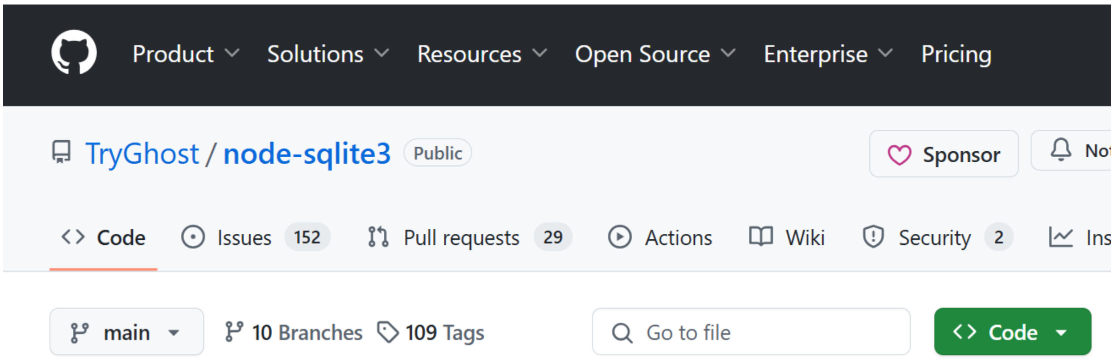
前置要求：ubuntu22.04环境，安装node.js，下载并编译代码仓库electron代码

一、下载node-sqlite3源码到Ubuntu编译环境下（本文使用的是5.1.7版本，其他版本类似）

GitHub地址：https://github.com/TryGhost/node-sqlite3

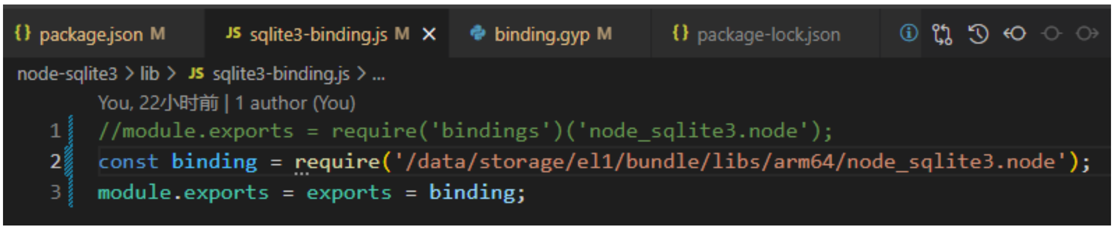


二、打开node-sqlite3源码并进行适当修改

a) 修改文件node-sqlite3/lib/sqlite3-binding.js为如下图所示：

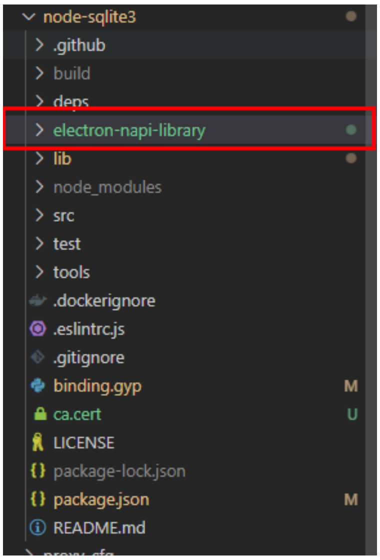
const binding = require('/data/storage/el1/bundle/libs/arm64/node_sqlite3.node');

module.exports = exports = binding;

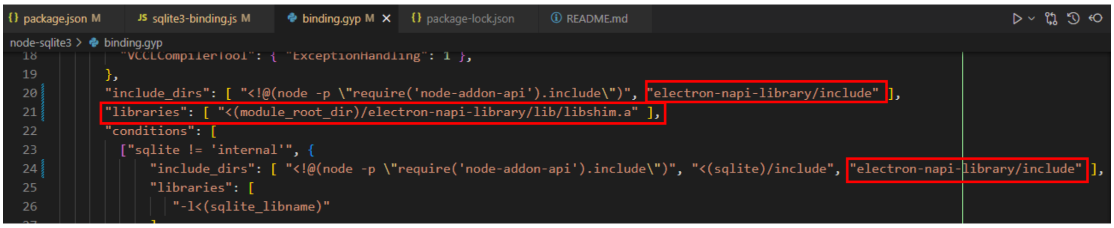

package.json M
sqlite3-binding.js M
binding.gyp M
package-lock.json
node-sqlite3 > lib > sqlite3-binding.js > ...
You, 22小时前 | 1 author (You)

```
//module.exports = require('bindings')('node_sqlite3.node');
const binding = require('/data/storage/el1/bundle/libs/arm64/node_sqlite3.node');
module.exports = exports = binding;
```

