



**银河麒麟桌面操作系统 V10
Electron 应用开发者打包指南**

麒麟软件有限公司
生态与技术服务中心
2023 年 08 月 18 日



版本说明

版本号	版本说明	作者	日期	变更内容
V1.0	首次发布	吴兆惠	2022-03-24	首次创建
V1.1	内容变更	吴兆惠	2022-04-07	增加 mips 架构下使用 electron-builder 工具打包的方法
V1.2	模板变更	刘佳鑫	2022-11-30	模板变更
V2.0	内容变更	吴兆惠	2023-01-10	修改龙芯架构 electron 安装方法、新增本地 electron 镜像源安装方法、简化章节、修改 fpm 安装方法等。
V3.0	内容变更	吴兆惠	2023-8-18	改用 nvm 来安装管理 Nodejs、添加 nvm 离线安装 Nodejs 方法、修改龙芯 Mips64el 和 Loongarch64 架构可用的 Nodejs 版本和 Electron 版本、V10 系统使用 gem 安装 fpm 报错解决方法、优化目录结构等。



目 录

1 目的	5
2 X86_64 架构	6
2.1 安装开发基础环境	6
2.1.1 安装 Nodejs	6
2.1.2 从源中安装 git	7
2.1.3 设置 npm 仓库源(可选)	7
2.1.4 设置 electron 镜像源(可选)	7
2.2 项目开发	8
2.2.1 下载 electron-quick-start 项目	8
2.2.2 修改 electron 版本	8
2.2.3 安装 electron 及依赖包	8
2.3 启动项目	9
2.4 打包	9
2.4.1 安装 electron-builder	9
2.4.2 添加打包命令	10
2.4.3 添加应用图标	12
2.4.4 打 deb 包	12
2.4.5 修改 deb 包	13
2.5 验包	13
2.5.1 安装	13
2.5.2 启动	14
3 ARM64 架构	15
3.1 安装开发基础环境	15
3.1.1 安装 Nodejs	15
3.1.2 从源中安装 git	15
3.1.3 设置 npm 仓库源(可选)	15
3.1.4 设置 electron 镜像源（可选）	15
3.2 项目开发	15
3.2.1 下载 electron-quick-start 项目	15
3.2.2 修改 electron 版本	15
3.2.3 安装 electron 及依赖包	15
3.3 启动项目	15
3.4 打包	16
3.4.1 安装 electron-builder	16
3.4.2 添加打包命令	16
3.4.3 添加应用图标	16
3.4.4 安装 fpm	16
3.4.5 打 deb 包	18
3.4.6 修改 deb 包	18
3.5 验包	18
3.5.1 安装	18
3.5.2 启动	18



4 MIPS64EL 架构	20
4.1 安装开发基础环境	20
4.1.1 安装 Nodejs	20
4.1.2 从源中安装 git	22
4.1.3 设置 npm 仓库源(必选)	22
4.1.4 设置 electron 镜像源(必选)	22
4.2 项目开发	23
4.2.1 下载 electron-quick-start 项目	23
4.2.2 修改 electron 版本	23
4.2.3 安装 electron 及依赖包	23
4.3 启动项目	24
4.4 打包	25
4.4.1 安装 electron-builder	25
4.4.2 添加打包命令	25
4.4.3 添加应用图标	25
4.4.4 安装 fpm	26
4.4.5 打 deb 包	26
4.4.6 修改 deb 包	27
4.5 验包	27
4.5.1 安装	27
4.5.2 启动	27
5 LOONGARCH64 架构	29
5.1 安装开发基础环境	29
5.1.1 安装 Nodejs	29
5.1.2 从源中安装 git	29
5.1.3 设置 npm 仓库源(必选)	29
5.1.4 设置 electron 镜像源(必选)	29
5.2 项目开发	30
5.2.1 下载 electron-quick-start 项目	30
5.2.2 修改 electron 版本	30
5.2.3 安装 electron 及依赖包	30
5.3 启动项目	31
5.4 打包	31
5.4.1 安装 electron-builder	31
5.4.2 添加打包命令	31
5.4.3 添加应用图标	31
5.4.4 安装 fpm	31
5.4.5 打 deb 包	31
5.4.6 修改 deb 包	32
5.5 验包	32
5.5.1 安装	32
5.5.2 启动	32
6 附录	34
6.1 搭建 ELECTRON 本地镜像源	34



6.1.1 创建 electron 镜像源目录	34
6.1.2 将 electron 包放入上述目录	34
6.1.3 启动 electron 镜像源	34
6.1.4 设置 electron 镜像源(必选)	35

1 目的

本指南使用 Electron 官方提供的简单模板项目 `electron-quick-start`，演示在银河麒麟桌面操作系统各个架构开发 Electron 应用时，安装开发基础环境、仓库源设置、Electron 及依赖安装到打包成符合银河麒麟桌面操作系统 V10 软件商店上架规范的 deb 包的方法；由于各个架构中依赖包存在差异，所以各个架构单独进行演示编写，请参考对应架构进行打包。

本文中提到的银河麒麟桌面操作系统 V10 包括 V10-4.4 内核、V10(SP1)-内核 5.4 和 V10(SP1-HWE)-5.10 内核，下文中的 V10 系统指的是老 V10 系统，V10(SP1)系统指的是 V10(SP1)和 V10(SP1-HWE)。

本文中只是介绍了众多打包方式中比较常见的一种，仅供参考使用。

Electron 是一个使用 JavaScript、HTML 和 CSS 构建跨平台桌面应用程序的框架。它通过使用 Node.js 和 Chromium 的渲染引擎完成跨平台的桌面 GUI 应用程序的开发。Electron 兼容 Mac、Windows 和 Linux，可以构建出三个平台的应用程序。

npm 是 JavaScript 世界的包管理工具，并且是 Node.js 平台的默认包管理工具。通过 npm 可以安装、共享、分发代码，管理项目依赖关系。npm 是随同 Nodejs 一起安装的包管理工具，我们经常使用它来下载第三方包到本地。但在使用 npm 过程很多人估计都知道，在国内下载第三方包的速度极其之慢。国内推荐使用淘宝 npm 镜像，它是一个完整 npmjs.org 镜像，可以用此代替官方版本，同步频率目前为 10 分钟一次以保证尽量与官方服务同步。

nvm 是 Node.js 的版本管理器，可以简单操作 Nodejs 版本的切换、安装、查看等功能，可以满足一个电脑中安装多个 Nodejs 版本，当我们想使用哪个版本就切换成哪个版本，而 nvm 则是提供切换 Nodejs 版本的工具。

注：

npm 官方网站：<https://www.npmjs.com/>

npm 官方源：<https://registry.npmjs.org/>

npm 淘宝源：<https://registry.npm.taobao.org/>

2 X86_64 架构

2.1 安装开发基础环境

2.1.1 安装 Nodejs

麒麟大部分系统默认没有预装 Nodejs，需要从源里安装，V10 源里的 node 版本为 v4.2.6，npm 版本为 v3.5.2。V10(SP1)源里的 node 版本为 v10.19.0，npm 版本为 v6.14.4。实际开发过程中可能需要高版本的 Nodejs，我们可以通过其他方式来直接安装高版本的 Nodejs。

例如

- 1、直接下载 Nodejs 的二进制 tar 包解压到指定目录：（需要手动配置环境变量）
 - 2、使用 n 模块：（需要先从源里安装 npm，再使用 npm 安装 n 模块）
 - 3、使用 nvm 模块；
- 等。

下面以使用 nvm 安装 Nodejs v16.17.1 版本为例，介绍具体安装步骤：

a. 安装 nvm

首先，我们需要先安装 nvm 模块，可以通过官方提供的安装脚本进行安装，也可以通过从下面的链接下载我们已经打好的 deb 包进行安装。

nvm 模块下载链接：<https://wwpp.lanzoum.com/ib2ps15ipocb> 密码:99ik

```
$ sudo dpkg -i nvm_0.39.3_all.deb
[sudo] kylin 的密码:
正在选中未选择的软件包 nvm。
(正在读取数据库 ... 系统当前共安装有 228542 个文件和目录。)
正准备解包 nvm_0.39.3_all.deb ...
正在解包 nvm (0.39.3) ...
正在设置 nvm (0.39.3) ...
####配置环境变量####
kylin
####环境变量配置完毕####
```

