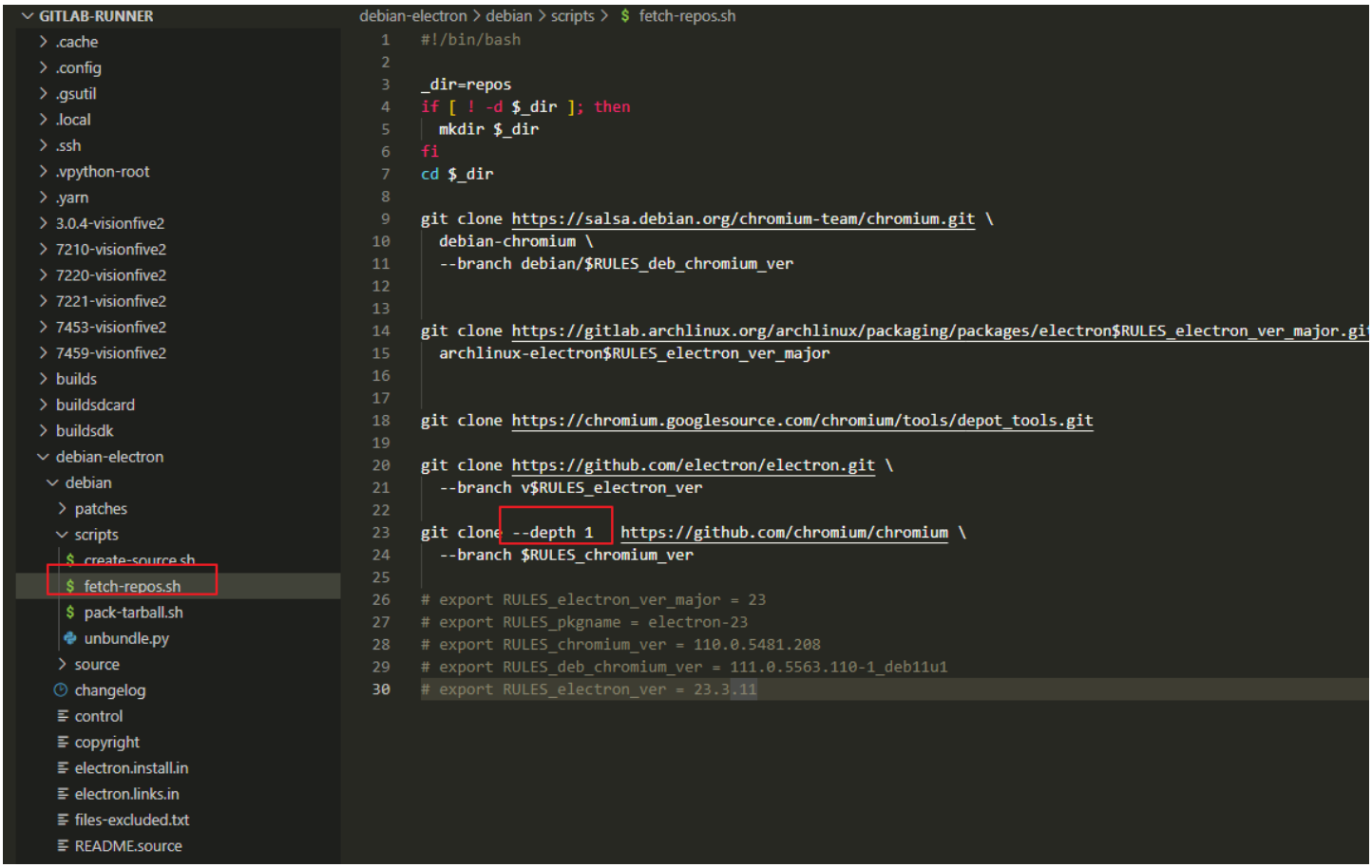
Some Electron scripts run before packing actually depend on this file structure, and I didn't bother move them into a different name. Also, this is what Arch Linux did in their build scripts (although they do not need something like .orig.tar)

As a side effect, I find this hierachy 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

