基于 dotnet 系的自发布自包含的能力，可以将 dotnet 系的应用进行发布为独立应用，无需框架依赖，如此即可执行 deb 包的二进制打包过程，从而很方便对接上 UOS 统信系统软件安装包的要求

整体步骤可以分为三个大步骤：

步骤一：发布应用

步骤二：组织文件结构

步骤三：打包 deb 文件

本文接下来将按照步骤顺序一步步告诉大家如何在 Windows 上为 dotnet 系的应用创建符合要求的 UOS 统信系统软件安装包。本文写于 2023.12.25 如果你阅读本文的时间距离编写时间过远，可能本文有些知识不正确

## 步骤一 发布应用

无论是 CPF 应用还是 UNO 应用或者 Avalonia 应用，在进行构建符合要求的 UOS 软件安装包的第一步都是对应用项目进行发布，且为了后续打包和分发的方便，十分推荐发布为独立应用。这些应用的大概的发布命令行如下，请根据你的实际需求进行更改

dotnet publish -c release -r linux-x64 --self-contained true

完成发布之后，咱即可拿发布出来的文件夹进行制作符合要求的 UOS 软件安装包。为了减少大家的学习成本，本文将一步步进行，本文在编写过程中，将会先略去在本文所没有用到的知识点。但请大家放心的是，按照本文的方法是可以构建出一个符合要求的 UOS 软件安装包，只有一些扩展可选部分的功能被本文略过

在进入步骤二之前，我推荐将发布的文件夹拷贝到 UOS 系统上测试一下，确保应用本身发布出来是能够正常运行

## 步骤二 组织文件结构

本文属于尽可能多的手动化打包过程，所使用的工具只有文本编辑器以及 WSL （适用于 Linux 的 Windows 子系统） 和 dpkg 工具。本文这里使用 WSL 仅仅只是为了使用 dpkg 工具而已，尽量使用官方推荐的打包工具。事实上，咱是完全可以将此过程在 Windows 上完成的。但为了减少引入额外的工具，本文依然使用 dpkg 工具进行打包，在本文后面，大家可以看到 dpkg 工具的戏份也只有一句话

**2.1 准备打包环境**

在开始之前，需要先准备好打包环境

* 文本编辑器
  + 这里的文本编辑器可不推荐选用 Windows 自带的记事本哦。因为 Windows 自带的记事本绝大部分情况下的默认行为都会写入文件的 BOM 头，导致脆弱的 Linux 不认识而挂掉。如果一定要用 Windows 自带的记事本工具，记得保存选项里面选用的是 UTF-8 版本，不能选用 UTF-8 with BOM 等编码
  + 推荐使用 SublimeText 或 Vim 工具都可以
* Debian 的 WSL 工具
  + 下载地址：
  + <https://www.microsoft.com/store/productId/9MSVKQC78PK6?ocid=pdpshare>
  + 安装方法： 安装 WSL - Microsoft Learn
* dpkg 工具
  + 默认就在 Debian 里面带了，你只需要打开 Debian WSL 命令行，输入 dpkg 试试看，即可知道安装了没。如没安装，那随便网上搜搜就知道怎么安装了

**2.2 准备文件结构**

开始之前的准备工作完成之后，接下来就可以开始正式的打包大业了。先来明确一下咱的目标：现在咱手头上有一个 Debian 的 WSL 工具，也有使用 UI 框架的应用进行独立发布之后的文件夹，意味着其二进制文件可以直接在 UOS 上双击运行。咱需要做的就是将独立发布之后的文件夹打进安装包里面，再搭配上 UOS 规范要求的图标、应用信息等即可

根据 应用打包规范 文档中心-[统信UOS](https://so.csdn.net/so/search?q=%E7%BB%9F%E4%BF%A1UOS&spm=1001.2101.3001.7020)生态社区 文档，咱可以了解到符合 UOS 规范要求的安装包其实也是一个 deb 包。与之不同的只有文件结构组织形式，以及部分配置内容不相同而已