安装完成后，需要重新打开一个终端，然后使用 `nvm -v` 查看是否安装成功。

```
$ nvm -v
0.39.3
```

如上表示 nvm 模块安装成功，下面就可以使用 nvm 来安装高版本 Nodejs。

b. 安装 Nodejs

执行 `nvm install 16.17.1` 进行 Nodejs 在线安装。

```
$ nvm install 16.17.1
Downloading and installing node v16.17.1...
```

```
Downloading https://npm.taobao.org/dist/v16.17.1/node-v16.17.1-linux-x64.tar.xz...
--2023-08-16 10:14:39-- https://npm.taobao.org/dist/v16.17.1/node-v16.17.1-linux-x64.tar.xz
正在解析主机 npm.taobao.org (npm.taobao.org)... 114.55.80.225
正在连接 npm.taobao.org (npm.taobao.org)|114.55.80.225|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 302 Moved Temporarily
位置: https://cdn.npmirror.com/binaries/node/v16.17.1/node-v16.17.1-linux-x64.tar.xz [跟随至新的 URL]
--2023-08-16 10:14:39-- https://cdn.npmirror.com/binaries/node/v16.17.1/node-v16.17.1-linux-x64.tar.xz
正在解析主机 cdn.npmirror.com (cdn.npmirror.com)... 111.2.79.91, 111.2.90.107, 36.158.203.214, ...
正在连接 cdn.npmirror.com (cdn.npmirror.com)|111.2.79.91|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度: 22423816 (21M) [application/x-xz]
正在保存至: "/home/kylin/.npm/.cache/bin/node-v16.17.1-linux-x64/node-v16.17.1-linux-x64.tar.xz"

/home/kylin/.npm/.cache/bi 100%[=====>] 21.38M 3.11MB/s
in 6.7s

2023-08-16 10:14:46 (3.20 MB/s) - 已保存
"/home/kylin/.npm/.cache/bin/node-v16.17.1-linux-x64/node-v16.17.1-linux-x64.tar.xz"
[22423816/22423816]

Computing checksum with sha256sum
Checksums matched!
Now using node v16.17.1 (npm v8.15.0)
Creating default alias: default -> 16.17.1 (-> v16.17.1)
```

c. 验证 Nodejs 安装的版本

```
$ node -v
v16.17.1
$ npm -v
8.15.0
```

2.1.2 从源中安装 git

```
$ sudo apt install git -y
```

2.1.3 设置 npm 仓库源(可选)

npm 原始的源(<https://registry.npmjs.org>)在国外服务器上，拉取依赖相对较慢，建议将 npm 的仓库源修改为国内淘宝源，配置方法如下：

```
$ npm config set registry https://registry.npm.taobao.org
```

2.1.4 设置 electron 镜像源(可选)

在项目依赖安装过程中，electron 模块会通过 electron-download 下载 Electron 的预编译二进制文件。如果没有切换 npm 仓库源的情况下，默认通过访问 GitHub 的发布下载页面来完成，拉取相对较慢，建议将 electron 镜像源修改为国内淘宝源，配置方法如下：


```
$ npm config set electron_mirror https://npm.taobao.org/mirrors/electron/
```

2.2 项目开发

此处以 electron-quick-start 项目使用 electron-v20.0.3 开发为例：

2.2.1 下载 electron-quick-start 项目

```
$ git clone https://github.com/electron/electron-quick-start
$ cd electron-quick-start
#删除 lock 文件
$ rm -rf package-lock.json
```

2.2.2 修改 electron 版本

编辑 package.json 文件，将 electron 版本修改为 20.0.3，如下所示：

```
$ cat package.json
{
  "name": "electron-quick-start",
  "version": "1.0.0",
  "description": "A minimal Electron application",
  "main": "main.js",
  "scripts": {
    "start": "electron ."
  },
  "repository": "https://github.com/electron/electron-quick-start",
  "keywords": [
    "Electron",
    "quick",
    "start",
    "tutorial",
    "demo"
  ],
  "author": "GitHub",
  "license": "CC0-1.0",
  "devDependencies": {
    "electron": "20.0.3"
  }
}
```

2.2.3 安装 electron 及依赖包

执行 npm install 命令来安装 electron 及依赖包，安装完成后会有如下提示，表示 electron 及依赖包安装成功。

```
$ npm install
```

added 92 packages in 8m

2.3 启动项目

执行 `npm start` 命令来启动项目。



如上图所示，即表示项目运行成功。

2.4 打包

项目调试完成后，需要打包成符合银河麒麟桌面操作系统 V10 软件商店上架规范的 deb 包，大部分项目是使用 `electron-builder` 工具来进行打包。当然，也可以使用 `electron-packager` 和 `electron-installer-debian` 工具进行打包，下面使用 `electron-builder-v21.2.0` 对项目进行打包。

2.4.1 安装 electron-builder

执行下面的命令来安装 `electron-builder`，如果后面不加版本号会默认安装当前仓库源中最新的版本，最新的版本对 `node` 和 `npm` 会有一定的要求，安装过程中会有一些告警，可以忽略。

```
$ npm install electron-builder@21.2.0 --save-dev

> ejs@2.7.4 postinstall /home/kylin/electron-quick-start/node_modules/ejs
> node ./postinstall.js

Thank you for installing EJS: built with the Jake JavaScript build tool (https://jakejs.com/)

npm WARN notsup Unsupported engine for fs-extra@10.0.1: wanted: {"node": ">=12"} (current: {"node": "10.19.0", "npm": "6.13.4"})
npm WARN notsup Not compatible with your version of node/npm: fs-extra@10.0.1

+ electron-builder@21.2.0
added 158 packages from 117 contributors in 14.689s

13 packages are looking for funding
  run `npm fund` for details
```

```
npm update check failed
Try running with sudo or get access
to the local update config store via
sudo chown -R $USER:$(id -gn $USER) /home/kylin/.config
```

```
kylin@kylin-v10-2101:~/electron-quick-start$
```

2.4.2 添加打包命令

编辑 package.json 文件，添加打包相关的参数，如下所示：

```
$ cat package.json
{
  "name": "electron-quick-start",
  "version": "1.0.0",
  "description": "A minimal Electron application",
  "main": "main.js",
  "scripts": {
    "start": "electron .",
    "builder": "electron-builder -l deb"
  },
  "build": {
    "productName": "electron-quick-start",
    "asar": "false",
    "appId": "electron-quick-start",
    "directories": {
      "output": "dist"
    },
    "linux": {
      "icon": "icons",
      "category": "Education",
      "target": "deb"
    }
  },
  "repository": "https://github.com/electron/electron-quick-start",
  "keywords": [
    "Electron",
    "quick",
    "start",
    "tutorial",
    "demo"
  ],
  "author": {
```

```
"name": "Wu Zhaohui",  
"email": "wuzhaohui@kylinos.cn"  
},  
"license": "CC0-1.0",  
"devDependencies": {  
  "electron": "20.0.3"  
}  
}
```

注:

- productName: 项目名,这也是生成的 deb 文件的包名
- appId: 包名
- directories: 目录
 - output: 生成 deb 包的存放目录
- linux: 构建 linux 的选项
 - category: 应用分类

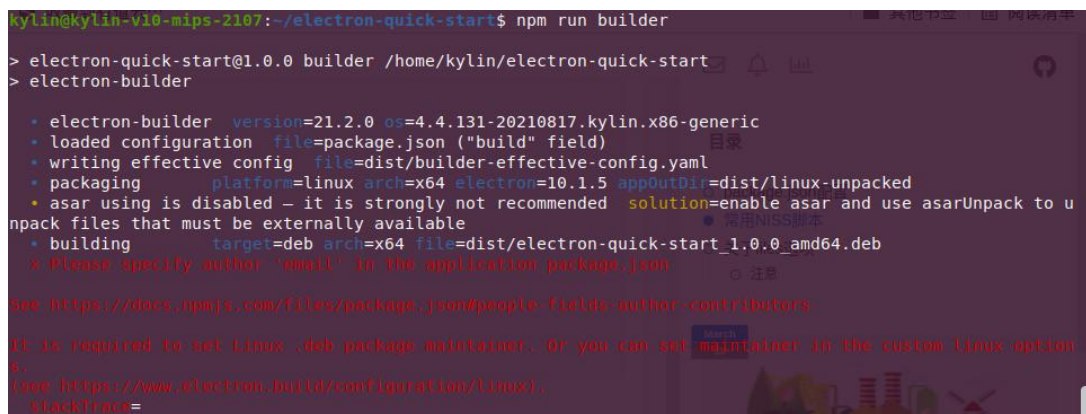
#Categories 分类要求:

