KYLINSOFT 麒麟软件

银河麒麟桌面操作系统软件适配

常见问题指导手册

麒麟软件有限公司

生态发展中心

2025 年 12 月 08 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	康佳楠、吴兆惠、赵鹏鹏、冯培培	2022-07-18	首次创建
V1.1	模板变更	刘佳鑫	2022-11-30	模板变更
V1.2	内容添加	唐金义	2023-06-28	内容添加
V1.3	内容添加	唐金义	2023-11-20	内容添加
V1.4	内容添加	冯培培	2025-10-23	添加 4 个常见问题、更新 5 个已有问题解决方案、增加联系方式
V1.5	模板变更	冯培培	2025-12-08	模板变更

目 录

1 目的

为了提高在银河麒麟桌面操作系统上的软件适配效率、增强麒麟软件在适配方面的服务质量，现对适配过程中遇到的有代表性的问题及解决方案进行梳理汇总，修订此指导手册，以供适配相关人员参考。

2 范围

本手册适用于第三方软件适配相关人员在与银河麒麟桌面操作系统适配过程中遇到问题时查阅参考。

3 桌面常见软件适配问题

3.1 信息查询

3.1.1 如何下载麒麟系统镜像

解决方法：登录麒麟生态网站 https://eco.kylinos.cn 点击合作伙伴-镜像下载。

备注：请先进行注册，填写信息提交申请，注册通过后即可下载系统镜像文件试用版。

3.1.2 如何查看系统版本

解决方法：

1) 使用以下命令中的一个能查到即可：

```
$ cat /etc/kylin-build
$ cat /etc/.kyinfo
```

```
kylin@kylin-QiTianM435-B733:~$ cat /etc/kylin-build
Kylin-Desktop V10-SP1
Build 20210722
```

```
kylin@kylin-QiTianM435-B733:~$ cat /etc/.kyinfo
[dist]
name=Kylin
milestone=Desktop-V10-Professional-Release-Build1-210203
arch=x86_64
beta=False
time=2021-02-03 15:22:36
dist_id=Kylin-Desktop-V10-Professional-Release-Build1-210203-x86_64-2021-02-03 15:22:36

[servicekey]
key=69844028

[os]
to=银河麒麟全体内部员工试用
term=2025-02-28
```

2）图形化：右击桌面“计算机”“属性” “通知关于” “关于”

3.1.3 如何查看系统的 cpu 信息

解决方法：使用以下命令中的一个能查到即可：

```
# cat /proc/cpuinfo
```

```
processor       : 2
vendor_id       : GenuineIntel
cpu family      : 6
model           : 165
model name      : Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
stepping        : 5
microcode       : 0xec
cpu MHz         : 3680.279
cache size      : 16384 KB
physical id     : 0
siblings        : 16
core id         : 2
cpu cores       : 8
apicid          : 4
initial apicid  : 4
fpu             : yes
fpu_exception   : yes
```

$ lscpu

```
kylin@kylin-QiTianM435-B733:~$ lscpu
架构：                           x86_64
CPU 运行模式：                   32-bit, 64-bit
字节序：                         Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
CPU：                            16
在线 CPU 列表：                  0-15
每个核的线程数：                 2
每个座的核数：                   8
座：                             1
NUMA 节点：                      1
厂商 ID：                        GenuineIntel
CPU 系列：                       6
型号：                           165
型号名称：                       Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
步进：                           5
CPU MHz：                        2794.469
CPU 最大 MHz：                   4800.0000
CPU 最小 MHz：                   800.0000
BogoMIPS:                        5799.77
虚拟化：                         VT-x
L1d 缓存：                       256 KiB
L1i 缓存：                       256 KiB
L2 缓存：                        2 MiB
L3 缓存：                        16 MiB
NUMA 节点0 CPU:                  0-15
Vulnerability Itlb multihit:     KVM: Mitigation: Split huge pages
Vulnerability L1tf:              Not affected
Vulnerability Mds:               Not affected
Vulnerability Meltdown:          Not affected
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled v
                                 ia prctl and seccomp
Vulnerability Spectre v1:        Mitigation; usercopy/swapgs barriers and __user
                                  pointer sanitization
Vulnerability Spectre v2:        Mitigation; Enhanced IBRS, IBPB conditional, RS
                                 B filling
Vulnerability Tsx async abort:   Not affected
标记：                           fpu vme de pse tsc msr pae mce cx8 apic sep mtr
                                 r pge mca cmov pat pse36 clflush dts acpi mmx f
                                 xsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rd
                                 tscp lm constant_tsc art arch_perfmon pebs bts
                                 rep_good nopl xtopology nonstop_tsc cpuid aperf
                                 mperf pni pclmulqdq dtes64 monitor ds_cpl vmx s
                                 mx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid s
                                 se4_1 sse4_2 x2apic movbe popcnt tsc_deadline_t
                                 imer aes xsave avx f16c rdrand lahf_lm abm 3dno
                                 wprefetch cpuid_fault epb invpcid_single ssbd i
                                 brs ibpb stibp ibrs_enhanced tpr_shadow vnmi fl
                                 expriority ept vpid ept_ad fsgsbase tsc_adjust
```

3.1.4 如何获取系统的架构信息

解决方法：使用以下命令中的一个能查到即可：

$ uname -a

```
kylin@kylin-QiTianM435-B733:~$ uname -a
Linux kylin-QiTianM435-B733 5.4.18-35-generic #21-KYLINOS SMP Tue Jul 20 13:33:5
8 UTC 2021 x86 64 x86 64 x86 64 GNU/Linux
```

$ arch

```
kylin@kylin-QiTianM435-B733:~$ arch
x86_64
```

3.1.5 如何使用 apt 安装本地 deb 软件包

解决方法：使用 apt 安装时要带路径信息和 deb 包全名：

```
$ sudo apt install -y ./xxx.deb
```

这里要注意的是不要直接在 apt 命令后指定 deb 包的名字，必须有路径信息，否则 apt 命令会尝试从远程仓库中搜索 deb 包同名的 package，从而导致安装失败。

3.1.6 如何获取系统内核头文件脚本

解决方法：大部分系统的内核头文件在系统安装时已经预装。

内核头文件主要在以下两个目录下：

1) /usr/src。目录名称类似于：linux-headers-5.4.18-49-generic

2) /lib/modules。目录名称类似于：5.4.18-49-generic

有些系统可能会没有预装，这时需要手动从源里进行安装。

安装方法如下：

1) 先确认系统内核版本，在终端执行 uname -a 来查看，其中标红的 5.10.0-3-generic 就是当前系统的内核版本。

```
$ uname -a
```

```
wyx@wyx-QiTianM435-N000:~$ uname -a
Linux wyx-QiTianM435-N000 5.10.0-3-generic #14~v10pro-KYLINOS SMP
    Tue Mar 16
11:02:07 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
```

2) 安装内核头文件

打开终端，执行如下命令：

```
$ sudo apt update
$ sudo apt install linux-headers-5.10.0-3-generic
```

3) 重新查看上面的两个目录是否有相应的文件。

3.1.7 如何查看系统安装时间

解决方法：

1) 首先，打开终端，查看系统安装位置，在终端中输入：

```
$ df -Th
```

```
wyx@wyx-QiTianM435-N000:~$ df -Th
文件系统        类型      容量  已用  可用 已用% 挂载点
udev            devtmpfs  7.7G     0  7.7G    0% /dev
tmpfs           tmpfs     1.6G  3.2M  1.6G    1% /run
/dev/nvme0n1p3  ext4       98G   45G   49G   49% /
tmpfs           tmpfs     7.8G  293M  7.5G    4% /dev/shm
tmpfs           tmpfs     5.0M  4.0K  5.0M    1% /run/lock
tmpfs           tmpfs     7.8G     0  7.8G    0% /sys/fs/cgroup
/dev/nvme0n1p2  ext4      976M  132M  778M   15% /boot
/dev/nvme0n1p5  ext4       91G   61M   86G    1% /data
/dev/nvme0n1p1  vfat      511M  280K  511M    1% /boot/efi
tmpfs           tmpfs     1.6G   84K  1.6G    1% /run/user/1000
```

查询/分区的挂载位置，该位置为系统安装位置（比如挂载位置为/dev/nvme0n1p3）

2) 执行以下命令查询系统安装时间

```
$ sudo dumpe2fs /dev/nvme0n1p3 |grep -i created
```

```
wyx@wyx-QiTianM435-N000:~$ sudo dumpe2fs /dev/nvme0n1p3 | grep -i created
dumpe2fs 1.45.5 (07-Jan-2020)
Filesystem created:       Tue Jun  8 13:46:34 2021
```

返回结果即为系统的安装时间。

3.2 系统安装启动问题

3.2.1 如何制作系统镜像 U 盘启动盘

3.2.1.1 系统版本

适用系统：V10、V10(SP1)

适用架构：全架构

其他版本可作参考。

3.2.1.2 问题描述

在 windows 系统上制作的系统镜像 U 盘启动盘,安装系统时无法识别到 U 盘或无法引导系统，有些可以正常引导但是安装过程中会出现问题，那如何才

会减少这种问题呢?