b) 将附件中的electron-napi-library文件夹拷贝到node-sqlite3目录下（如下图所示）



c) 修改文件node-sqlite3/binding.gyp



```
        "VCCLCompilerTool": { "ExceptionHandling": 1 },
      },
      "include_dirs": [ "<!@(node -p \"require('node-addon-api').include\")", "electron-napi-library/include" ],
      "libraries": [ "<(module_root_dir)/electron-napi-library/lib/libshim.a" ],
      "conditions": [
        ["sqlite != 'internal'", {
          "include_dirs": [ "<!@(node -p \"require('node-addon-api').include\")", "<(sqlite)/include", "electron-napi-library/include" ],
          "libraries": [
            "-l<(sqlite_libname)"
```

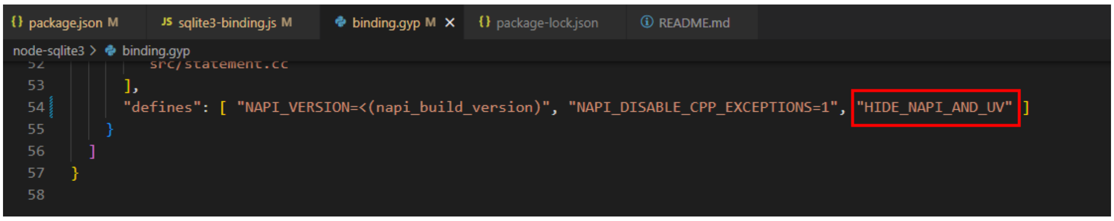
如上图所示：

在"include_dirs"中增加："electron-napi-library/include"；

在"include_dirs"下新增一行：

"libraries": ["<(module_root_dir)/electron-napi-library/lib/libshim.a"],

在"conditions"的"include_dirs"中增加："electron-napi-library/include"

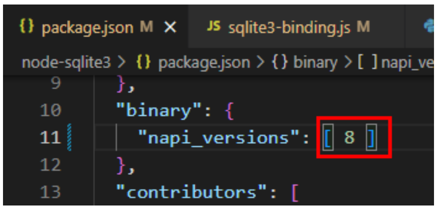


```
          src/statement.cc
        ],
        "defines": [ "NAPI_VERSION=<(napi_build_version)", "NAPI_DISABLE_CPP_EXCEPTIONS=1", "HIDE_NAPI_AND_UV" ]
      }
    ]
  }
```

如上图所示：

在"defines"中增加："HIDE_NAPI_AND_UV"

d)　修改文件node-sqlite3/package.json



```
  },
  "binary": {
    "napi_versions": [ 8 ]
  },
  "contributors": [
```

如上图所示：

将"napi_versions"修改为8

三、配置编译环境

Electron和addon必须用相同的编译工具链来编译

例如，electron源码路径是：/opt/code/dev_chromium/dev_electron/

那么，编译工具链的路径就是：

/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/

根据自己的编译环境设置环境变量：

```
export CC="/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/clang --target=aarch64-linux-ohos"
export CXX="/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/clang++ --target=aarch64-linux-ohos"

export LD="/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/lld --target=aarch64-linux-ohos"

export STRIP="/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/llvm-strip"

export RANLIB="/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/llvm-ranlib"

export OBJDUMP="/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/llvm-objdump"

export OBJCOPY="/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/llvm-objcopy"

export NM="/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/llvm-nm"

export AR="/opt/code/dev_chromium/dev_electron/src/ohos_sdk/openharmony/native/llvm/bin/llvm-ar"

export CFLAGS="-fPIC -D__MUSL__=1"

export CXXFLAGS="-fPIC -D__MUSL__=1"
```

四、编译

1. 进入node-sqlite3目录并配置华为镜像仓库

```
cd ~/node-sqlite3

npm config set registry http://mirrors.tools.huawei.com/npm/
```

2. 开始编译

```
npm install --verbose --build-from-source --runtime=electron --debug --target=18.18.2 --dist-url=https://electronjs.org/headers
```