安卓 Android;
网络 Network;
社交 Messaging;
影音 Audio、Video;
开发 Development;
图像 Graphics;
游戏 Game;
办公 Office、Calculator、Spreadsheet、Presentation、WordProcessor、TextEditor;
教育 Education;
系统 System、Settings、Security;

- target: 目标封装类型,这里要打成 deb 包,所以写成 deb
- icon: 自定义图标路径,如果不指定就用 electron 默认图标
- name: 姓名
- email: 邮箱

使用 electron-builder 打包时需要使用打包者信息,特别是 name 和 email 字段,不填写

打包时会有报错信息: “× Please specify author 'email' in the application package.json”



```
kylin@kylin-v10-mips-2107:~/electron-quick-start$ npm run builder  
> electron-quick-start@1.0.0 builder /home/kylin/electron-quick-start  
> electron-builder  
  
  * electron-builder version=21.2.0 os=4.4.131-20210817.kylin.x86-generic  
  * loaded configuration file=package.json ("build" field)  
  * writing effective config file=dist/builder-effective-config.yaml  
  * packaging platform=linux arch=x64 electron=10.1.5 appOutDir=dist/linux-unpacked  
  * asar using is disabled - it is strongly not recommended solution=enable asar and use asarUnpack to u  
  * unpack files that must be externally available  
  * building target=deb arch=x64 file=dist/electron-quick-start_1.0.0_amd64.deb  
  * Please specify author 'email' in the application package.json  
  
See https://docs.npmjs.com/files/package.json#people-fields-author-contributors  
  
It is required to set linux .deb package maintainer. Or you can set maintainer in the custom linux option  
s.  
(see https://www.electron.build/configuration/linux).  
stackTrace=
```

2.4.3 添加应用图标

由于 electron-quick-start 项目中没有自带图标，这里我们需要自己添加一下图标目录及相应尺寸的图标。

```
$ mkdir -p icons
```

添加图标文件

```
$ ls icons
```

```
128x128.png 16x16.png 256x256.png 32x32.png 512x512.png 64x64.png
```

另外，某些应用打包后，在系统安装启动后，任务栏图标显示异常，此时可以通过修改 main.js 中的代码来规避此问题。具体操作方法如下：

```
$ vim main.js
```

找到 main.js 中的 createWindow 函数，添加 icon

```
function createWindow () {  
  // Create the browser window.  
  const mainWindow = new BrowserWindow({  
    width: 800,  
    height: 600,  
    webPreferences: {  
      preload: path.join(__dirname, 'preload.js')  
    }  
  })  
}
```

添加 icon，修改后如下：

```
function createWindow () {  
  // Create the browser window.  
  const mainWindow = new BrowserWindow({  
    width: 800,  
    height: 600,  
    icon: path.join(__dirname, 'icons/512x512.png'),  
    webPreferences: {  
      preload: path.join(__dirname, 'preload.js')  
    }  
  })  
}
```

注：注意此处图标的写法，如果图标位置不对有可能会造成打包后报：“段错误，核心已转储”，图标需要使用 512x512 以上的 png 格式；

2.4.4 打 deb 包

执行 npm run builder 命令进行打包，打包完成后会有如下提示，表示打包成功。

```
$ npm run builder  
> electron-quick-start@1.0.0 builder /home/kylin/electron-quick-start  
> electron-builder
```

```
• electron-builder version=21.2.0 os=4.4.131-20210817.kylin.X86_64-generic
• loaded configuration file=package.json ("build" field)
• writing effective config file=dist/builder-effective-config.yaml
• packaging platform=linux arch=x64 electron=20.0.3 appOutDir=dist/linux-unpacked
• asar using is disabled — it is strongly not recommended solution=enable asar and use asarUnpack to
unpack files that must be externally available
• building target=deb arch=x64 file=dist/electron-quick-start_1.0.0_amd64.deb
• downloading
url=https://github.com/electron-userland/electron-builder-binaries/releases/download/fpm-1.9.3-2.3.1-linux-X86_
64_64/fpm-1.9.3-2.3.1-linux-X86_64_64.7z size=5.0 MB parts=1
• downloaded
url=https://github.com/electron-userland/electron-builder-binaries/releases/download/fpm-1.9.3-2.3.1-linux-X86_
64_64/fpm-1.9.3-2.3.1-linux-X86_64_64.7z duration=29.631s
```

打包后可以看到 dist 目录下生成的 deb 包

```
$ ls dist/
builder-effective-config.yaml electron-quick-start_1.0.0_amd64.deb linux-unpacked
```

2.4.5 修改 deb 包

某些情况下，我们打的包可能会存在一些问题但又不需要重新编译，我们通过下面的命令来对 deb 包进行一定修改。

a. 解包

使用如下命令将打好的 deb 包解包

```
$ fakeroot dpkg-deb -R electron-quick-start_1.0.0_amd64.deb electron-quick-start_1.0.0_amd64
```

按照打包规范对 deb 包进行调试

b. 重新打包

然后，使用如下命令重新打包

```
$ fakeroot dpkg-deb -b electron-quick-start_1.0.0_amd64 .
```

注：

不写打包名称会按照 control 文件自动进行命名打包，会将原包覆盖，可以使用如下命令通过自定义新包名称来进行打包

```
$ fakeroot dpkg-deb -b electron-quick-start_1.0.0_amd64
electron-quick-start_1.0.0_amd64_new.deb
```

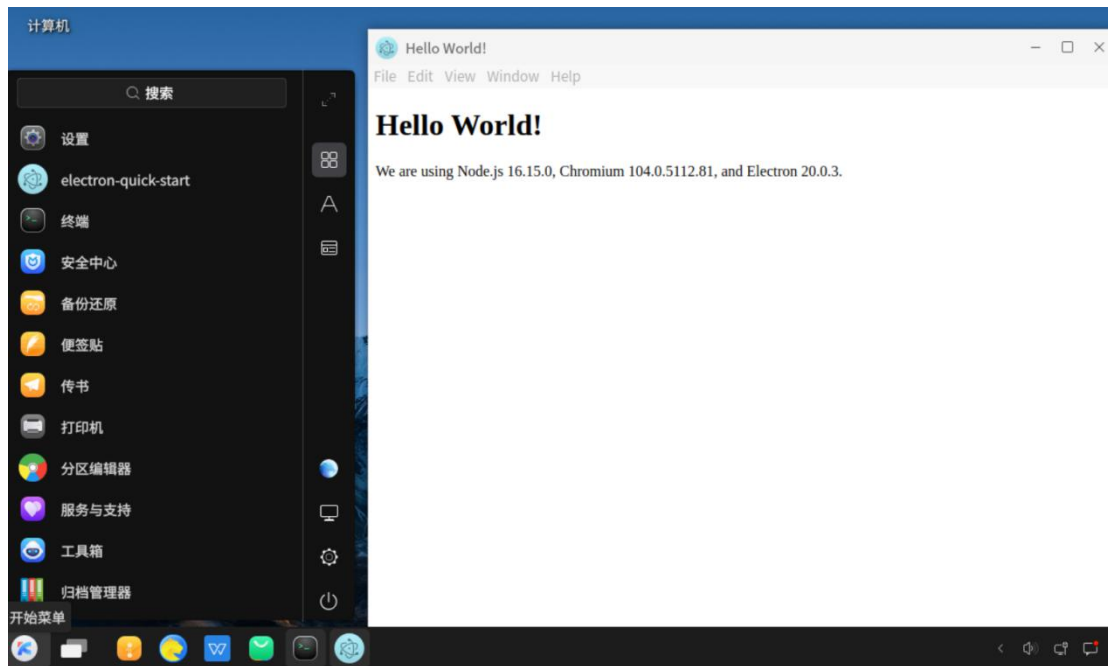
2.5 验包

2.5.1 安装

在 X86_64 架构机器上，双击或在终端执行 `sudo dpkg -i *.deb` 来安装 deb 包。

2.5.2 启动

从开始菜单，找到 electron-quick-start 点击启动，如下图，即表示打包成功



3 ARM64 架构

3.1 安装开发基础环境

3.1.1 安装 Nodejs

[参考 2.1.1 章节。](#)