根据 如何构建符合要求的 UOS 软件安装包 - VVavE 文档，可以看到在 UOS 规范下的安装包的文件结构组织形式如下

├─DEBIAN

│ control

│

└─opt

└─apps

└─org.dotnetcampust.exampleapplication

│ info

│

├─entries

│ ├─applications

│ │ org.dotnetcampust.exampleapplication.desktop

│ │

│ └─icons

│ └─hicolor

│ └─24x24

│ └─apps

│ org.dotnetcampust.exampleapplication.png

│

└─files

└─bin

│ ExampleApplication

│ ExampleApplication.deps.json

│ ExampleApplication.dll

│ ExampleApplication.pdb

│ ExampleApplication.runtimeconfig.json

│ ExampleApplication.uprimarker

│ WindowsBase.dll

│ Microsoft.CSharp.dll

│ ...

本文接下来将按照文件夹组织的顺序，告诉大家如何进行组织

**2.2.1 DEBIAN 文件夹**

安装包的文件结构组织的根目录包含了 DEBIAN 和 opt 两个文件夹。其中的 DEBIAN/ 文件夹是构建过程文件夹，包含软件包构建过程相关的控制文件。在本文这里，只需放入 control 构建控制文件即可，此文件完全遵照 Debian 官方规范

为了方便大家理解，本文将对 UnoFileDownloader 下载器 进行打包作为例子，方便大家了解手动打包的细节

在开始制作安装包的时候，咱还需要给应用进行命名，即对 AppID 应用的唯一标识进行命名。在 UOS 里采用和 Android 系统类似的规范，应用商店只接受使用域名倒置规则命名的应用。请务必使用厂商的倒置域名+产品名作为应用包名，如 com.example.demo 格式，前半部分为厂商域名倒置，后半部分为产品名，如果使用非拥有者的域名作为前缀，可能会引起该域名拥有者进行申诉，导致软件被申诉下架或者删除

这里必须特别强调，只允许小写字母。且在本文下方出现的任何 AppID 或 appid 或 ${appid} 等占位符里，都需要采用此应用的唯一标识字符串，请务必确保手工打包时，各处命名都是使用相同的字符串

本文这里对 UnoFileDownloader 下载器 命名为 org.dotnetcampust.unofiledownloader 应用。尽管看起来有些不习惯且可读性较差，但 UOS 官方规范就是这样

先创建工作文件夹，接下来咱的所有文件组织结构都在此工作文件夹进行。如我这里创建的是 C:\lindexi\Work 文件夹

在工作文件夹里面先创建 DEBIAN 文件夹，请确保大小写哦，在 Linux 上是大小写敏感的

进入 DEBIAN 文件夹，再创建名为 control 的文本文件。接着打开文本编辑器，编辑 control 文件。此 control 文件要求使用 UTF-8 格式编码，且是不带 BOM 头的。所谓 BOM 头就是在文件的前两个字节里面写入编码信息，这是在 Windows 上常用的方式，原本 BOM 头是用来指示编码大小端的，后续被大家更开森用在了识别区分文件的编码上了。使用 SublimteText 编辑时，默认的配置就是不带 BOM 头的

按照 Debian 的规范，编写 control 的文件的内容，如以下例子

Package: org.dotnetcampust.unofiledownloader

Version: 1.1.3

Section: utils

Priority: optional

Architecture: amd64

Multi-Arch: foreign

Build-Depends: debhelper (>=9)

Standards-Version: 3.9.6

Maintainer: lindexi<lindexi\_gd@outlook.com>

Homepage: https://blog.lindexi.com/

Description: 下载器.

以上的 control 的各个字段/属性的说明如下，以下内容来自于 从零开始制作 deb 文件 - hzSomthing 博客

* Package : 包的唯一标识符，即 AppID 值
* Version : 安装包的版本号，即应用版本号。可使用 a.b.c 格式，也可以比较复杂的语义版本号格式，如 1.2.3-2+b1 等格式
* Section : 这是软件包的分类子类别，可以选用 utils，admin, devel, doc, libs, net, 或者 unknown 等等，代表着该软件包在 Debian 仓库中将被归属到什么样的逻辑子分类中
* Priority : optional 优先级适用于与优先级为 required、important 或 standard 的软件包不冲突的新软件包。也可以做其它取值。若是不明了，请使用 optional
* Architecture : 目标系统的 CPU 架构，一般来说是 amd64，具体情况以后会有所讨论。有时候，它们可以是 any 或者 all
* Maintainer : deb 包的维护者，及其邮件地址
* Description : 软件包的描述信息。多行也可以，前面缩进至少一个空格
* 最后一个空行 : 不能省略，否则打包提示 missing final newline 错误