注：“--target=18.18.2”该项为编译机nodejs的版本。编译时需要更换为自己本地的版本；

执行“node –version”命令可查询当前编译机的nodejs版本

编译报错解决：

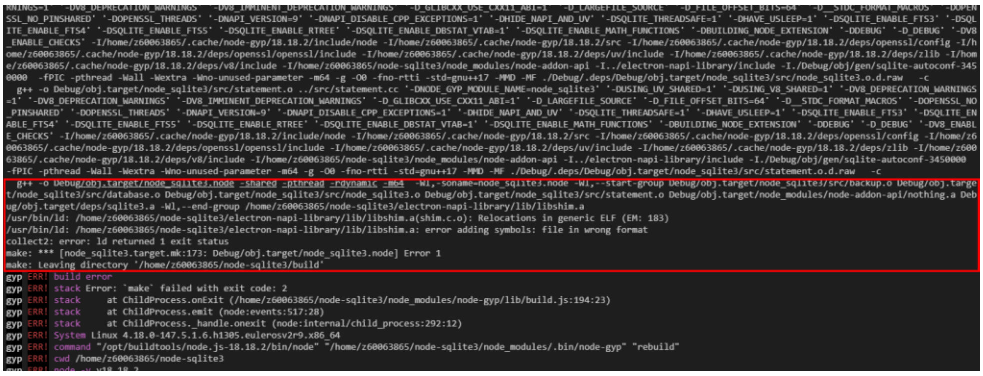
(一) 如果在编译时报类似于：

audit error FetchError: request to https://registry.npmjs.org/-/npm/v1/security/audits/quick failed, reason: self-signed certificate in certificate chain；