3.2.1.3 问题分析

在 windows 系统上使用 UltraISO 或其他启动盘制作工具一般制作的都是针对 windows 系统的启动盘，对 linux 系统不是很适用，需要使用其他软件或命令来制作，一般有如下两个方法：

1) 在 windows 系统安装 FedoraMediaWriter,使用自定义的方式制作镜像；（容易导致 U 盘制作成启动盘后在 windows 系统无法识别）；

2) 在 linux 下使用 dd 命令来制作 U 盘启动盘（推荐），下面我们就来介绍这种方法。

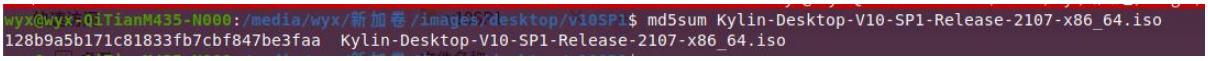
3.2.1.4 解决方案

首先我们需要找一台 linux 系统，比如银河麒麟。

1) 下载镜像，检验 MD5 码

在终端，使用 md5sum ***.iso 命令来查看镜像文件的 md5 值（***.iso 指的是下载的 iso 镜像），查看是否和下载时提供的 MD5 值是否一致，如果一致则镜像完整，如果不一致说明镜像下载不完整，需要重新下载。

```
$ md5sum ***.iso
```



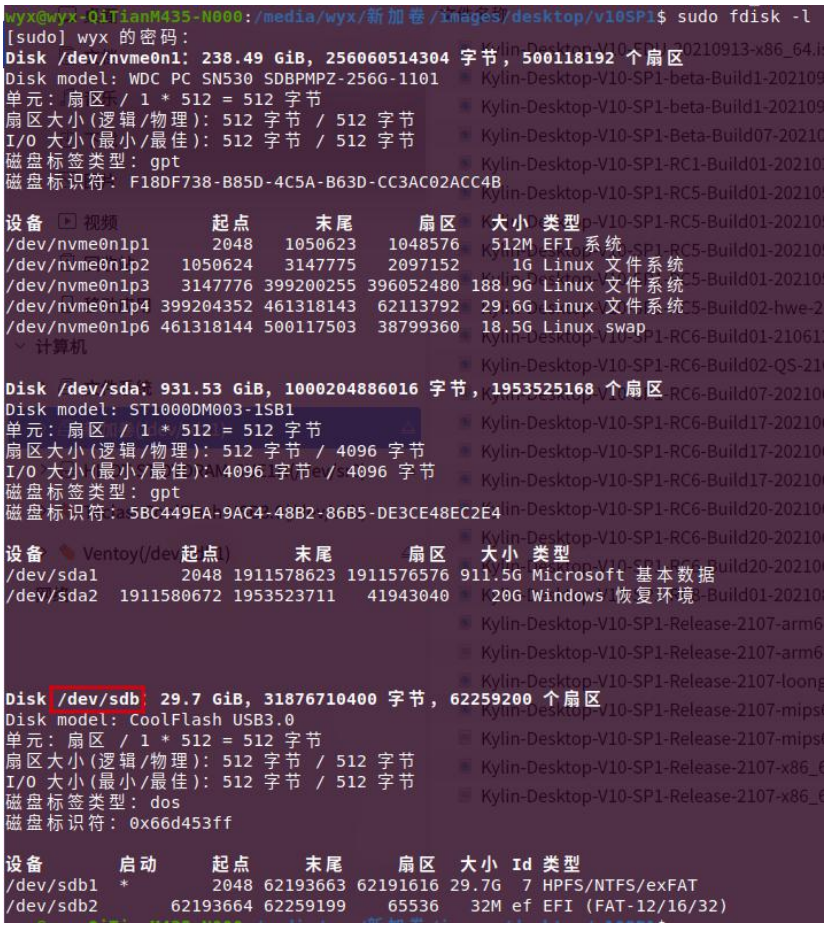
2) U 盘连接电脑

将 U 盘插入电脑。

3) 查看挂载的 U 盘设备路径：

使用命令 sudo fdisk -l 查看 U 盘挂载的设备路径,此处可以看到是 /dev/sdb，一般都是这个，有些可能是 sda 或 sdc。

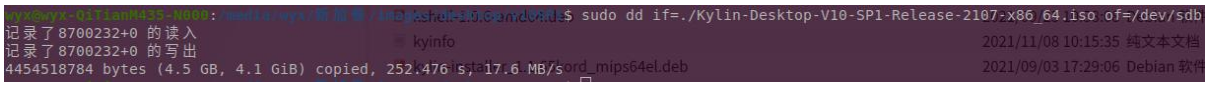
```
$ sudo fdisk -l
```



4) 写入镜像

```
$ sudo dd if=/***.iso of=/dev/xxx
```

其中，if 后接镜像文件路径，of 后接写入 U 盘的路径



如上图所示，这种代表已经写入完成了，现在就可以拿着 U 盘去做系统了。

3.2.2 如何在物理机安装 V10 操作系统

3.2.2.1 系统版本

适用系统：V10

适用架构：X86、ARM、MIPS

其他版本和架构可作参考。

3.2.2.2 问题描述

如何安装麒麟 V10 操作系统？

3.2.2.3 问题分析

先制作启动盘，然后根据步骤提示一步步进行安装。

3.2.2.4 解决方案

1) 系统启动盘制作

需要工具:

a. 系统镜像；

b. U 盘或光盘(推荐 U 盘制作)；

c. 烧录工具

启动盘制作：

a. linux 系统：

Linux 系统可使用 dd 命令进行启动盘制作

dd if=iso 镜像绝对路径 of=/dev/U 盘 bs=4M

b. windows 系统：

windows 系统推荐使用烧录工具(fedora media writer)制作，烧录工具已添加至附件。可以双击安装使用。

FedoraMediaWriter-win32-4.0.7-4-g7cc3c2b.rar

下载链接：https://pan.baidu.com/s/1bkuxyiPdlmSdb_PH-KkXLg

提取码：3g9r

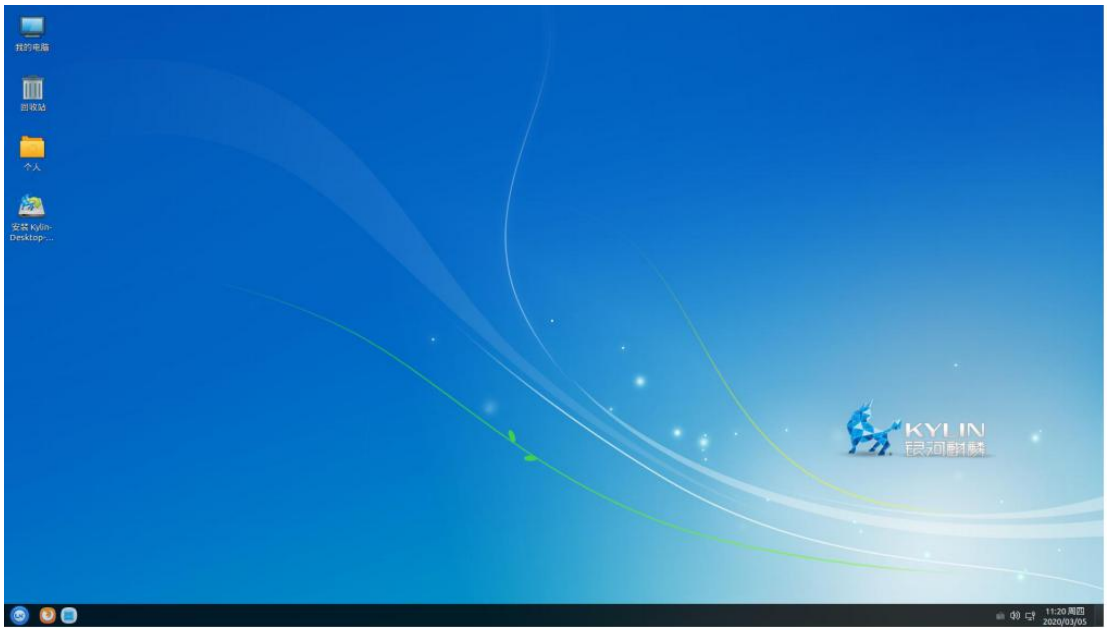
2) 启动引导

将安装光盘放入光驱中，重启机器。根据固件启动时的提醒，进入固件管理界面。若使用的是内置光驱，“第一启动选项”选择“光驱”；

若使用的是 USB 或者 USB 外置光驱，“第一启动选项”选择“USB”。本系统支持体验模式，可试用一个全功能的操作系统而不安装。

3) 系统安装

a. 双击图标“安装 Kylin-Desktop-V10”，开始安装引导。



b. 此处可选择语言，点击“继续”

c. 勾选同意许可协议，再点击“继续”，进入安装方式选择，选择“从 Live

镜像安装”，点击“继续”，进入安装类型界面。

以下是对 4 个选项的详细介绍：

①“创建备份还原分区”：挂载点为“/backup”。勾选后，选择“快速安装 Kylin”时，分区大小默认与根分区相同。只有创建了该分区，备份还原功能才可以使用。备份还原对用户恢复数据或系统非常有帮助，建议创建。