在 DEBIAN/ 文件夹里面还可以包含其他的很多文件，只是本例子这里没有需求用到，就此略过。如感兴趣，请参阅 Debian 新维护者手册

**2.2.2 opt 文件夹**

完成 control 文件编写之后，接下来创建 /opt/apps/${appid}/ 文件夹，请将 ${appid} 替换为你的 AppID 值，此文件夹就是应用根目录文件夹

在本文这里也就是创建 C:\lindexi\Work\opt\apps\org.dotnetcampust.unofiledownloader\ 文件夹，请特别小心文件命名的小写问题

应用根目录下面应包含 files 和 entries 两个目录和一个 info 文件。文件结构如下

└─opt

└─apps

└─org.dotnetcampust.exampleapplication

│ info

│

├─entries

│ ├─applications

│ │ org.dotnetcampust.exampleapplication.desktop

│ │

│ └─icons

│ └─hicolor

│ └─24x24

│ └─apps

│ org.dotnetcampust.exampleapplication.png

│

└─files

└─bin

│ UnoFileDownloader.Skia.Gtk

│ ...

以下继续按照此顺序逐个为大家介绍

**2.2.2.1 files 文件夹**

先创建 files 文件夹，按照规范上所述，可以在 files 文件夹里面塞入任意的文件，也包括应用的可执行文件。规范上建议将可执行文件放入到 files/bin 文件夹，咱是听话的开发者，于是将 dotnet publish -c release -r linux-x64 --self-contained true 的输出文件夹（我的是 X:\lindexi\Code\dotnetCampus.FileDownloader\src\UnoFileDownloader\UnoFileDownloader\UnoFileDownloader.Skia.Gtk\bin\Release\net8.0\linux-x64\publish\ 文件夹）的文件全部拷贝（移动）到 files/bin 文件夹里面

**2.2.2.2 entries 文件夹**

再创建 entries 文件夹，在 entries 文件夹里面应该放入的内容是桌面/开始菜单的图标以及应用启动信息。在 entries 文件夹里面再创建 applications 文件夹，在 applications 文件夹里面创建 AppId.desktop 文本文件。同理请将 AppId.desktop 的 AppId 换成你的应用标识符，在本文例子里面，将创建的是 C:\lindexi\Work\opt\apps\org.dotnetcampust.unofiledownloader\entries\applications\org.dotnetcampust.unofiledownloader.desktop 文件

接下来打开文本编辑器，编辑 .desktop 文本文件，同理要求此文件采用 UTF-8 编码格式。以下是 org.dotnetcampust.unofiledownloader.desktop 文件例子内容，此文件当前使用的是标准的 desktop 格式，相关标准及其他字段可以参考 https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-1.1.html 文档

[Desktop Entry]

Categories=Network

Name=UnoFileDownloader

Name[zh\_CN]=简单的下载器

Keywords=deepin;uniontech;downloader

Keywords[zh\_CN]=深度;统信;下载器

Comment=The UnoFileDownloader can download file.

Comment[zh\_CN]=可以简单下载文件。

Exec=/opt/apps/org.dotnetcampust.unofiledownloader/files/bin/UnoFileDownloader.Skia.Gtk

Icon=org.dotnetcampust.unofiledownloader

Type=Application

Terminal=false

StartupNotify=true

MimeType=audio/aac;application/aac;