的错误，可以尝试配置npm忽略SSL证书验证来解决该问题：

```
npm config set strict-ssl false
```

(二) 如果报如下图所示的错误：



```
RNINGS=1' '-DV8_DEPRECATION_WARNINGS' '-DV8_IMMINENT_DEPRECATION_WARNINGS' '-D_GLIBCXX_USE_CXX11_ABI=1' '-D_LARGEFILE_SOURCE' '-D_FILE_OFFSET_BITS=64' '-D__STDC_FORMAT_MACROS' '-DOPEN
SSL_NO_PINSHARED' '-DOPENSSL_THREADS' '-DNAPI_VERSION=9' '-DNAPI_DISABLE_CPP_EXCEPTIONS=1' '-DHIDE_NAPI_AND_UV' '-DSQLITE_THREADSAFE=1' '-DHAVE_USLEEP=1' '-DSQLITE_ENABLE_FTS3' '-DSQL
ITE_ENABLE_FTS4' '-DSQLITE_ENABLE_FTS5' '-DSQLITE_ENABLE_RTREE' '-DSQLITE_ENABLE_DBSTAT_VTAB=1' '-DSQLITE_ENABLE_MATH_FUNCTIONS' '-DBUILDING_NODE_EXTENSION' '-DDEBUG' '-D_DEBUG' '-DV8
_ENABLE_CHECKS' -I/home/z60063865/.cache/node-gyp/18.18.2/include/node -I/home/z60063865/.cache/node-gyp/18.18.2/src -I/home/z60063865/.cache/node-gyp/18.18.2/deps/openssl/config -I/h
ome/z60063865/.cache/node-gyp/18.18.2/deps/openssl/openssl/include -I/home/z60063865/.cache/node-gyp/18.18.2/deps/uv/include -I/home/z60063865/.cache/node-gyp/18.18.2/deps/zlib -I/hom
e/z60063865/.cache/node-gyp/18.18.2/deps/v8/include -I/home/z60063865/node-sqlite3/node_modules/node-addon-api -I../electron-napi-library/include -I./Debug/obj/gen/sqlite-autoconf-345
0000  -fPIC -pthread -Wall -Wextra -Wno-unused-parameter -m64 -g -O0 -fno-rtti -std=gnu++17 -MMD -MF ./Debug/.deps/Debug/obj.target/node_sqlite3/src/node_sqlite3.o.d.raw   -c
  g++ -o Debug/obj.target/node_sqlite3/src/statement.o ../src/statement.cc '-DNODE_GYP_MODULE_NAME=node_sqlite3' '-DUSING_UV_SHARED=1' '-DUSING_V8_SHARED=1' '-DV8_DEPRECATION_WARNINGS
=1' '-DV8_DEPRECATION_WARNINGS' '-DV8_IMMINENT_DEPRECATION_WARNINGS' '-D_GLIBCXX_USE_CXX11_ABI=1' '-D_LARGEFILE_SOURCE' '-D_FILE_OFFSET_BITS=64' '-D__STDC_FORMAT_MACROS' '-DOPENSSL_NO
_PINSHARED' '-DOPENSSL_THREADS' '-DNAPI_VERSION=9' '-DNAPI_DISABLE_CPP_EXCEPTIONS=1' '-DHIDE_NAPI_AND_UV' '-DSQLITE_THREADSAFE=1' '-DHAVE_USLEEP=1' '-DSQLITE_ENABLE_FTS3' '-DSQLITE_EN
ABLE_FTS4' '-DSQLITE_ENABLE_FTS5' '-DSQLITE_ENABLE_RTREE' '-DSQLITE_ENABLE_DBSTAT_VTAB=1' '-DSQLITE_ENABLE_MATH_FUNCTIONS' '-DBUILDING_NODE_EXTENSION' '-DDEBUG' '-D_DEBUG' '-DV8_ENABL
E_CHECKS' -I/home/z60063865/.cache/node-gyp/18.18.2/include/node -I/home/z60063865/.cache/node-gyp/18.18.2/src -I/home/z60063865/.cache/node-gyp/18.18.2/deps/openssl/config -I/home/z6
0063865/.cache/node-gyp/18.18.2/deps/openssl/openssl/include -I/home/z60063865/.cache/node-gyp/18.18.2/deps/uv/include -I/home/z60063865/.cache/node-gyp/18.18.2/deps/zlib -I/home/z600
63865/.cache/node-gyp/18.18.2/deps/v8/include -I/home/z60063865/node-sqlite3/node_modules/node-addon-api -I../electron-napi-library/include -I./Debug/obj/gen/sqlite-autoconf-3450000
-fPIC -pthread -Wall -Wextra -Wno-unused-parameter -m64 -g -O0 -fno-rtti -std=gnu++17 -MMD -MF ./Debug/.deps/Debug/obj.target/node_sqlite3/src/statement.o.d.raw   -c
  g++ -o Debug/obj.target/node_sqlite3.node -shared -pthread -rdynamic -m64  -Wl,-soname=node_sqlite3.node -Wl,--start-group Debug/obj.target/node_sqlite3/src/backup.o Debug/obj.targe
t/node_sqlite3/src/database.o Debug/obj.target/node_sqlite3/src/node_sqlite3.o Debug/obj.target/node_sqlite3/src/statement.o Debug/obj.target/node_modules/node-addon-api/nothing.a Deb
ug/obj.target/deps/sqlite3.a -Wl,--end-group /home/z60063865/node-sqlite3/electron-napi-library/lib/libshim.a
/usr/bin/ld: /home/z60063865/node-sqlite3/electron-napi-library/lib/libshim.a(shim.c.o): Relocations in generic ELF (EM: 183)
/usr/bin/ld: /home/z60063865/node-sqlite3/electron-napi-library/lib/libshim.a: error adding symbols: file in wrong format
collect2: error: ld returned 1 exit status
make: *** [node_sqlite3.target.mk:173: Debug/obj.target/node_sqlite3.node] Error 1
make: Leaving directory '/home/z60063865/node-sqlite3/build'
gyp ERR! build error
gyp ERR! stack Error: `make` failed with exit code: 2
gyp ERR! stack     at ChildProcess.onExit (/home/z60063865/node-sqlite3/node_modules/node-gyp/lib/build.js:194:23)
gyp ERR! stack     at ChildProcess.emit (node:events:517:28)
gyp ERR! stack     at ChildProcess._handle.onexit (node:internal/child_process:292:12)
gyp ERR! System Linux 4.18.0-147.5.1.6.h1305.eulerosv2r9.x86_64
gyp ERR! command "/opt/buildtools/node.js-18.18.2/bin/node" "/home/z60063865/node-sqlite3/node_modules/.bin/node-gyp" "rebuild"
gyp ERR! cwd /home/z60063865/node-sqlite3
gyp ERR! node -v v18.18.2
```