②“创建数据盘”：挂载点为“/data”。勾选后，选择“快速安装 Kylin”时，分区大小为整个磁盘除掉其他分区外的所有空间。/data 类似于 Windows 系统除 C 盘外的其他盘符，建议创建。注意，a、b 两个选项的勾选，是针对快

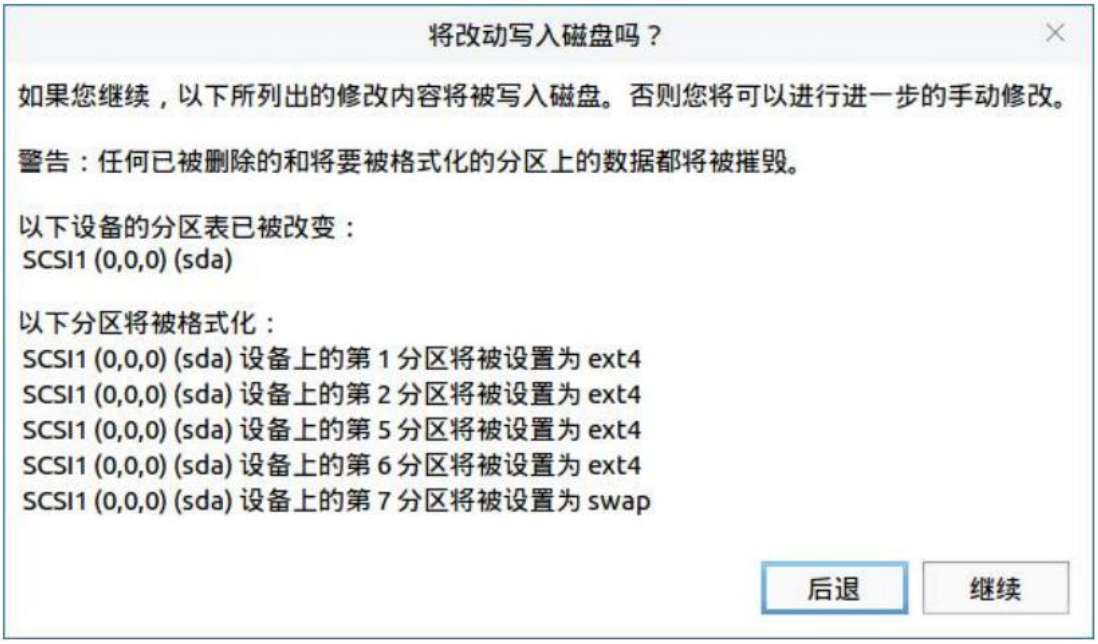
速安装的设置。

③“高级安装”：用户自行根据实际需求，进行分区创建和分区大小分配。详细说明见第 5 部分。

④“快速安装 Kylin”：全盘安装，该选项将会格式化整个硬盘，并进行自动分区。

当选择“快速安装 Kylin”选项时

①同时勾选“创建备份还原分区”和“创建数据盘”，点击“现在安装”按钮，弹出格式化分区警告信息，然后，点击“继续”（此时硬盘已经被格式化和重新分区），弹出创建用户信息窗口。



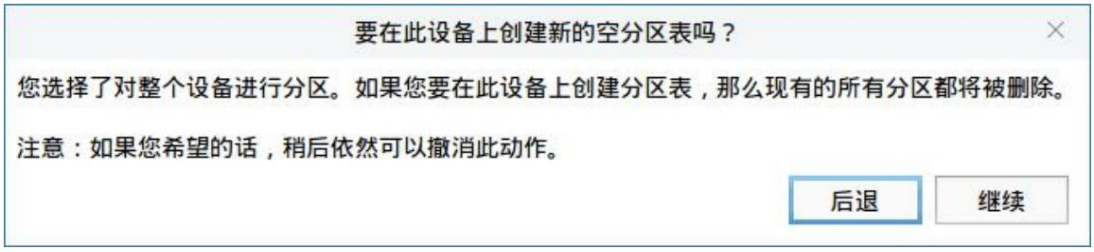
②信息正确填完后，“继续”按钮由灰变亮，点击“继续”按钮，此时会将系统信息写入硬盘。