以下是从官方文档 里抄到的 Desktop 文件中重点字段的解释

* [Desktop Entry]: 必填 desktop文件标识，标准格式，不需要改动。
* Categories: 选填 程序分类，用于标注程序所属类型。该字段不存在时，应用属于“其他应用”。
* Name: 必填 程序在启动器中显示的通用名称，在没有单独设置语言参数的情况下，默认显示该名称。
* Keywords: 选填 程序的通用关键搜索词，当在启动器中搜索该词而非程序名称时，即可索引出该程序的快捷方式。
* Comment: 选填 关于本程序的通用简述，在没有单独设置语言参数的情况下，默认显示该段内容。
* Exec: 必填 应用的运行程序所在路径，一般为程序的可执行二进制文件或启动脚本，后期会被沙箱启动。 填写时需要关注路径的有效性，如果路径无效，在安装该程序后，程序快捷方式将不会显示在启动器中。一般都是 /opt/apps/${appid}/files/bin/Xxxxx 文件路径，这个路径请关注一下文件命名大小写问题，防止因为大小写不相同导致路径文件找不到，请将 Xxxxx 替换为你的构建输出的可执行程序
* Icon : 必填 显示的图标所在路径，推荐使用相对名称以便于系统根据主题规范查找对应的图标文件，特殊情况下(显示异常等)需要使用绝对路径来获取图标。按照图标命名的约定，基本上这里填写的就是 ${appid} 的值，不需要带上图片后缀名
* Type: 必填 程序类型，必须为“Application”。
* Terminal : 必填 该字段决定程序是否以终端的形式运行，默认为“false”关闭状态，若该项为“true”开启状态，则应用将会在终端中打开。即设置为 true 时，将以命令行控制台方式启动
* StartupNotify: 必填 程序是否支持发送启动通知事件。为“true”时，允许桌面环境跟踪应用程序的启动，提供用户反馈和其他功能。例如鼠标的等待动画等。为保障应用使用体验，该项建议填写“true”。
* MimeType : 选填 程序支持的关联文件类型，根据实际需求来填写。如果没有需要支持关联文件，则可以整行不写

Categories 应用类型可参考以下填写值来设置

* Network： 网络应用
* Chat： 社交沟通
* Audio： 音乐欣赏
* AudioVideo： 视频播放
* Graphics： 图形图像
* Game： 游戏娱乐
* Office： 办公学习
* Reading： 阅读翻译
* Development： 编程开发
* System： 系统管理

除上述分类外，填写其他分类将会被归为“其他应用”

完成 .desktop 文本文件，接下来再创建 icons 文件夹用来存放应用图标。图标可以使用 svg 矢量格式和 png 非矢量格式。尽管官方文档说的是非矢量格式建议使用 png 格式，然而实际是只能使用png格式，我试过 jpg 等其他格式都不生效，都会被替换为默认的应用图标，将 jpg 改后缀为 png 也是不行的

矢量格式图标只需放一份，路径是 entries/icons/hicolor/scalable/apps/${appid}.svg 请在放入图标之后，拷贝其路径与本文档进行对比，看是否文件夹层级正确

如果使用非矢量格式，请按照 分辨率 来放置图标，如：

1. entries/icons/hicolor/24x24/apps/${appid}.png
2. entries/icons/hicolor/16x16/apps/${appid}.png

* 支持的分辨率包括: 16/24/32/48/128/256/512 ，不同分辨率的图标会被用在不同的地方，比如桌面、通知栏、任务栏、应用抽屉中，如果没有全部包括则会自动进行缩放，影响最终效果
* 强烈建议遵循分辨率放置目录，尽量不要在 128x128 文件夹内放置其他分辨率的图标
* 如果实在没有其他分辨率资源，只有一个分辨率图标，放置在 128x128 文件夹即可

同样，在放入图标之后，请拷贝其路径与本文档进行对比。我在制作安装包时，就因为少了一层 hicolor 文件夹，调查了半天才发现是文件夹层级不对

**2.2.2.3 info 文件**

完成应用根目录下面包含的 entries 和 files 两个目录之后，接下来开始编写 info 文件。此文件是应用的描述文件，用于dde桌面环境安装软件包时自动配置程序的安装文件，使用 JSON 格式，内容一般如下

{

"appid": "org.dotnetcampust.unofiledownloader",

"name": "UnoFileDownloader",

"version": "1.1.3",

"arch": ["amd64", "arm64"],

"permissions":