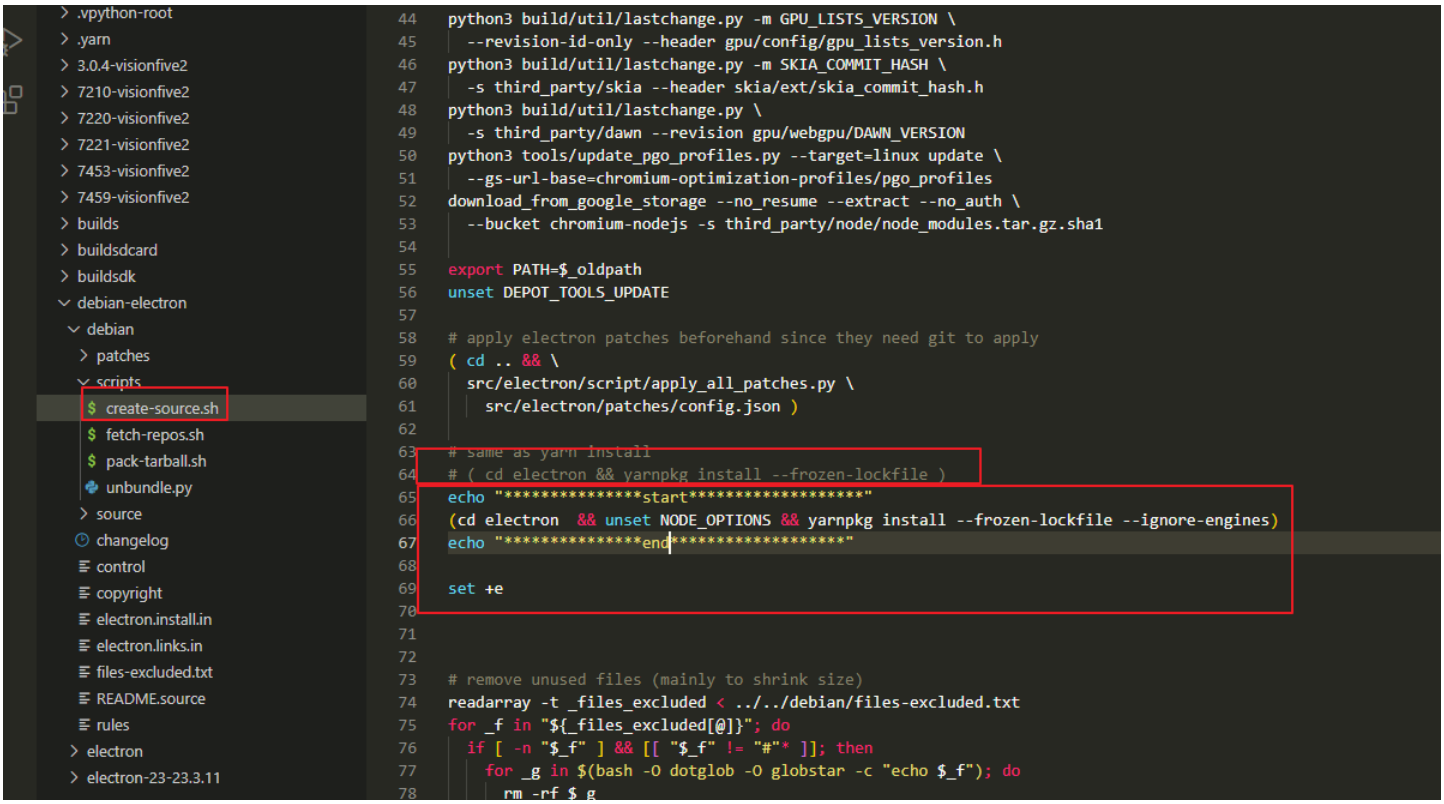
1. Before the first step

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




2. Before the two step

```
# ( cd electron && yarnpkg install --frozen-lockfile )
echo "**********start**********"
(cd electron  && unset NODE_OPTIONS && yarnpkg install --frozen-lockfile --ignor
echo "**********end**********"
```