③安装完成后，会弹出提示窗口。点击“现在重启”按钮，系统会重新启动。

重启过程中系统会自动弹出光驱或提示拔出 U 盘。取回光驱或 U 盘后，等待系统进入登录界面，输入密码后即可进入系统。

当选择“高级安装”选项时

①在安装类型界面选择“高级安装”，点击“继续”，则出现硬盘分区界面。点击“新建分区表”，弹出提示窗口，选择“继续”，即可创建硬盘分区。

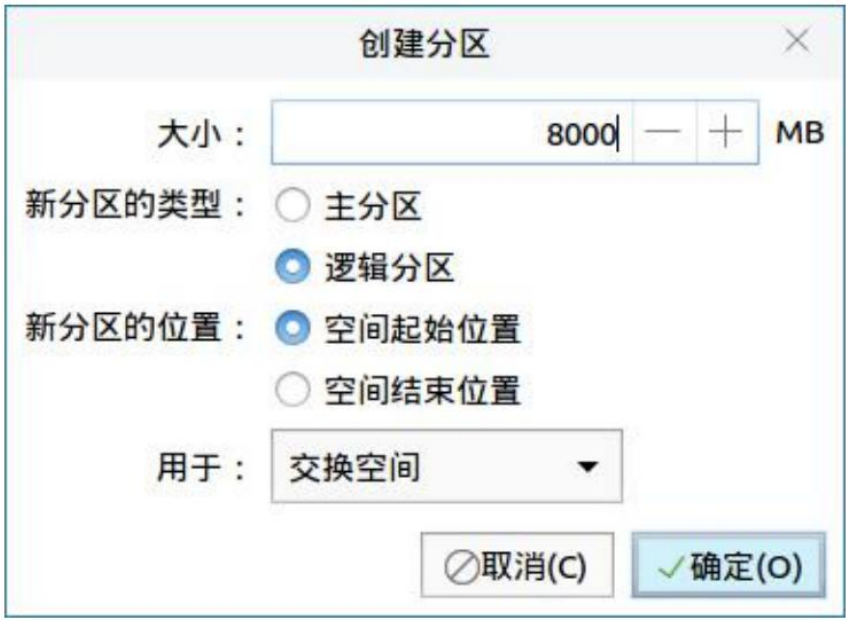


②选中“空闲”所在行，此时“+”由灰变亮。点击“+”，则弹出创建分区窗口，开始创建分区。

需要注意的是，/boot 必须是主分区中的第一个分区。

③在创建根分区的时候，“新分区的类型”选择“主分区”，“新分区的位置”默认为“空间起始位置”，“用于”选择“Ext4 日志文件系统”；

④交换分区大小一般设置为内存的 2 倍大小，“新分区的类型”选择“逻辑分区，“新分区的位置”保持默认，“用于”选择“交换空间”；



⑤不管是否勾选“创建备份还原分区”和“创建数据盘”，用户都可以创建“/backup”分区和“/data”分区。这两个分区创建时，“新分区的类型”选

择“逻辑分区”，“新分区的位置”默认为“空间起始位置”，“用于”选择“Ext4 日志文件系统”，挂载点选择对应的/backup、/data 即可；

建议/backup 分区和根分区大小一致。

⑥另外，建议用户创建 EFI 系统分区。EFI 分区大小应在 100M-2G 之间

注：若是中途需要改变已创建的分区，具体方式如下所示：

①添加分区：选中空闲分区所在行，点击“+”按钮。

②编辑分区：选中已创建的分区，点击“更改”按钮。

③删除分区：选中已创建的分区，点击“-”按钮。

分区完成后，点击“现在安装”。

3.2.3 如何在物理机安装 V10(SP1)操作系统

3.2.3.1 系统版本

适用系统：V10（SP1）

适用架构：X86、ARM、MIPS

其他版本和架构可做参考。

3.2.3.2 问题描述

如何安装麒麟 V10(SP1)操作系统？

3.2.3.3 问题分析

1) 先制作启动盘，然后根据步骤提示一步步进行安装。

3.2.3.4 解决方案

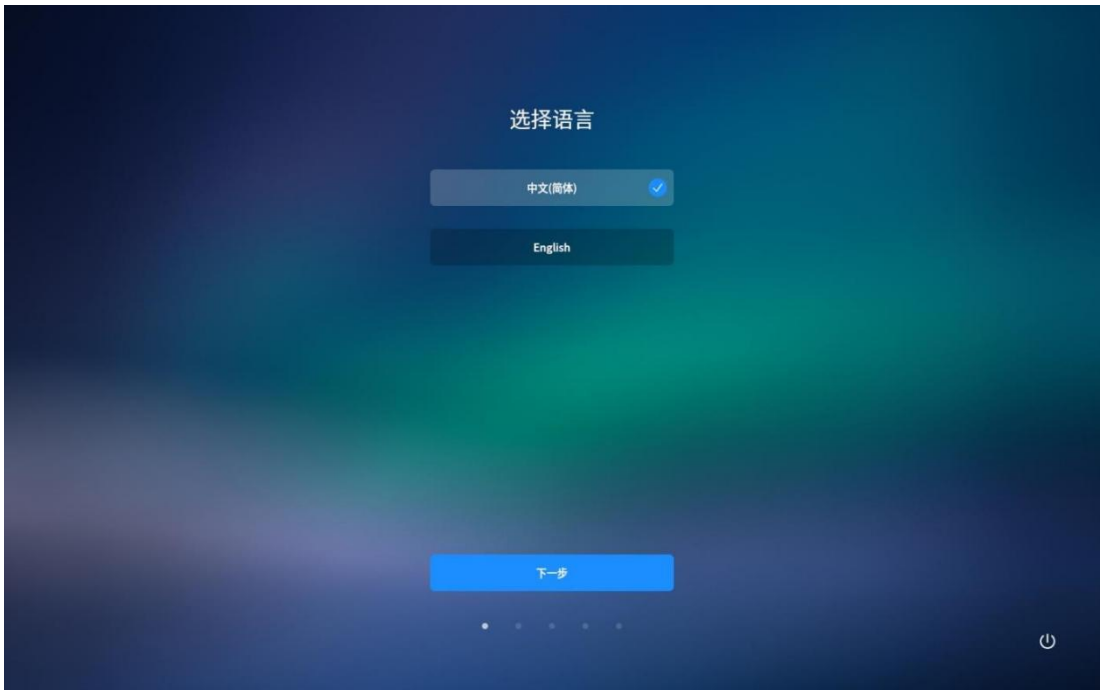
启动盘制作可参考 3.2.1.4 中的系统盘制作。

具体安装过程和产品激活如下：

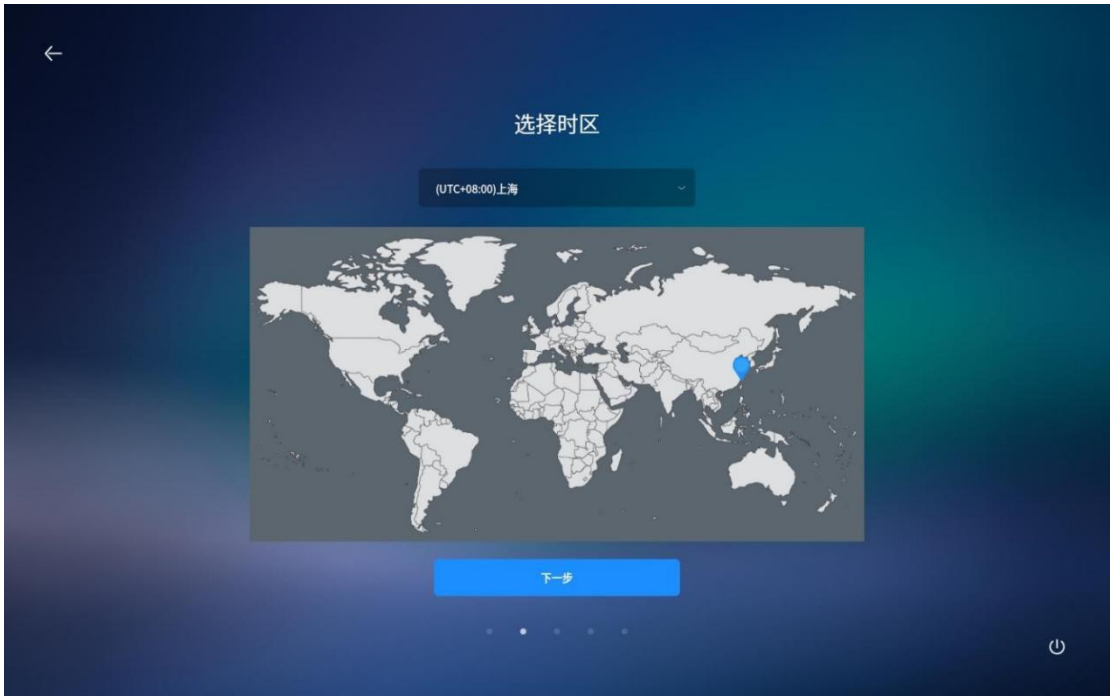
一、安装过程

将安装光盘放入光驱中，重启机器。根据固件启动时的提醒，进入固件管理界面。若使用的是内置光驱，“第一启动选项”选择“光驱”；若使用的是 USB 或者 USB 外置光驱，“第一启动选项”选择“USB”，然后根据引导进行安装。

1) 选择安装语言



2） 选择时区



3） 下拉滚动条至底部阅读许可协议并勾选，点击【下一步】。

4) 创建本计算机上第一个用户。

5) 选择安装方式，默认快速安装。如使用快速安装，将会覆盖电脑中原有的系统，请务必做好数据备份，且保证安装磁盘可用空间大于 50G。

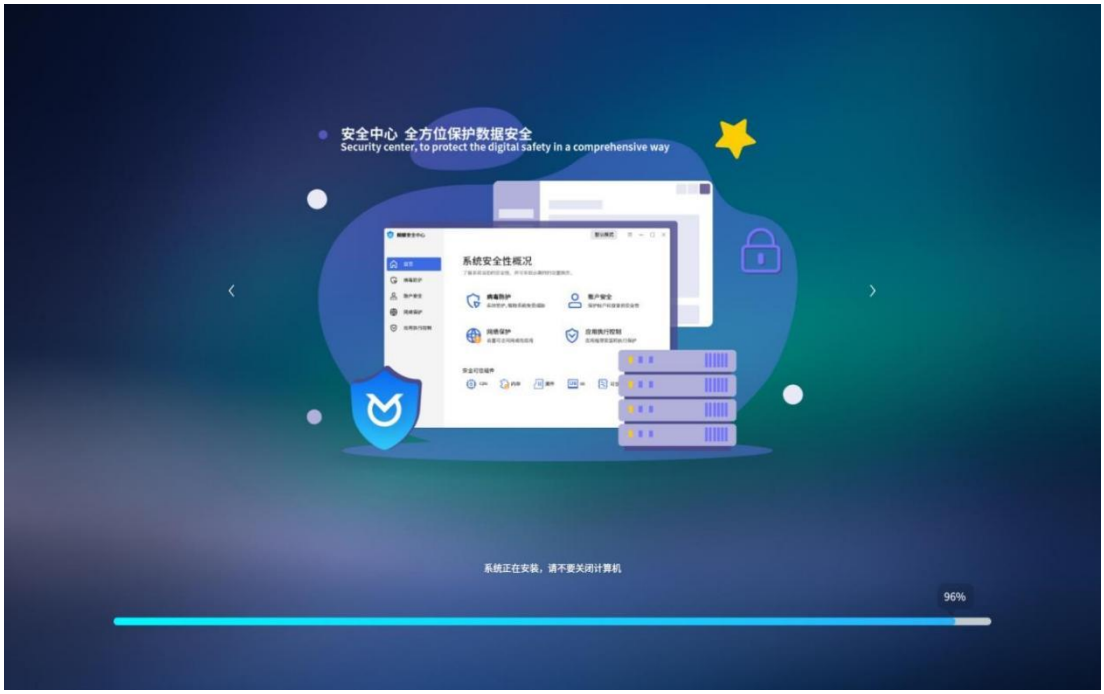
6) 如需使用多个系统，可选择自定义安装方式，设定分区方案。安装需创建基本的 linux-swap 和根分区，请保证根分区大小至少为 15G，如系统无 efi 分区，同时需要创建 efi 分区。