{

"autostart": false,

"notification": false,

"trayicon": false,

"clipboard": false,

"account": false,

"bluetooth": false,

"camera": false,

"audio\_record": false,

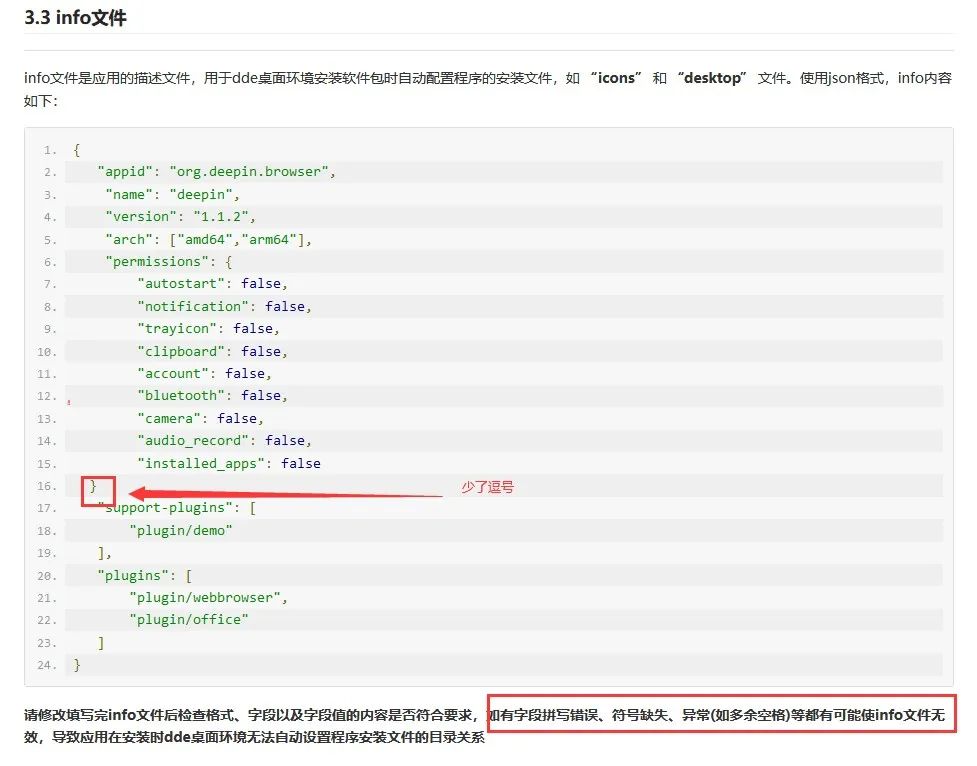
"installed\_apps": false

}

}

* appid ： 填入的就是 ${appid} 内容，作为应用标识，软件包名
* name ：应用默认名称，允许和其他应用重名
* version ：应用版本，使用 {MAJOR}.{MINOR}.{PATCH}.{BUILD} 格式，即 [主线版本].[次要版本].[补丁版本].[构建版本] 格式，所有版本号均为纯数字。此处填写的值应与deb安装包所标识的版本号一致，即和 DEBIAN/control 的 Version 相同
* arch ：应用支持的架构，目前商店支持以下几种架构 amd64, mips64el, arm64, sw\_64, loongarch64
* permissions ：应用权限。开发者需要注意，应用只允许使用普通用户权限启动，禁止应用以任何形式 root 提权。此部分只限制通过 Dbus 调用，其他方式调用则无影响。具体含义如下
* autostart : 是否允许自启动
* notification : 是否允许使用通知
* trayicon : 是否运行显示托盘图标
* clipboard : 是否允许使用剪切板
* account : 是否允许读取登录用户信息
* bluetooth : 是否允许使用蓝牙设备
* camera : 是否允许使用视频设备
* audio\_record : 是否允许进行录音
* installed\_apps : 是否允许读取安装软件列表

这里需要额外提醒大家，在 UOS 的官方文档提供的 info 文件例子里面，其文件内容的 json 格式是错误的，在 permissions 属性和下一个属性之间少了一个逗号，如果想要拷贝官方的例子，还请自行确保 json 格式正确