3.1.2 从源中安装 git

[参考 2.1.2 章节。](#)

3.1.3 设置 npm 仓库源(可选)

[参考 2.1.3 章节。](#)

3.1.4 设置 electron 镜像源（可选）

[参考 2.1.4 章节。](#)

3.2 项目开发

此处以 electron-quick-start 项目使用 electron-v20.0.3 开发为例：

3.2.1 下载 electron-quick-start 项目

[参考 2.2.1 章节。](#)

3.2.2 修改 electron 版本

[参考 2.2.2 章节。](#)

3.2.3 安装 electron 及依赖包

[参考 2.2.3 章节。](#)

3.3 启动项目

执行 npm start 命令来启动项目。



如上图所示，即表示项目运行成功。

3.4 打包

3.4.1 安装 electron-builder

[参考 2.4.1 章节。](#)

3.4.2 添加打包命令

[参考 2.4.2 章节。](#)

3.4.3 添加应用图标

[参考 2.4.3 章节。](#)

3.4.4 安装 fpm

使用 electron-builder 打包时需要用到 fpm 包，但 fpm 包 npm 仓库源中仅有 X86_64 架构的包，没有其他架构的包，打包时会有报错：

```
$ npm run builder

> electron-quick-start@1.0.0 builder /home/kylin/electron-quick-start
> electron-builder

  • electron-builder version=21.2.0 os=4.4.131-20210120.kylin.desktop-generic
  • loaded configuration file=package.json ("build" field)
  • writing effective config file=dist/builder-effective-config.yaml
  • packaging platform=linux arch=arm64 electron=20.0.3 appOutDir=dist/linux-arm64-unpacked
  • asar using is disabled — it is strongly not recommended solution=enable asar and use asarUnpack to
unpack files that must be externally available
  • building target=deb arch=arm64 file=dist/electron-quick-start_1.0.0_arm64.deb
  • default Electron icon is used reason=application icon is not set
  • downloading
url=https://github.com/electron-userland/electron-builder-binaries/releases/download/fpm-1.9.3-2.3.1-linux-X86_
64/fpm-1.9.3-2.3.1-linux-X86_64.7z size=4.6 MB parts=1
  • downloaded
url=https://github.com/electron-userland/electron-builder-binaries/releases/download/fpm-1.9.3-2.3.1-linux-X86_
64/fpm-1.9.3-2.3.1-linux-X86_64.7z duration=2m20.499s
  ✖ cannot execute cause=exit status 1

errorOut=/home/kylin/.cache/electron-builder/fpm/fpm-1.9.3-2.3.1-linux-X86_64/lib/ruby/bin/ruby: 行 6:
/home/kylin/.cache/electron-builder/fpm/fpm-1.9.3-2.3.1-linux-X86_64/lib/ruby/bin.real/ruby: cannot execute
binary file: 可执行文件格式错误
/home/kylin/.cache/electron-builder/fpm/fpm-1.9.3-2.3.1-linux-X86_64/lib/ruby/bin/ruby: 行 6:
/home/kylin/.cache/electron-builder/fpm/fpm-1.9.3-2.3.1-linux-X86_64/lib/ruby/bin.real/ruby: 成功

command=/home/kylin/.cache/electron-builder/fpm/fpm-1.9.3-2.3.1-linux-X86_64/fpm -s
```

```
dir --force -t deb -d libgtk-3-0 -d libnotify4 -d libnss3 -d libxss1 -d libxtst6 -d xdg-utils -d libatspi2.0-0 -d
libuid1 -d libappindicator3-1 -d libsecret-1-0 --deb-compression xz --architecture arm64 --name
electron-quick-start --after-install /tmp/t-BtXXvc/0-after-install --after-remove /tmp/t-BtXXvc/1-after-remove
--description '
    A minimal Electron application' --version 1.0.0 --package
/home/kylin/electron-quick-start/dist/electron-quick-start_1.0.0_arm64.deb --maintainer 'Wu Zhaohui
<wuzhaohui@kylinos.cn>' --url 'https://github.com/electron/electron-quick-start#readme' --vendor 'Wu Zhaohui
<wuzhaohui@kylinos.cn>' --license CC0-1.0
/home/kylin/electron-quick-start/dist/linux-arm64-unpacked=/opt/electron-quick-start
/home/kylin/electron-quick-start/node_modules/app-builder-lib/templates/icons/electron-linux/16x16.png=/usr/sh
are/icons/hicolor/16x16/apps/electron-quick-start.png
/home/kylin/electron-quick-start/node_modules/app-builder-lib/templates/icons/electron-linux/32x32.png=/usr/sh
are/icons/hicolor/32x32/apps/electron-quick-start.png
/home/kylin/electron-quick-start/node_modules/app-builder-lib/templates/icons/electron-linux/48x48.png=/usr/sh
are/icons/hicolor/48x48/apps/electron-quick-start.png
/home/kylin/electron-quick-start/node_modules/app-builder-lib/templates/icons/electron-linux/64x64.png=/usr/sh
are/icons/hicolor/64x64/apps/electron-quick-start.png
/home/kylin/electron-quick-start/node_modules/app-builder-lib/templates/icons/electron-linux/128x128.png=/usr/
share/icons/hicolor/128x128/apps/electron-quick-start.png
/home/kylin/electron-quick-start/node_modules/app-builder-lib/templates/icons/electron-linux/256x256.png=/usr/
share/icons/hicolor/256x256/apps/electron-quick-start.png
/tmp/t-BtXXvc/2-electron-quick-start.desktop=/usr/share/applications/electron-quick-start.desktop
workingDir=

npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! electron-quick-start@1.0.0 builder: `electron-builder`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the electron-quick-start@1.0.0 builder script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR! /home/kylin/.npm/_logs/2022-03-21T08_35_21_121Z-debug.log
kylin@kylin-v10-2101:~/electron-quick-start$
```

我们可以从系统源中安装 ruby，然后使用 `gem install fpm` 来安装 fpm，然后设置全局变量，使用系统 fpm。具体操作方法如下：

```
$ sudo apt update
$ sudo apt install ruby -y
$ sudo gem install fpm
$ export USE_SYSTEM_FPM="true"
```

注：

V10 系统使用 gem 安装 fpm 时会有如下报错，这是由于 V10 系统中 ruby 版本低导致的，

需要升级系统 ruby 版本。

```
$ sudo gem install fpm
Fetching: rexml-3.2.6.gem (100%)
ERROR:  Error installing fpm:
  rexml requires Ruby version >= 2.5.0.
```

ruby 下载链接:

<https://wwpp.lanzoum.com/b03es12ba> 密码:1usn

下载后解压, 进入目录, 使用 `sudo dpkg -i *.deb` 安装, 然后再进行上面的操作。

3.4.5 打 deb 包

执行 `npm run builder` 命令进行打包, 打包完成后会有如下提示, 表示打包成功。

```
$ npm run builder

> electron-quick-start@1.0.0 builder /home/kylin/electron-quick-start
> electron-builder

• electron-builder version=21.2.0 os=4.4.131-20210120.kylin.desktop-generic
• loaded configuration file=package.json ("build" field)
• writing effective config file=dist/builder-effective-config.yaml
• packaging platform=linux arch=arm64 electron=20.0.3 appOutDir=dist/linux-arm64-unpacked
• asar using is disabled — it is strongly not recommended solution=enable asar and use asarUnpack to
unpack files that must be externally available
• building target=deb arch=arm64 file=dist/electron-quick-start_1.0.0_arm64.deb
```

打包后可以看到 dist 目录下生成的 deb 包

```
$ ls dist/
builder-effective-config.yaml electron-quick-start_1.0.0_arm64.deb linux-unpacked
```