7) 点击【＋】创建 linux-swap，默认 2G。

8) 点击【＋】创建根分区，可将剩余空间分配至其中。

9) 分配完毕后点击【下一步】，查看分区结果。

10)点击【开始安装】后，执行系统安装。

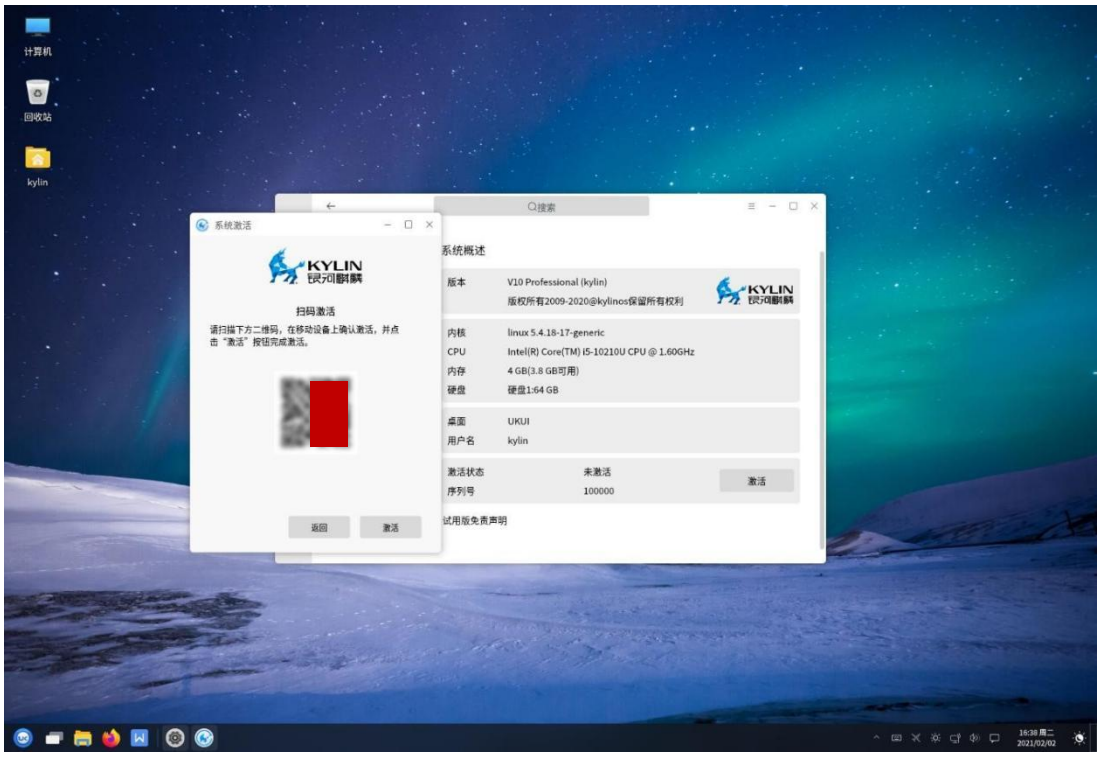


11)完成安装后重启计算机。

二、产品激活

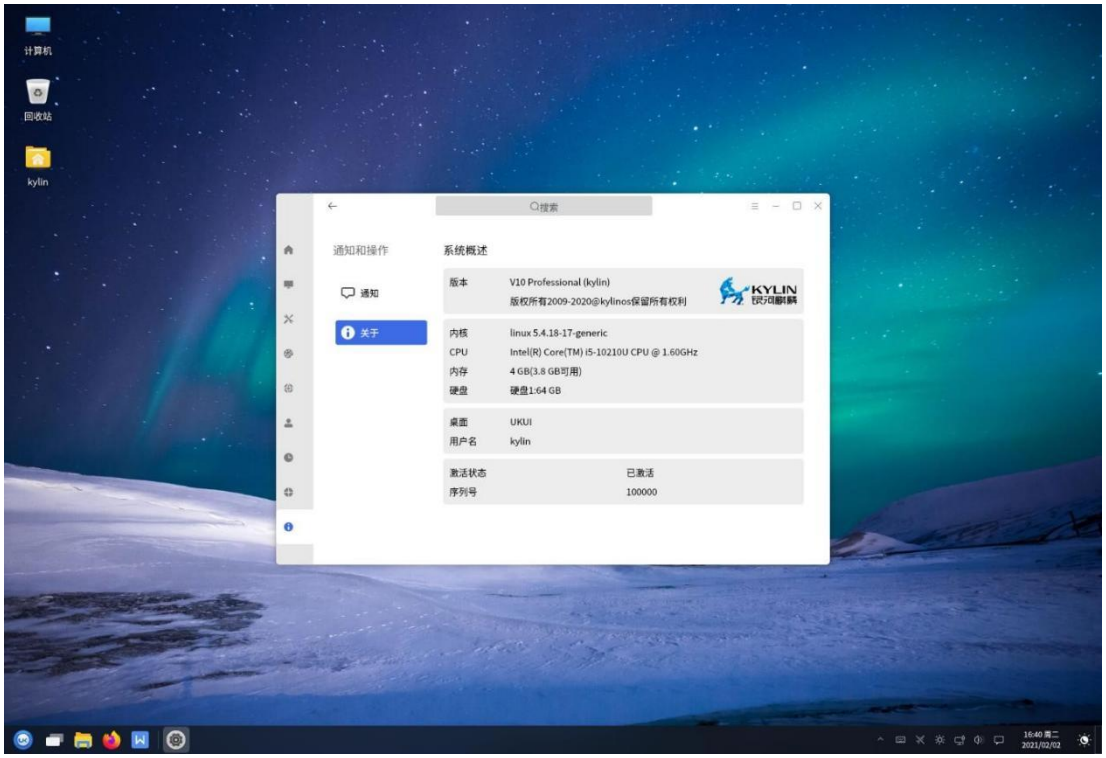
连接网络后，在【计算机】右键点击【属性】打开【关于】菜单，点击【激活】，使用移动设备扫码，在移动端输入企业内部邮箱并绑定，桌面点击【激活】按钮完成激活。

1) 系统激活



2） 微信扫码绑定

3） 激活完成



3.2.4 系统安装报错问题排查方法

3.2.4.1 系统版本

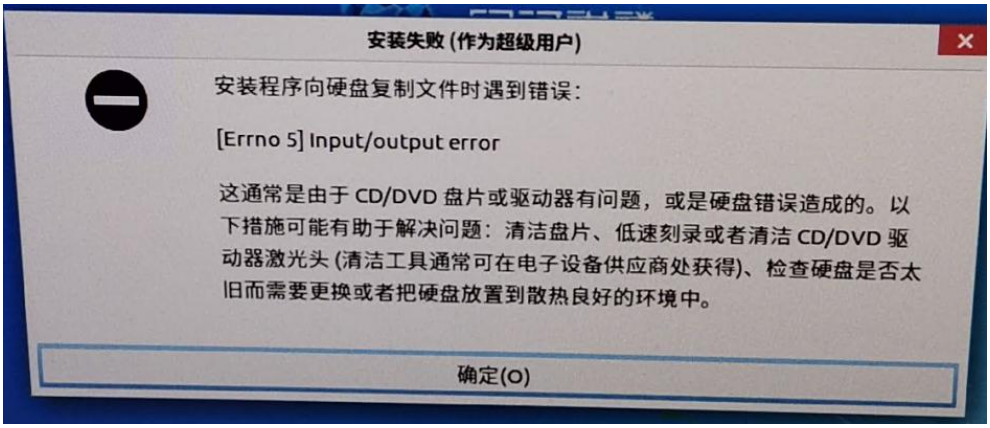
适用系统：V10（SP1）

适用架构：X86、ARM、MIPS

其他版本和架构可做参考。

3.2.4.2 问题描述

系统安装过程中报错如下：“提示：安装程序向硬盘复制文件时遇到错误：[Errno 5]Input/output error”。



3.2.4.3 问题分析

该问题可能由三个情况导致：

1) 镜像不完整；

2) 硬盘空间不充足；

3) 磁盘有问题。

3.2.4.4 解决方案

1) 此问题一般是由于镜像不完整导致，通过核对镜像大小或校验镜像 MD5 值确认镜像是否完整，如果不完整需要重新下载镜像后重新安装；

2) 如果排查镜像完整且做的启动盘没有问题，可以检查硬盘大小是否充足，一般需要 50G 的空间；

3) 上述都没问题的情况下，可以对磁盘进行格式化后重新安装。

3.2.5 如何解决忘记密码问题

3.2.5.1 系统版本

适用系统：V10

适用架构：X86、ARM、MIPS

其他版本和架构可做参考。

3.2.5.2 问题描述

V10 系统忘记用户密码，无法登录系统。

3.2.5.3 问题分析

进入 recovery mode，修改用户密码。

3.2.5.4 解决方案

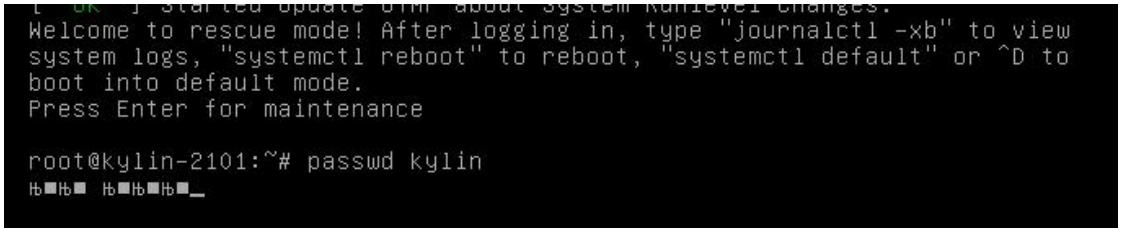
开机进入“Kylin V10 高级选项”，然后选择“recovery mode”,系统会进入恢复模式，使用 passwd 用户名修改密码。