写到这里，文件结构就完成了，本例子完成之后的文件结构如下

C:\lindexi\Work

│

│

├─DEBIAN

│ control

│

└─opt

└─apps

└─org.dotnetcampust.unofiledownloader

│ info

│

├─entries

│ ├─applications

│ │ org.dotnetcampust.unofiledownloader.desktop

│ │

│ └─icons

│ └─hicolor

│ └─24x24

│ └─apps

│ org.dotnetcampust.unofiledownloader.png

│

└─files

└─bin

│ UnoFileDownloader.dll

│ UnoFileDownloader.pdb

│ UnoFileDownloader.Skia.Gtk

│ UnoFileDownloader.Skia.Gtk.deps.json

│ UnoFileDownloader.Skia.Gtk.dll

│ UnoFileDownloader.Skia.Gtk.pdb

│ UnoFileDownloader.Skia.Gtk.runtimeconfig.json

│ UnoFileDownloader.uprimarker

│ WindowsBase.dll

│ Microsoft.CSharp.dll

│ ...

完成了文件结构，下面将进入打包环境，开始创建 deb 包

**步骤三 打包 deb 文件**

本例子打包采用的是 Debian 的 WSL 里的 dpkg 工具进行创建 deb 文件，开始之前请确保准备好 Debian 环境。下载地址：https://www.microsoft.com/store/productId/9MSVKQC78PK6?ocid=pdpshare

先使用 cd 命令进入工作路径，如本文例子使用的 C:\lindexi\Work 文件夹

cd /mnt/c/lindexi/Work

进入其他盘符请使用 /mnt/[盘符] 的方式。值得一提的是 WSL 里面对中文支持还不错，可以放心进入中文文件夹

进入之后使用 ls 命令确保进入正确的文件夹，预期输出如下

lindexi@DESKTOP-51A5UGG:/mnt/c/lindexi/Work$ ls

DEBIAN opt

接下来输入 dpkg-deb -b . 命令进行打包

第一行是命令，第二行是结果

lindexi@DESKTOP-51A5UGG:/mnt/c/lindexi/Work$ dpkg-deb -b . UnoFileDownloader.deb

dpkg-deb: building package 'org.dotnetcampust.unofiledownloader' in 'UnoFileDownloader.deb'.

如此拿到的 UnoFileDownloader.deb 包就是可在 UOS 上使用的安装包文件

**文件权限**

在 WSL 里面 dpkg-deb 打包时，如果和我一样采用的是挂载某个磁盘路径，且磁盘是 NTFS 格式时，可能会遇到如下错误提示

dpkg-deb: error: control directory has bad permissions 777 (must be >=0755 and <=0775)

以上错误提示即使使用 chmod -R 0755 DEBIAN/control 或 chmod -R 0755 DEBIAN 命令也是没有用的。因为 wsl 如果进入的是 mnt 的 NTFS 磁盘格式下，是无法通过 chmod 设置 0755 权限的，详细请参阅 https://www.askingbox.com/question/error-message-dpkg-deb-error-control-directory-has-bad-permissions-777

For NTFS drives, this value defaults to 000, which equates to the rights of 777 and is too much here. Therefore, we change the value to umask=022, which equals the right of 755.

解决方法是在 WSL 配置里面，设置 umask=22 从而将权限转换为 755 的方式，具体步骤如下

进入 Debian 命令行，先提权，再创建或编辑 /etc/wsl.conf 文件

1. lindexi@DESKTOP-51A5UGG: sudo su
2. lindexi@DESKTOP-51A5UGG: vi /etc/wsl.conf

在 /etc/wsl.conf 文件配置 umask=22 的值，完成配置之后的文件内容大概如下，如果本身就不存在 /etc/wsl.conf 文件，则可以完全替换为下面内容，详细请参阅 https://unix.stackexchange.com/questions/252244/dpkg-deb-error-control-directory-has-bad-permissions

[automount]

enabled = true

root = /mnt/

options = "metadata,umask=22,fmask=11"

mountFsTab = false

配置完成之后，需要重启 WSL 才能生效。重启 WSL 需要使用命令行（新开一个终端）输入 wsl --shutdown 命令进行关闭，随后再启动 Debian 命令行即可完成重启