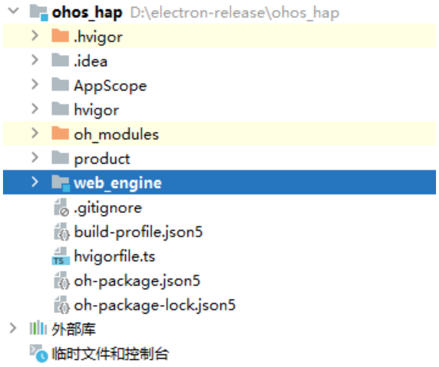
这时请检查上面编译工具链的环境变量的配置，是否配置成功。

(三) 如果出现下载node-v18.0.0-headers.tar.gz失败的报错，可能是网络问题，设置网络代理让linux计算云可以访问外网，再次运行编译命令即可。

五、在Electron工程中使用编译好的sqlite3

A. 从Electron源码中下载DevEco工程到本地Windows系统上并打开（如下图所示）；

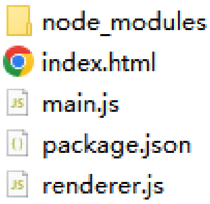
路径：~/chromium-electron-release/src/ohos/app/ohos_hap



```
ohos_hap D:\electron-release\ohos_hap
  > .hvigor
  > .idea
  > AppScope
  > hvigor
  > oh_modules
  > product
  > web_engine
    .gitignore
    build-profile.json5
    hvigorfile.ts
    oh-package.json5
    oh-package-lock.json5
> 外部库
  临时文件和控制台
```

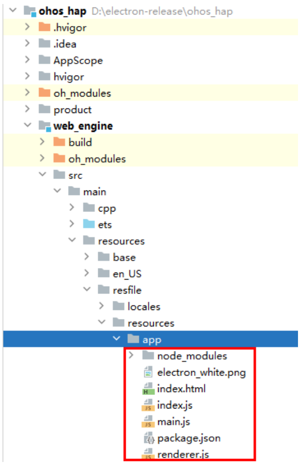
B. 将Electron项目需求的.so 等文件从Ubuntu编译机拷贝到本地的DevEco工程中（具体文件请参考《Electron鸿蒙化指导文档》，在这里不再重复叙述）

C. 将附件electron-example.zip解压后的文件（如下图）覆盖拷贝到工程



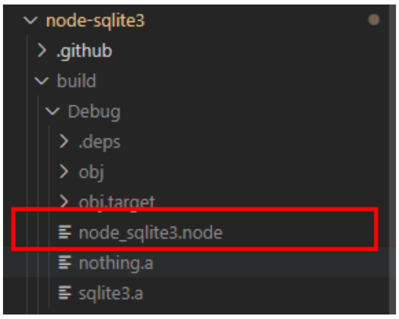
```
node_modules
index.html
main.js
package.json
renderer.js
```

ohos_hap/web_engine/src/main/resources/resfile/resources/app/下（如下图）：



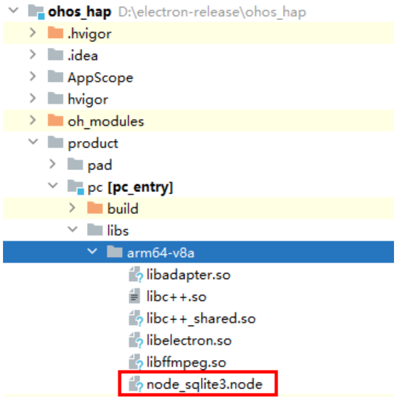
```
ohos_hap D:\electron-release\ohos_hap
  .hvigor
  .idea
  AppScope
  hvigor
  oh_modules
  product
  web_engine
    build
    oh_modules
    src
      main
        cpp
        ets
        resources
          base
          en_US
          resfile
            locales
            resources
              app
                node_modules
                electron_white.png
                index.html
                index.js
                main.js
                package.json
                renderer.js
```

D. 在Ubuntu系统中的node-sqlite3/build/Debug/node_sqlite3.node目录下找到node_sqlite3.node（如下图），把它放置在你的Electron工程如下目录：ohos_hap\product\pc\libs\arm64-v8a\



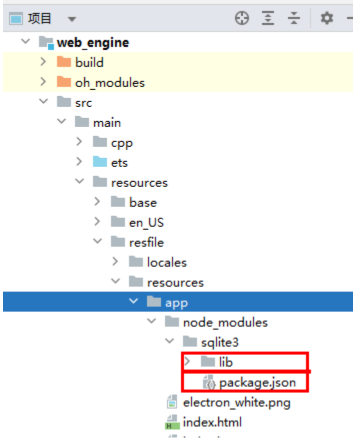
```
node-sqlite3
  .github
  build
    Debug
      .deps
      obj
      obj.target
      node_sqlite3.node
      nothing.a
      sqlite3.a
```