1) 开机进入“Kylin V10 高级选项”

2) 选择“recovery mode”,系统会进入恢复模式

使用命令

```
$ passwd 用户名
```



输入两次密码

然后输入 reboot 重启：

```
$ reboot
```

3.2.6 如何解决系统登录界面输入密码无法进入系统

3.2.6.1 系统版本

适用系统：V10、V10（SP1）

适用架构：X86、ARM、MIPS

其他版本和架构可做参考。

3.2.6.2 问题描述

进入系统的登陆界面，输入正确的用户名和密码，屏幕闪烁后回到登陆界面，无法进入桌面。

3.2.6.3 问题分析

有可能是多方面的问题：

1) 认证问题

2) 环境变量问题

3) 磁盘空间不足问题

4) ……

3.2.6.4 解决方案

1) 认证问题

删除.Xauthority 文件

操作方法如下：

a. 使用快捷键 Ctrl+Alt+F1 进入命令行界面，输入用户名和密码（用户名为出现问题的用户，不同用户的环境不同）。

b. 删除当前用户主目录下的.Xauthority 文件；

c. 重启

2) 环境变量问题

删除/etc/profile 中后添加的环境变量

操作方法如下：

a. 使用快捷键 Ctrl+Alt+F1 进入命令行界面，输入用户名和密码（用户名为出现问题的用户，不同用户的环境不同）。

b. 删除多余的环境变量；

```
$ sudo vim /etc/profile
```

c. 重启

3) 磁盘空间不足问题

操作方法如下：

a. 使用快捷键 Ctrl+Alt+F1 进入命令行界面，输入用户名和密码（用户名为出现问题的用户，不同用户的环境不同）。

b. 使用 df -h 查看磁盘使用情况，如果磁盘占用过高，使用 rm -rf 删除部分文件释放磁盘空间；

c. 重启

3.2.7 如何解决磁盘空间不足导致无法进入系统桌面的问题

3.2.7.1 系统版本

适用系统：V10（SP1）

适用架构：X86、ARM、MIPS

其他版本和架构可做参考。

3.2.7.2 问题描述

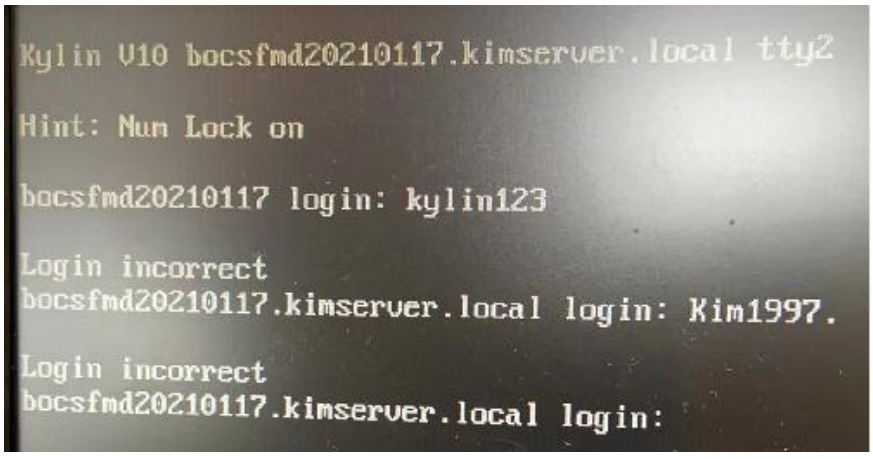
如何解决系统磁盘空间不足导致无法进入系统桌面的问题?