这里请不要使用网上乱抄的关闭 LxssManager 服务的方法

或者保险一些，自己重启电脑也可以

重启完成之后，再输入 dpkg-deb -b . 命令进行打包

**签名**

通过以上步骤构建所拿到的 deb 包是还没带签名的，想要进行签名，请参阅 开发者调试签名 文档中心-统信UOS生态社区

签名步骤需要进入到 UOS 系统上进行，对于开发环境来说，即使没有签名的 deb 包也是可以安装的。也就是签名这一步是可选的

在开始签名之前，请到 https://www.chinauos.com/ 注册开发者账号

接着打开统信应用商店，搜索“证书工具”，安装证书工具



安装完成之后，进入 UOS 的命令行，可使用快捷键 ctrl+alt+T 进入，输入 cert-tool -username="UOS帐号" -password="UOS密码" 进行初始化，如下面命令行

cert-tool -username=lindexi -password=123123123

完成初始化之后接下来即可对未签名的 deb 包进行签名，命令行格式如下

deepin-elf-sign-deb [deb-file]

请将 [deb-file] 替换为要签名的deb包路径

如本文例子的命令行如下

deepin-elf-sign-deb UnoFileDownloader.deb

输入之后慢慢等签名，最后看到一句成功的警告就是完成签名了

Signed successfully!

签名完成的 deb 包将会放在待签名的 deb 包目录下创建 signed\_deb 目录里面，咱可以进入此 signed\_deb 文件夹看一下签名完成之后的 deb 文件

为了确保签名已经成功，可以使用 deepin-deb-verify 命令验证签名

先使用 cd 命令进入到 signed\_deb 文件夹，再使用如下格式命令验证签名

deepin-deb-verify [签名的 deb 包]

请将 [签名的 deb 包] 替换为你实际的 deb 包，在本例子里面输入的是以下命令

deepin-deb-verify UnoFileDownloader.deb

如输出 [INFO] signature verified! 就是证明签名成功了

接下来可以试试双击此签名完成的 deb 进行安装

**常见问题**

**dpkg 提示 missing final newline 错误**

原因是 /DEBIAN/control 文件里面最后一行少了空行

**dpkg 提示 missing ‘Package’ field 错误**

可能是编辑 /DEBIAN/control 文件时，编辑器写入 BOM 头

可以使用二进制查看器，看一下 /DEBIAN/control 文件开头两个字节是否 BOM 头，如 EF BB BF 开头就是 UTF-8 BOM 头。详细关于 BOM 头，请看 BOM（字节顺序标记(ByteOrderMark)）\_百度百科

解决方法是换个文本编辑器或手动干掉 BOM 信息

**dpkg 提示 No such file or directory 错误**

详细错误信息是

dpkg-deb: error: failed to open package info file '/DEBIAN/control' for reading: No such file or directory

请确保进入了正确的文件夹，工作路径正确

**开始菜单图标空白或是默认应用图标**

原因是图标文件错误，或者是文件路径错误

比如我少了一层 hicolor 文件夹，以及将 512x512 写成 512\*512 文件夹

推荐放入图标之后，拷贝路径和本文档进行对比，了解是否放错文件夹

确保图标文件的格式在非矢量格式下使用 .png 格式，无论是后缀名还是图片格式本身，特别是从网上下载的图标。如不确定的话，可使用 MediaInfo 或 Windows 自带图片查看工具了解其文件格式

矢量 svg 图标请确保 svg 使用的是简单的格式，而不是超级复杂的 svg 格式

**应用启动闪退**

打包之前，推荐先将发布的输出文件夹拷贝到 UOS 上测试运行情况，先确保发布二进制文件本身可以正常运行

如二进制本身可以运行但开始菜单点击无法启动或无法找到安装的应用，请检查 info 文件格式、字段以及字段值的内容是否符合要求，以及 desktop 文件是否正确

**打开控制台调试**

如果期望在打包之后运行安装的应用程序进行调试，要看到输出控制台，可以在打包过程中编辑 .desktop 文本文件，设置 Terminal 为 true 的值。如此将会在启动程序时显示控制台