3.4.6 修改 deb 包

[参考 2.4.5 章节](#)。

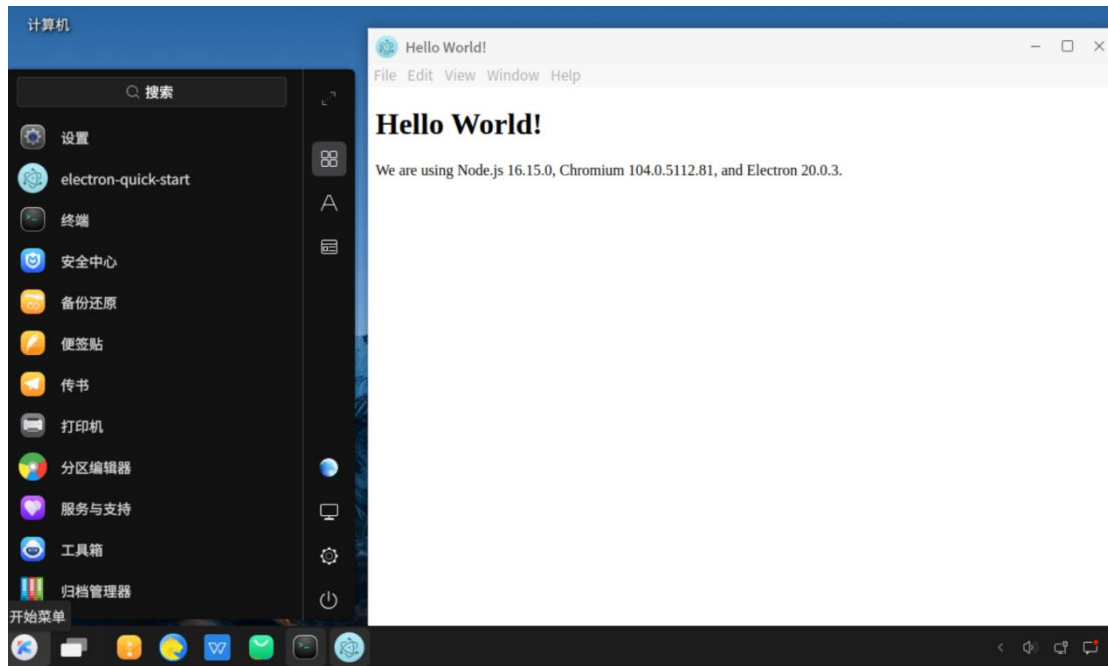
3.5 验包

3.5.1 安装

在 ARM64 架构机器上, 双击或在终端执行 `sudo dpkg -i ***.deb` 来安装 deb 包。

3.5.2 启动

从开始菜单, 找到 electron-quick-start 点击启动, 如下图, 即表示打包成功



4 Mips64el 架构

Mips64el 架构与其他两个架构（X86_64 和 ARM64）Electron 应用打包过程有些差异，因为 Nodejs 官方默认没有提供 Mips64el 架构的二进制 tar 包，所以无法在线使用 n 模块或 nvm 模块来直接安装 Nodejs。另外，npm 官方源中也并没有 Mips64el 架构的 electron 及相关包，需要使用龙芯移植后的 npm 仓库源。

4.1 安装开发基础环境

4.1.1 安装 Nodejs

Mips64el 架构 V10-2107 默认预装 node（8.9.4）和 npm（5.6.0），V10-2107 之前的版本默认没有预装，需要从源里安装，源内版本为 node（4.2.6）、npm（3.5.2）。高版本的 nodejs 需要使用其他方式进行安装，目前 V10 系统上 Mips64el 架构已经适配了 v12.19.1、v16.17.1、v18.13.0 等高版本的 Nodejs。

V10 系统已经适配的 Nodejs 下载链接：

<https://wwpp.lanzoum.com/b03es15tg> 密码:9u67

V10(SP1)默认没有预装 npm 和 node，需要从源里安装，源里的 node 版本为 v10.19.0，npm 版本为 v6.14.4。高版本的 nodejs 需要使用其他方式进行安装，目前 V10(SP1)系统上 Mips64el 架构已经适配了 v12.16.1、v12.16.3、v16.17.1、v18.13.0 等高版本的 Nodejs。

V10(SP1)已经适配的 Nodejs 下载链接：

<https://wwpp.lanzoum.com/b03es18sd> 密码:8cat

由于 Nodejs 官方没有提供 Mips64el 架构的二进制 tar 包，所以无法在线使用 nvm 安装 Nodejs(如果直接使用 nvm install 来进行安装，会直接下载源码编译安装，但由于编译参数问题无法编译成功)，故采用离线安装的方式来进行安装高版本的 Nodejs。

下面以使用 nvm 离线安装 Nodejs v16.17.1 版本为例，介绍具体安装步骤。

a. 安装 nvm

首先，我们需要先安装 nvm 模块，可以通过官方提供的安装脚本进行安装，也可以通过从下面的链接下载我们已经打好的 deb 包进行安装。

nvm 模块下载链接：<https://wwpp.lanzoum.com/ib2ps15ipocb> 密码:99ik

```
$ sudo dpkg -i nvm_0.39.3_all.deb
[sudo] kylin 的密码:
正在选中未选择的软件包 nvm。
(正在读取数据库 ... 系统当前共安装有 228542 个文件和目录。)
```

```
正准备解包 nvm_0.39.3_all.deb ...
正在解包 nvm (0.39.3) ...
正在设置 nvm (0.39.3) ...
####配置环境变量####
kylin
####环境变量配置完毕####
```

安装完成后，需要重新打开一个终端，然后使用 `nvm -v` 查看是否安装成功。

```
$ nvm -v
0.39.3
```

如上表示 `nvm` 模块安装成功，下面就可以使用 `nvm` 来安装高版本 Nodejs。

b. 安装 openssl(仅限于 V10)

在 V10 系统 Mips64el 架构上，由于编译高版本的 nodejs 时需要使用高版本的 openssl，所以 nodejs 运行环境也需要高版本的 openssl 环境，需要手动下载高版本的 openssl 离线包，解压后进行离线安装。

高版本 openssl 下载链接：

<https://wwpp.lanzoum.com/b03es1i5a> 密码:5ywl

```
$ tar xvf openssl_1.1.1n-0+deb10u5kylin1_mips64el_v10.tar.gz
openssl_1.1.1n-0+deb10u5kylin1_mips64el_v10/libssl1.1_1.1.1n-0+deb10u5kylin1_mips64el.deb
openssl_1.1.1n-0+deb10u5kylin1_mips64el_v10/libssl-dev_1.1.1n-0+deb10u5kylin1_mips64el.deb
openssl_1.1.1n-0+deb10u5kylin1_mips64el_v10/libssl-doc_1.1.1n-0+deb10u5kylin1_all.deb
openssl_1.1.1n-0+deb10u5kylin1_mips64el_v10/openssl_1.1.1n-0+deb10u5kylin1_mips64el.deb
$ sudo dpkg -i openssl_1.1.1n-0+deb10u5kylin1_mips64el_v10/*.deb
[sudo] kylin 的密码:
正在选中未选择的软件包 libssl1.1:mips64el。
(正在读取数据库 ... 系统当前共安装有 243962 个文件和目录。)
正准备解包 .../libssl1.1_1.1.1n-0+deb10u5kylin1_mips64el.deb ...
正在解包 libssl1.1:mips64el (1.1.1n-0+deb10u5kylin1) ...
正准备解包 .../libssl-dev_1.1.1n-0+deb10u5kylin1_mips64el.deb ...
正在将 libssl-dev:mips64el (1.1.1n-0+deb10u5kylin1) 解包到 (1.0.2g-1kord4.15) 上 ...
正在选中未选择的软件包 libssl-doc。
正准备解包 .../libssl-doc_1.1.1n-0+deb10u5kylin1_all.deb ...
正在解包 libssl-doc (1.1.1n-0+deb10u5kylin1) ...
正准备解包 .../openssl_1.1.1n-0+deb10u5kylin1_mips64el.deb ...
正在将 openssl (1.1.1n-0+deb10u5kylin1) 解包到 (1.0.2g-1kord4.15) 上 ...
正在设置 libssl1.1:mips64el (1.1.1n-0+deb10u5kylin1) ...
正在设置 libssl-dev:mips64el (1.1.1n-0+deb10u5kylin1) ...
正在设置 libssl-doc (1.1.1n-0+deb10u5kylin1) ...
正在设置 openssl (1.1.1n-0+deb10u5kylin1) ...
正在安装新版本配置文件 /etc/ssl/openssl.cnf ...
正在处理用于 libc-bin (2.23-0kord11k20.8) 的触发器 ...
正在处理用于 man-db (2.7.5-1kord) 的触发器 ...
```

c. 安装 Nodejs

离线安装需要先下载 Nodejs 的二进制 tar 包，然后手动创建版本安装目录，然后将 tar 包解压后放到创建的目录，再使用 `nvm use` 命令来应用生效。