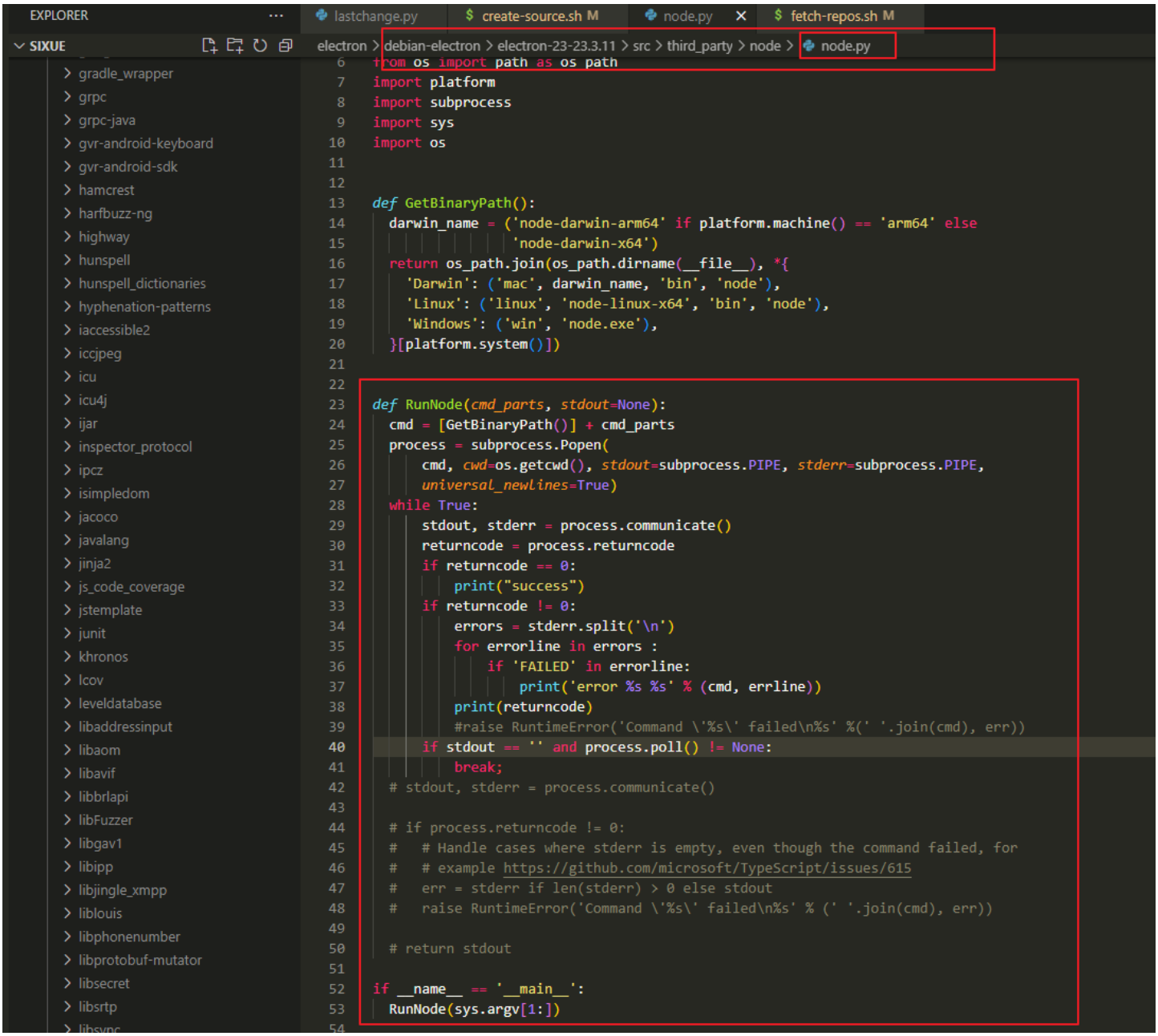



3. before the four step

```
def RunNode(cmd_parts, stdout=None):
cmd = [GetBinaryPath()] + cmd_parts
process = subprocess.Popen(
    cmd, cwd=os.getcwd(), stdout=subprocess.PIPE, stderr=subprocess.PIPE,
    universal_newlines=True)
while True:
    stdout, stderr = process.communicate()
    returncode = process.returncode
    if returncode == 0:
        print("success")
    if returncode != 0:
        errors = stderr.split('\n')
        for errorline in errors :
            if 'FAILED' in errorline:
                print('error %s %s' % (cmd, errline))
        print(returncode)
        #raise RuntimeError('Command \'%s\' failed\n%s' %(' '.join(cmd), err))
    if stdout == '' and process.poll() != None:
        break;
# stdout, stderr = process.communicate()

# if process.returncode != 0:
#   # Handle cases where stderr is empty, even though the command failed, for
#   # example https://github.com/microsoft/TypeScript/issues/615
#   err = stderr if len(stderr) > 0 else stdout
```

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




```
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
dpkg-buildpackage -us -uc -b
```