上图为Ubuntu，下图为Windows



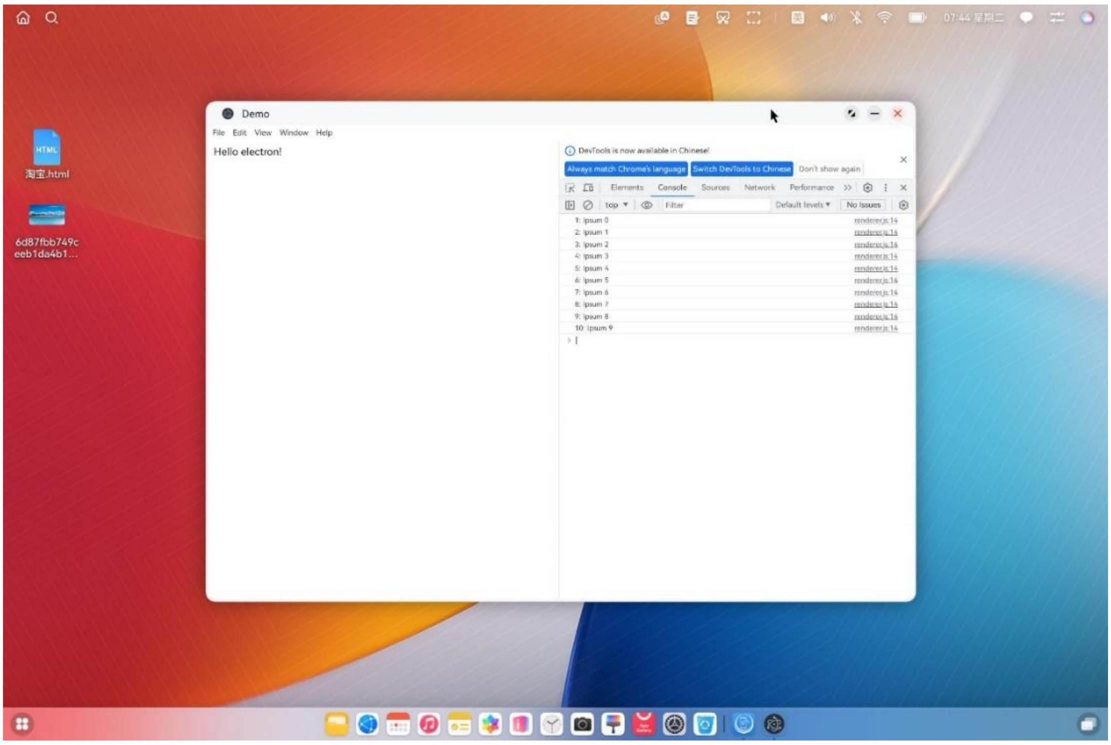
```
ohos_hap D:\electron-release\ohos_hap
  .hvigor
  .idea
  AppScope
  hvigor
  oh_modules
  product
    pad
    pc [pc_entry]
      build
      libs
        arm64-v8a
          libadapter.so
          libc++.so
          libc++_shared.so
          libelectron.so
          libffmpeg.so
          node_sqlite3.node
```

E. 将ubuntu系统中的node-sqlite3/lib文件夹和node-sqlite3/package.json文件，拷贝到ohos_hap\web_engine\src\main\resources\resfile\resources\app\node_modules\sqlite3目录下



```
项目
web_engine
  build
  oh_modules
  src
    main
      cpp
      ets
      resources
        base
        en_US
        resfile
          locales
          resources
            app
              node_modules
                sqlite3
                  lib
                  package.json
              electron_white.png
              index.html
```

F.　构建并安装hap包，打开electron，如果console里打印出了如下的日志，则调用成功：



Hello electron!

```
1: ipsum 0                                   renderer.js:14
2: ipsum 1                                   renderer.js:14
3: ipsum 2                                   renderer.js:14
4: ipsum 3                                   renderer.js:14
5: ipsum 4                                   renderer.js:14
6: ipsum 5                                   renderer.js:14
7: ipsum 6                                   renderer.js:14
8: ipsum 7                                   renderer.js:14
9: ipsum 8                                   renderer.js:14
10: ipsum 9                                  renderer.js:14
>
```