下载并解压 `node-v16.17.1-linux-mips64el.tar.gz`，然后进行离线安装。

```
$ tar xvf node-v16.17.1-linux-mips64el.tar.gz
```

手动创建版本安装目录

```
$ mkdir -p ~/.nvm/versions/node/v16.17.1
```

将上面解压的内容拷贝到手动创建的目录

```
$ cp -rp node-v16.17.1-linux-mips64el/* ~/.nvm/versions/node/v16.17.1/
```

使用 `nvm use` 命令进行应用生效

```
$ nvm use 16.17.1
```

```
Now using node v16.17.1 (npm v8.15.0)
```

d. 验证 Nodejs 安装的版本

```
$ node -v
```

```
v16.17.1
```

```
$ npm -v
```

```
8.15.0
```

4.1.2 从源中安装 git

[参考 2.1.2 章节](#)。

4.1.3 设置 npm 仓库源(必选)

龙芯架构需要使用龙芯的 npm 仓库源，配置方法如下：

```
$ npm config set registry https://registry.loongnix.cn:4873
```

注：

龙芯 npm 仓库源配置使用：<http://docs.loongnix.cn/nodejs/doc/list/03.龙芯 npm 的安装和仓库配置使用.html>

4.1.4 设置 electron 镜像源(必选)

龙芯开源生态社区资源目前主要以 Loongarch64 架构为主，Mips64el 架构基本已经不再网站维护。目前在龙芯源中可找到 Mips64el 架构的 Electron 版本有 v4.1.3、v6.1.7、v10.1.0，其他 Mips64el 架构高版本的 Electron 相关需求只能通过线下提需求获取。对于线下获取的 electron 包，可以通过本地搭建 electron 镜像源来进行安装。

Electron 镜像源搭建方法请参考[附录 6.1 章节](#)。

对于需要 Electron v4.1.3、v6.1.7、v10.1.0 版本的，可以直接配置龙芯存放 electron 的镜像源，配置方法如下：

```
$ npm config set electron_mirror http://ftp.loongnix.cn/os/loongnix/1.0/electron/releases/mips/
```

4.2 项目开发

此处以 electron-quick-start 项目使用 electron-v20.0.3 开发为例：

4.2.1 下载 electron-quick-start 项目

[参考 2.2.1 章节](#)。

4.2.2 修改 electron 版本

编辑 package.json 文件，将 electron 版本修改为 20.0.3。

```
$ cat package.json
{
  "name": "electron-quick-start",
  "version": "1.0.0",
  "description": "A minimal Electron application",
  "main": "main.js",
  "scripts": {
    "start": "electron ."
  },
  "repository": "https://github.com/electron/electron-quick-start",
  "keywords": [
    "Electron",
    "quick",
    "start",
    "tutorial",
    "demo"
  ],
  "author": "GitHub",
  "license": "CC0-1.0",
  "devDependencies": {
    "electron": "20.0.3"
  }
}
```

4.2.3 安装 electron 及依赖包

执行 `electron_use_remote_checksums=1 npm install` 命令来安装 electron 及依赖包，安装完成后会有如下提示，表示 electron 及依赖包安装成功。

```
$ electron_use_remote_checksums=1 npm install

added 91 packages in 3m

11 packages are looking for funding
  run `npm fund` for details
```


4.3 启动项目

执行 `npm start` 命令来启动项目。



如上图所示，即表示项目运行成功。

注：

V10 系统上面执行 `npm start` 的时候可能会有报错，报错如下：



```
npm ERR! errno 1
npm ERR! electron-quick-start@1.0.0 start: `electron`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the electron-quick-start@1.0.0 start script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR!     /home/kylin/.npm/_logs/2023-08-17T07_56_21_674Z-debug.log
```

根据报错提示可以看出是 chrome-sandbox 权限问题，需要修改权限

```
$ sudo chown root:root /home/kylin/electron-quick-start/node_modules/electron/dist/chrome-sandbox
$ sudo chmod 4755 /home/kylin/electron-quick-start/node_modules/electron/dist/chrome-sandbox
```

4.4 打包

项目调试完成后，需要打包成符合银河麒麟桌面操作系统 V10 软件商店上架规范的 deb 包，X86_64 和 ARM64 架构下一般是使用 electron-builder 工具来进行打包，但 npm 官方源中的 electron-builder 不支持 Mips64el 架构打包，需要使用龙芯 npm 仓库中已经适配的 electron-builder 版本。electron-builder 本身是不区分架构的，实际区分架构的是 electron-builder 的依赖包 builder-util，所以龙芯 npm 仓库中只有适配 builder-util 的记录。选择安装 electron-builder 版本时直接参考 builder-util 的版本号即可。

龙芯 npm 仓库适配列表见：[http://docs.loongnix.cn/nodejs/doc/list/04.npm 仓库适配列表.html](http://docs.loongnix.cn/nodejs/doc/list/04.npm%20仓库适配列表.html)。

此处以使用 electron-builder-v22.14.7 版本打包为例。

4.4.1 安装 electron-builder

执行下面的命令来安装 electron-builder，安装成功后，会有如下提示信息。

```
$ npm install electron-builder@22.14.7 --save-dev

added 189 packages in 2m

31 packages are looking for funding
  run `npm fund` for details
```

4.4.2 添加打包命令

[参考 2.4.2 章节。](#)

4.4.3 添加应用图标

[参考 2.4.3 章节。](#)

4.4.4 安装 fpm

[参考 3.4.4 章节](#)。

4.4.5 打 deb 包

使用 electron-builder 打包时会从 electron_mirror 镜像源中拉取 electron 包，Mips64el 架构拉取时的链接有一些问题，我们可以将本地已经存在的 electron 包拷贝到 electron 缓存目录，从而在打包时不需要再次拉取 electron 包，具体操作如下：

```
$ cp ~/.cache/electron/*/electron-v20.0.3-linux-mips64el.zip ~/.cache/electron/
```

注：上面命令中的“*”是一个很长的字符串，需要根据实际值进行填写

执行 `npm run builder` 命令进行打包，打包时间会有些长，等待完成即可。

打包完成后会有如下提示，表示打包成功。

```
$ npm run builder

> electron-quick-start@1.0.0 builder
> electron-builder

• electron-builder version=22.14.7 os=5.4.18-53-generic
• loaded configuration file=package.json ("build" field)
• writing effective config file=dist/builder-effective-config.yaml
• packaging platform=linux arch=mips64el electron=20.0.3 appOutDir=dist/linux-mips64el-unpacked
• asar usage is disabled — this is strongly not recommended solution=enable asar and use asarUnpack to
unpack files that must be externally available
• asar usage is disabled — this is strongly not recommended solution=enable asar and use asarUnpack to
unpack files that must be externally available
• building target=deb arch=mips64el file=dist/electron-quick-start_1.0.0_mips64el.deb
```

打包后可以看到 dist 目录下生成的 deb 包

```
$ ls dist/
builder-debug.yml          electron-quick-start_1.0.0_mips64el.deb
builder-effective-config.yaml  linux-mips64el-unpacked
```

注：

V10 系统打包过程中可能会出现卡死无法完成打包的情况，使用 `Ctrl+c` 取消会有如下报错：

```
• cancelled by SIGINT
/home/kylin/electron-quick-start/node_modules/app-builder-bin/linux/mips64el/app-builder exited with code
ERR_ELECTRON_BUILDER_CANNOT_EXECUTE failedTask=build stackTrace=Error:
/home/kylin/electron-quick-start/node_modules/app-builder-bin/linux/mips64el/app-builder exited with code
ERR_ELECTRON_BUILDER_CANNOT_EXECUTE
at ChildProcess.<anonymous>
(/home/kylin/electron-quick-start/node_modules/builder-util/src/util.ts:250:14)
```

```
at Object.onceWrapper (node:events:628:26)
at ChildProcess.emit (node:events:513:28)
at maybeClose (node:internal/child_process:1093:16)
at Socket.<anonymous> (node:internal/child_process:451:11)
at Socket.emit (node:events:513:28)
at Pipe.<anonymous> (node:net:757:14)
```