3.2.7.3 问题分析

系统磁盘空间不足，需要删除无用的文件，释放磁盘空间。

3.2.7.4 解决方案

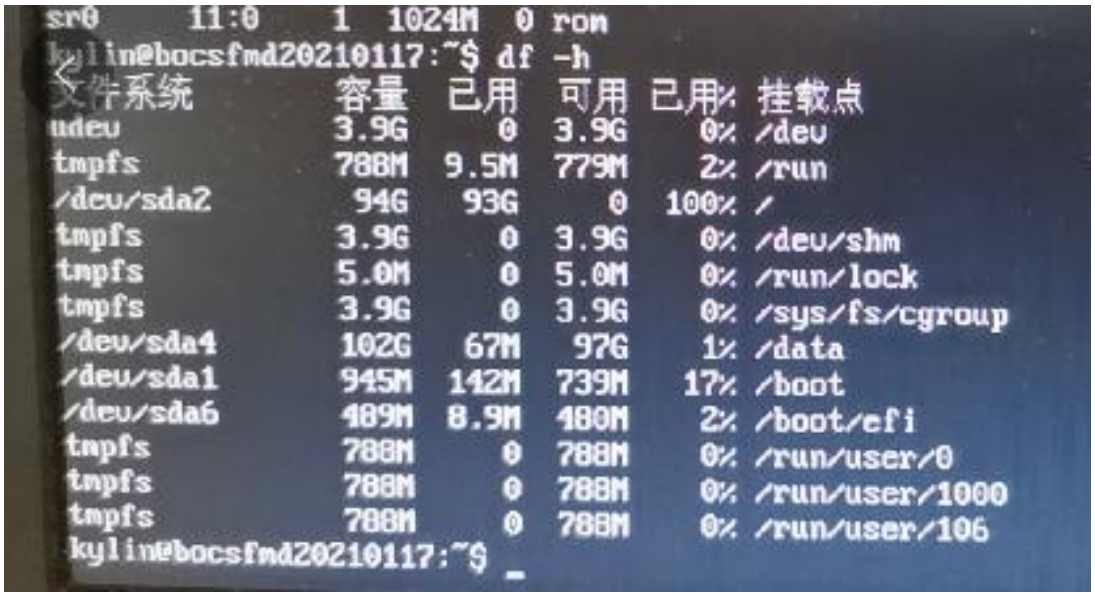
1) 使用 ctrl+alt+F2 命令进入命令行界面



2) 输入用户名和密码进入系统

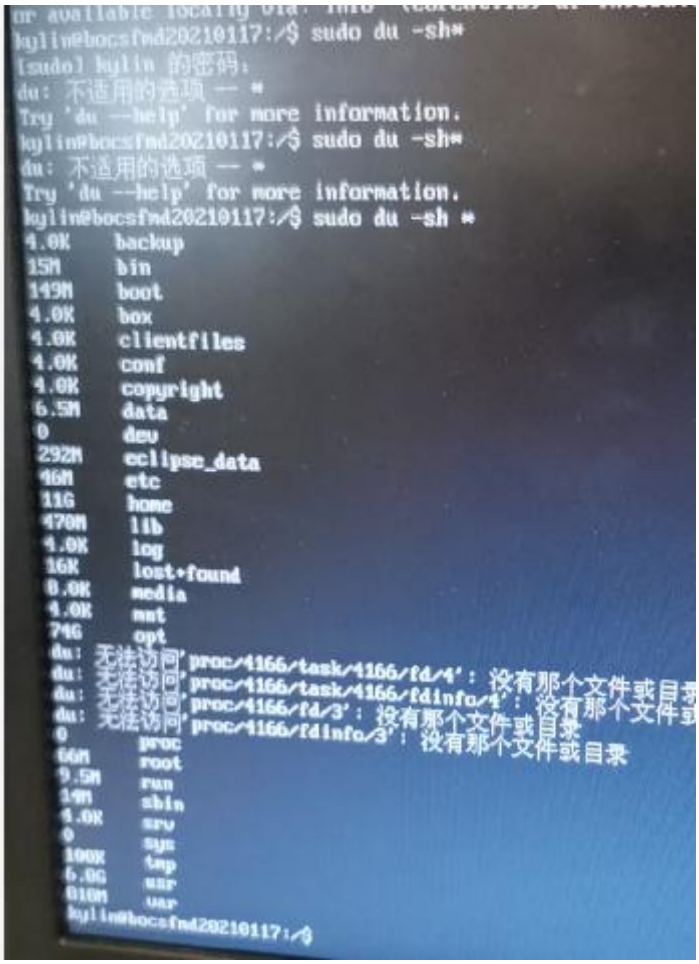
3) 使用 df -h 查看目录磁盘空间

```
$ df -h
```



4) 进入磁盘满的目录，使用 du -sh *列出当前目录下所有文件和目录的大小

```
$ sudo du -sh *
```



5) 查看比较大的目录，删除无用的文件

3.3 开发编译问题

3.3.1 系统如何自定义添加桌面右键菜单选项

3.3.1.1 系统版本

适用系统：V10（SP1）

适用架构：X86、ARM、MIPS

其他版本和架构可做参考。

3.3.1.2 问题描述

系统如何自定义添加桌面右键菜单选项？

3.3.1.3 问题分析

可以根据研发提供的 demo 编译产生。

3.3.1.4 解决方案

使用下面的这个 demo 编译就可以看到效果

peony-qt-menu-plugin-example.tar.gz

下载链接：https://pan.baidu.com/s/1vTegb4xfDfdr3u2zqgSZDA

提取码：9ejm

1) 解压

```
$ mkdir peony
$ cd peony
```

将上面的 demo 放入此目录

```
$ tar -zxvf peony-qt-menu-plugin-example.tar.gz
```

2) 编译

a. 执行 qmake 进行编译，依赖安装不全会有报错，需要使用 pkg-config --cflags peony 来进行查询依赖后然后进行安装。

```
$ qmake
$ pkg-config --cflags peony
```

```
kylin@kylin-VMware-Virtual-Platform:~/peony$ qmake
Info: creating stash file /home/kylin/peony/.qmake.stash
Project ERROR: peony development package not found
kylin@kylin-VMware-Virtual-Platform:~/peony$ pkg-config --cflags peony
Package peony was not found in the pkg-config search path.
Perhaps you should add the directory containing `peony.pc'
to the PKG_CONFIG_PATH environment variable
No package 'peony' found
kylin@kylin-VMware-Virtual-Platform:~/peony$ sudo apt install libpeony-dev
```

b. 通过几次查询一般需要安装以下依赖：

```
$ sudo apt install libpeony-dev
$ sudo apt install libpoppler-qt5-dev
$ sudo apt install libudisks2-dev
$ sudo apt install libnotify-dev
```

c. 安装完成后再重新执行 qmake，会生成 Makefile 文件:

```
$ qmake
```

d. 然后执行 make 进行编译，编译完成后会生成一个 libpeony-qt-menu-plugin-example.so 的库文件；

```
$ make
```

3) 安装

```
$ sudo make install
```

```
kylin@kylin-VMware-Virtual-Platform:~/peony$ sudo make install
[sudo] kylin 的密码：
/usr/lib/qt5/bin/qmake -install qinstall -exe libpeony-qt-menu-plugin-example.so /usr/lib/x86_64-linux-gnu/peony-extensions/libpeony-qt-menu-plugin-example.so
strip --strip-unneeded /usr/lib/x86_64-linux-gnu/peony-extensions/libpeony-qt-menu-plugin-example.so
kylin@kylin-VMware-Virtual-Platform:~/peony$
```

4） 查看

注销或重启系统后，在桌面点击右键查看选项，可以看到多出来的两个菜单选项。

3.3.2 如何解决 QtCreator 示例不显示问题

3.3.2.1 系统版本

适用系统：V10、V10（SP1）

适用架构：X86、ARM、MIPS

其他版本和架构可做参考。

3.3.2.2 问题描述

安装 qtcreator 后，打开 qtcreator，点击示例，没有示例显示。

3.3.2.3 问题分析

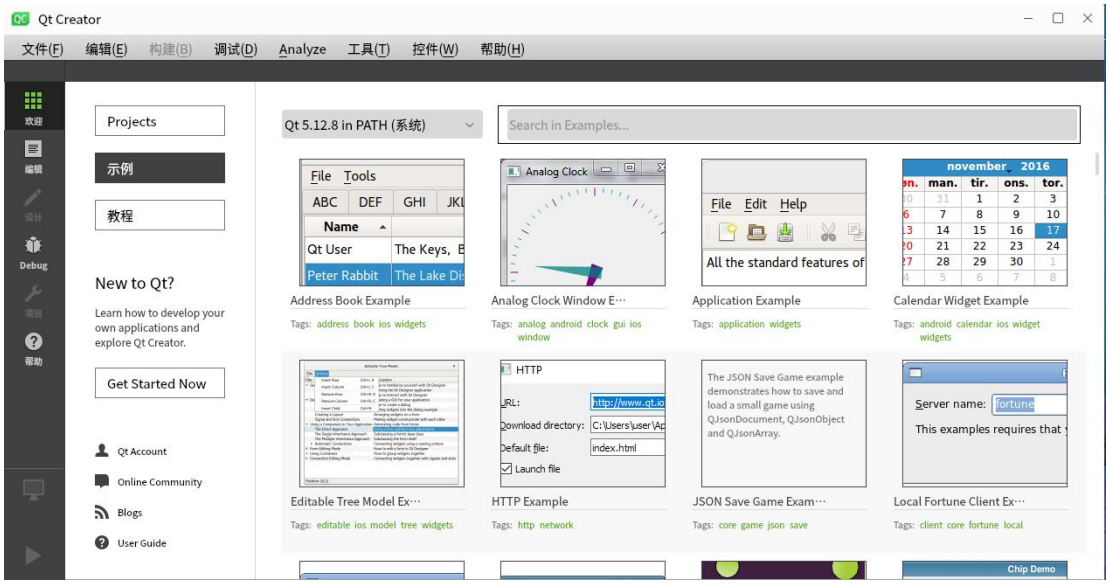
系统中缺少 qtbase5-examples qtbase5-doc-html qtbase5-doc 相关库文件。

3.3.2.4 解决方案

打开终端，执行

```
$ sudo apt install qtbase5-examples qtbase5-doc-html qtbase5-doc
```