出现如上问题，需要替换一下 app-builder 文件
文件下载地址链接：

<https://wwpp.lanzoum.com/iXQOA16t1xfg> 密码:c1xa

下载解压后，将~/electron-quick-start/node_modules/app-builder-bin/linux/mips64el/目录下的 app-builder 文件备份，然后将解压目录下的 app-builder 文件拷贝到此目录下，再次进行打包即可。

另外，V10 系统上如果打好的包如果无法运行，并且从终端启动报错如下：

```
$ ./electron-quick-start
[15793:0407/171417.990981:FATAL:setuid_sandbox_host.cc(158)] The SUID sandbox helper binary was found,
but is not configured correctly. Rather than run without sandboxing I'm aborting now. You need to make sure that
/home/kylin/electron-quick-start/dist/linux-mips64el-unpacked/chrome-sandbox is owned by root and has mode
4755.
追踪与中断点陷阱 (核心已转储)
```

根据报错提示可以看出是 chrome-sandbox 权限问题，需要参考 4.5.5 解包后进行权限修改，然后重新打包

```
$ sudo chown root:root ~/electron-quick-start_1.0.0_mips64el/opt/electron-quick-start/chrome-sandbox
$ sudo chmod 4755 ~/electron-quick-start_1.0.0_mips64el/opt/electron-quick-start/chrome-sandbox
```

4.4.6 修改 deb 包

[参考 2.4.5 章节](#)。

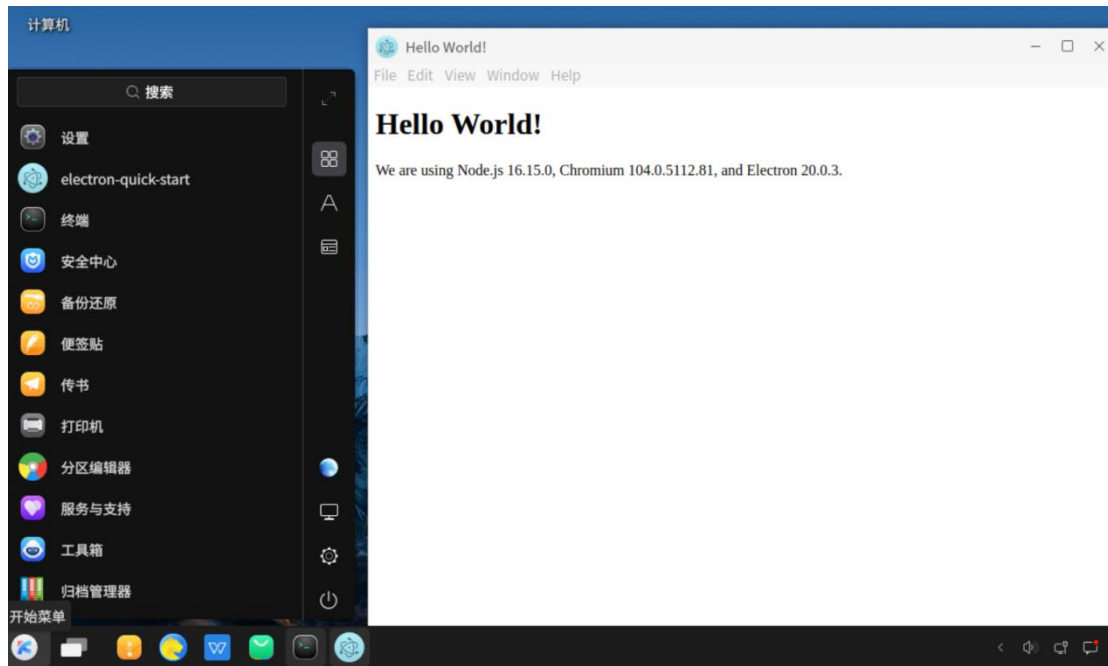
4.5 验包

4.5.1 安装

在 Mips64el 架构机器上，双击或在终端执行 `sudo dpkg -i *.deb` 来安装 deb 包。

4.5.2 启动

从开始菜单，找到 electron-quick-start 点击启动，如下图，即表示打包成功



5 LoongArch64 架构

LoongArch64 架构和 Mips64el 架构一样，Nodejs 官方没有提供 Loongarch64 架构的二进制 tar 包，npm 官方源中也没有对应架构的 electron 及相关包，需要从龙芯开源社区获取，目前龙芯已经适配了多个 Nodejs 版本和 electron 版本。

龙芯开源社区提供了 LoongArch64 架构上适配的 Electron 包，包括 v8.5.5、v12.0.9、v13.6.9、v15.0.0、v15.5.2、v16.2.3、v20.0.3、v17.4.7、v20.0.3，其他版本需要提需求适配。

5.1 安装开发基础环境

5.1.1 安装 Nodejs

龙芯开源社区提供了 Loongarch64 架构上适配的 Nodejs 包，包括 v10.24.1、v12.19.1、v14.16.1、v16.3.0、v16.5.0、v16.17.1、v18.13.0，其他版本需要提需求适配。

可以[参考 2.1.1 章节](#)，使用 nvm 进行安装，安装时可以使用 `nvm ls-remote node` 查看龙芯已经适配的 Nodejs 版本然后再进行选择安装。

```
$ nvm ls-remote node
v10.22.1 (LTS: Dubnium)
v10.24.1 (Latest LTS: Dubnium)
v12.19.1 (Latest LTS: Erbium)
v14.16.1 (Latest LTS: Fermium)
v16.3.0
v16.5.0
-> v16.17.1 (Latest LTS: Gallium)
v18.13.0 (Latest LTS: Hydrogen)
```

5.1.2 从源中安装 git

[参考 2.1.2 章节](#)。

5.1.3 设置 npm 仓库源(必选)

[参考 4.1.3 章节](#)。

5.1.4 设置 electron 镜像源(必选)

配置 npm 使用龙芯提供的 electron 镜像源，配置方法如下：

```
$ npm config set electron_mirror http://ftp.loongnix.cn/electron/LoongArch/
```

5.2 项目开发

此处以 electron-quick-start 项目使用 electron-v20.0.3 开发为例：

5.2.1 下载 electron-quick-start 项目

[参考 2.2.1 章节](#)。

5.2.2 修改 electron 版本

编辑 package.json 文件，将 electron 版本修改为 20.0.3。

```
$ cat package.json
{
  "name": "electron-quick-start",
  "version": "1.0.0",
  "description": "A minimal Electron application",
  "main": "main.js",
  "scripts": {
    "start": "electron ."
  },
  "repository": "https://github.com/electron/electron-quick-start",
  "keywords": [
    "Electron",
    "quick",
    "start",
    "tutorial",
    "demo"
  ],
  "author": "GitHub",
  "license": "CC0-1.0",
  "devDependencies": {
    "electron": "20.0.3"
  }
}
```

5.2.3 安装 electron 及依赖包

执行 `electron_use_remote_checksums=1 npm install` 命令来安装 electron 及依赖包，安装完成后会有如下提示，表示 electron 及依赖包安装成功。

```
$ electron_use_remote_checksums=1 npm install

added 13 packages, removed 191 packages, and changed 2 packages in 2m

11 packages are looking for funding
  run `npm fund` for details
```

5.3 启动项目

执行 `npm start` 命令来启动项目。



如上图所示，即表示项目运行成功。

5.4 打包

项目调试完成后，需要打包成符合银河麒麟桌面操作系统 V10 软件商店上架规范的 deb 包，此处以使用 `electron-builder-v22.14.7` 版本打包为例。

5.4.1 安装 electron-builder

执行下面的命令来安装 `electron-builder`，安装成功后，会有如下提示信息。

```
$ npm install electron-builder@22.14.7 --save-dev  
  
added 190 packages in 2m  
  
31 packages are looking for funding  
  run `npm fund` for detail
```

5.4.2 添加打包命令

[参考 2.4.2 章节。](#)

5.4.3 添加应用图标

[参考 2.4.3 章节。](#)

5.4.4 安装 fpm

[参考 3.4.4 章节。](#)

5.4.5 打 deb 包

使用 `electron-builder` 打包时会从 `electron_mirror` 镜像源中拉取 `electron` 包，Loongarch64 架构拉取时的链接有一些问题，我们可以将本地已经存在的 `electron` 包拷贝到 `electron` 缓存目录，从而在打包时不需要再次拉取 `electron` 包，具体操作如下：

```
$ cp ~/.cache/electron/*/electron-v20.0.3-linux-loong64.zip ~/.cache/electron/
```


注：上面命令中的“*”是一个很长的字符串，需要根据实际值进行填写。

执行 `npm run builder` 命令进行打包，打包时间会有些长，等待完成即可。

打包完成后会有如下提示，表示打包成功。

```
$ npm run builder

> electron-quick-start@1.0.0 builder
> electron-builder

  • electron-builder  version=22.14.7 os=5.4.18-55-generic
  • loaded configuration  file=package.json ("build" field)
  • writing effective config  file=dist/builder-effective-config.yaml
  • packaging            platform=linux arch=loong64 electron=20.0.3 appOutDir=dist/linux-loong64-unpacked
  • asar usage is disabled — this is strongly not recommended  solution=enable asar and use asarUnpack to
unpack files that must be externally available
  • asar usage is disabled — this is strongly not recommended  solution=enable asar and use asarUnpack to
unpack files that must be externally available
  • building            target=deb arch=loong64 file=dist/electron-quick-start_1.0.0_loong64.deb
```

打包后可以看到 `dist` 目录下生成的 `deb` 包。

```
$ ls dist/
builder-debug.yml          electron-quick-start_1.0.0_loong64.deb  linux-loong64-unpacked
builder-effective-config.yaml
```

5.4.6 修改 deb 包

[参考 2.4.5 章节](#)。

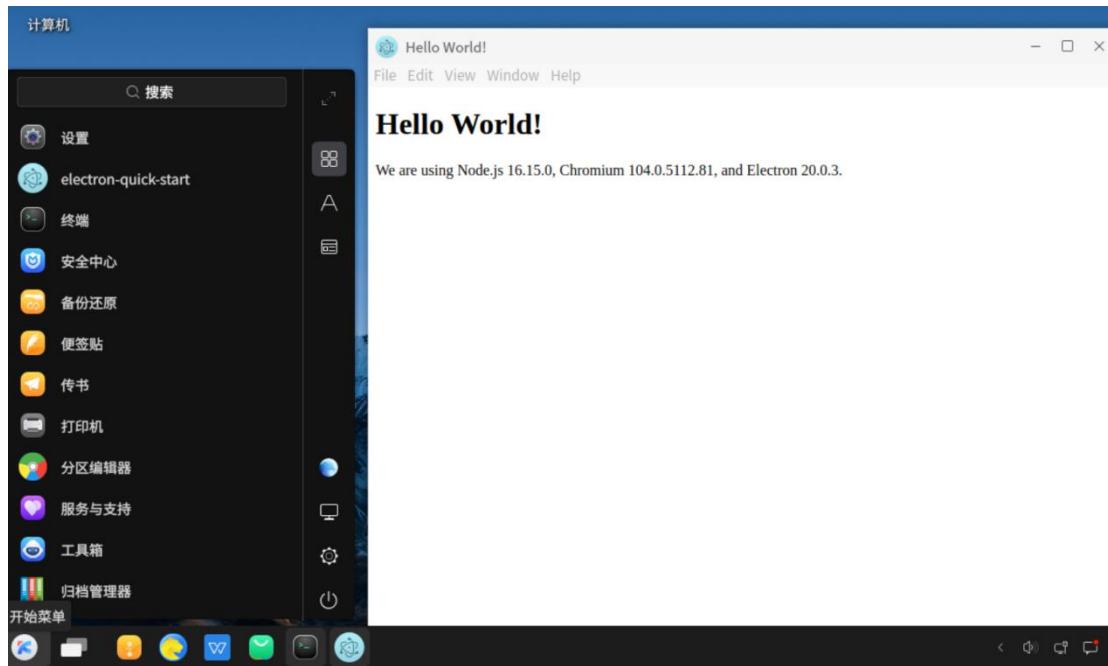
5.5 验包

5.5.1 安装

在 loongarch64 架构机器上，双击或在终端执行 `sudo dpkg -i *.deb` 来安装 `deb` 包。

5.5.2 启动

从开始菜单，找到 `electron-quick-start` 点击启动，如下图，即表示打包成功



如上图所示，即表示打包成功。

6 附录

6.1 搭建 electron 本地镜像源

下面介绍一种在本机快速搭建一个 electron 镜像源的方法，供打包使用。

6.1.1 创建 electron 镜像源目录

新开启一个终端，执行如下命令。

```
$ cd ~  
$ mkdir -p http/electron  
$ cd http/electron
```

6.1.2 将 electron 包放入上述目录

下载 electron_v20.0.3-bin.tar.gz，然后进行解压

```
$ tar xvf electron_v20.0.3-bin.tar.gz  
$ tree -L 3  
.  
├── electron_v20.0.3-bin.tar.gz  
└── v20.0.3  
    ├── chromedriver-v20.0.3-linux-loong64.zip  
    ├── chromedriver-v20.0.3-linux-mips64el.zip  
    ├── electron-api.json  
    ├── electron.d.ts  
    ├── electron-v20.0.3-linux-loong64.zip  
    ├── electron-v20.0.3-linux-mips64el.zip  
    ├── ffmpeg-v20.0.3-linux-loong64.zip  
    ├── ffmpeg-v20.0.3-linux-mips64el.zip  
    ├── hunspell_dictionaries.zip  
    ├── mksnapshot-v20.0.3-linux-loong64.zip  
    ├── mksnapshot-v20.0.3-linux-mips64el.zip  
    └── SHASUMS256.txt  
  
1 directory, 13 files
```

#创建不带 v 的版本目录，解决打包时 Response 404(Not Found)问题。

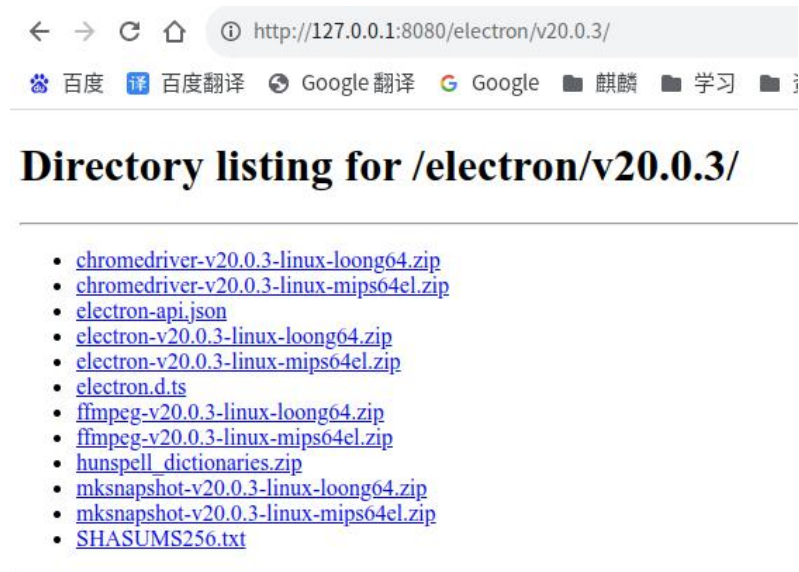
```
$ cp -r v20.0.3 20.0.3
```

6.1.3 启动 electron 镜像源

在 http 目录，使用如下命令启动 electron 镜像源。

```
$ cd ..  
$ python3 -m http.server 8080  
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

然后在浏览器输入 `http://127.0.0.1:8080/` 验证 electron 镜像源是否搭建成功。浏览器显示如下，表示 electron 镜像源搭建成功：



6.1.4 设置 electron 镜像源(必选)

配置 npm 使用本地搭建的 electron 镜像源，配置方法如下：

```
$ npm config set electron_mirror http://127.0.0.1:8080/electron/
```