安装相关库之后，重新打开 qtcreator 即可。



3.3.3 如何编译运行 QtCreator 中的示例

3.3.3.1 系统版本

适用系统：V10、V10（SP1）

适用架构：X86、ARM、MIPS

其他版本和架构可作参考。

3.3.3.2 问题描述

如何运行 qtcreator 中的 example？为什么有些 example 编译不通过？

3.3.3.3 问题分析

1) 安装 qtcreator 不会自动安装 examples，example 需要单独安装；

2) example 编译过程中会需要很多依赖，有些依赖系统没有自带，需要单独安装。

3.3.3.4 解决方案

下面就举一个例子来介绍一下如何来编译运行一个 Qt 的 example，此处我们以编译自带的 webengine 的 example 为例。

1) 安装 qt5 基础包

```
$ sudo apt install qt5-default
```

2) 安装 cmake 编译工具

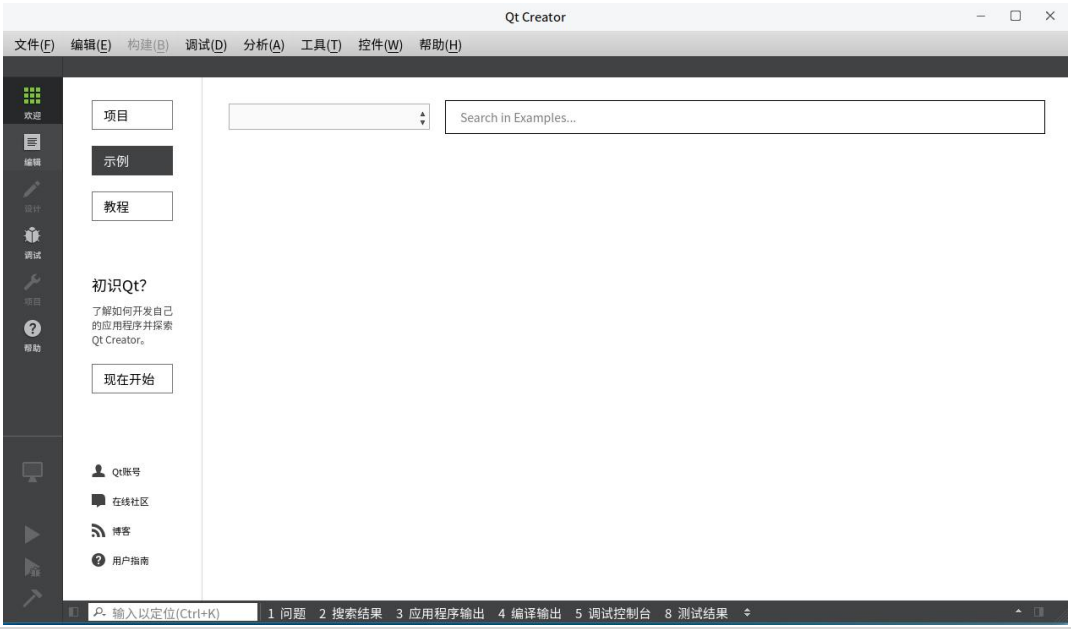
```
$ sudo apt install cmake
```

3) 安装 qtcreator

```
$ sudo apt install qtcreator
```

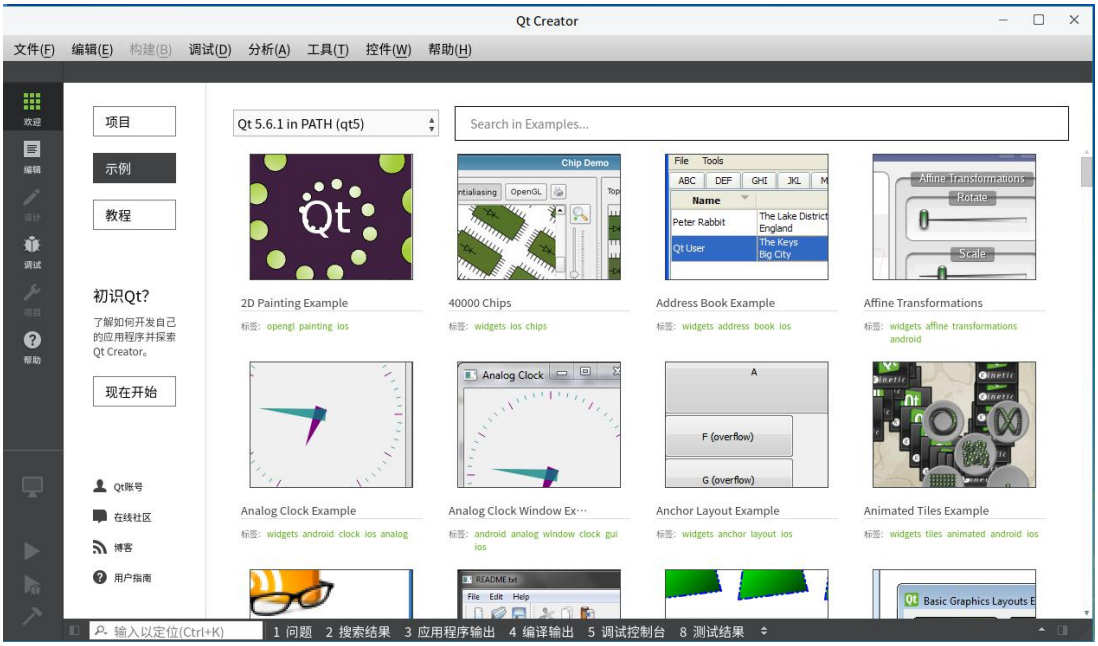
4) 安装 qt 基础 example 及文档介绍

安装完上面的包打开 qtcreator，点击示例是看不到示例的，需要安装基础的示例包。




```
$ sudo apt install qtbase5-examples qtbase5-doc qtbase5-doc-html
```

安装完成后重新打开 qtcreator，点击示例可以看到一些基础的 example。



5) 安装 qtwebengine5-examples 及相关的包

```
$ sudo apt install qtwebengine5-* -y
```

（包括：qtwebengine5-examples qtwebengine5-doc qtwebengine5-doc-html qtwebengine5-dev）

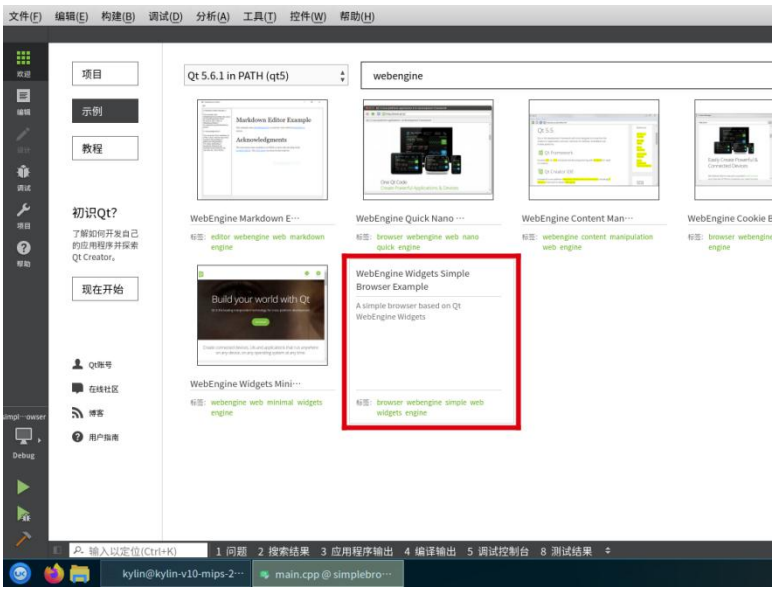
注：

示例代码包名：qtwebengine5-examples

qtcreator 示例显示相关包名：qtwebengine5-doc qtwebengine5-doc-html

开发库包名：qtwebengine5-dev；

6）搜索示例并打开项目

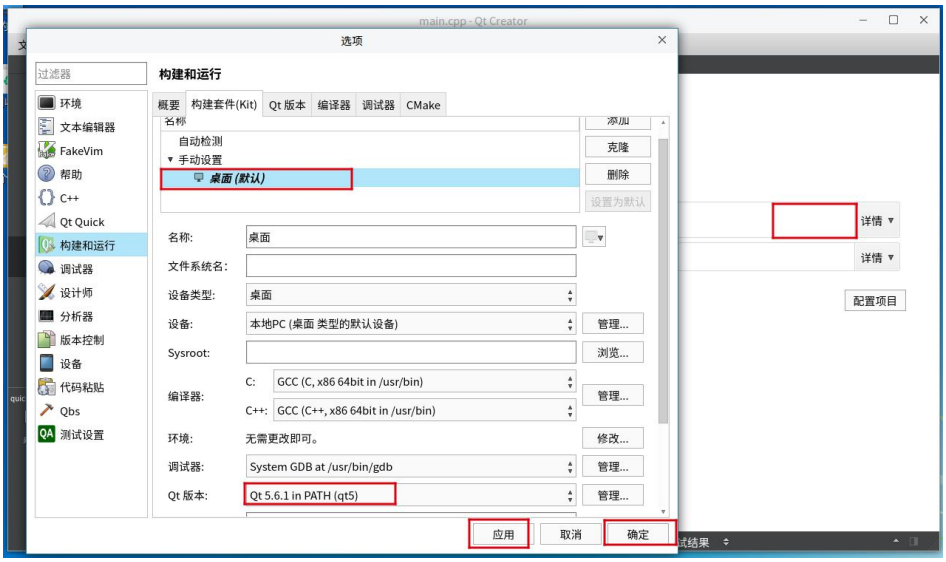


关闭帮助

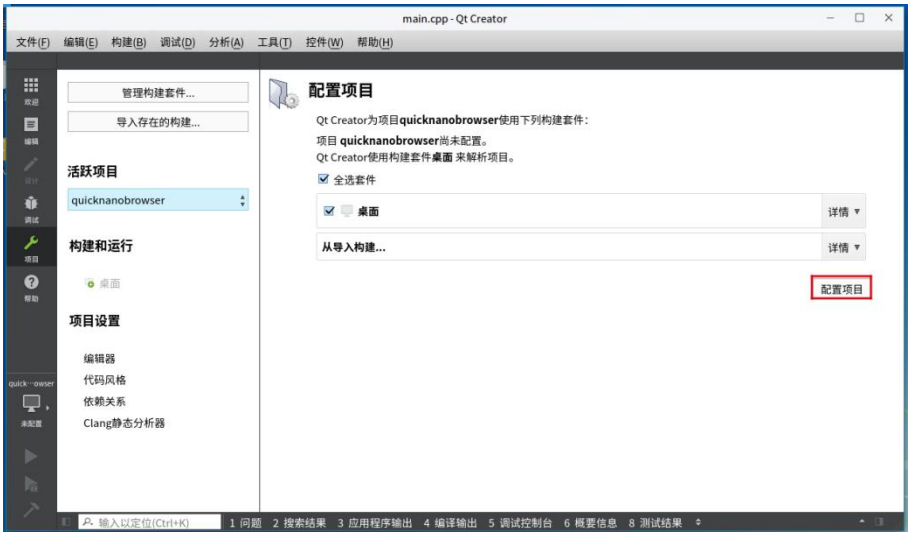
7）配置项目

点击管理，进入选项界面。

选择 qt 版本为 5.6.1，点击“应用”-“确定”

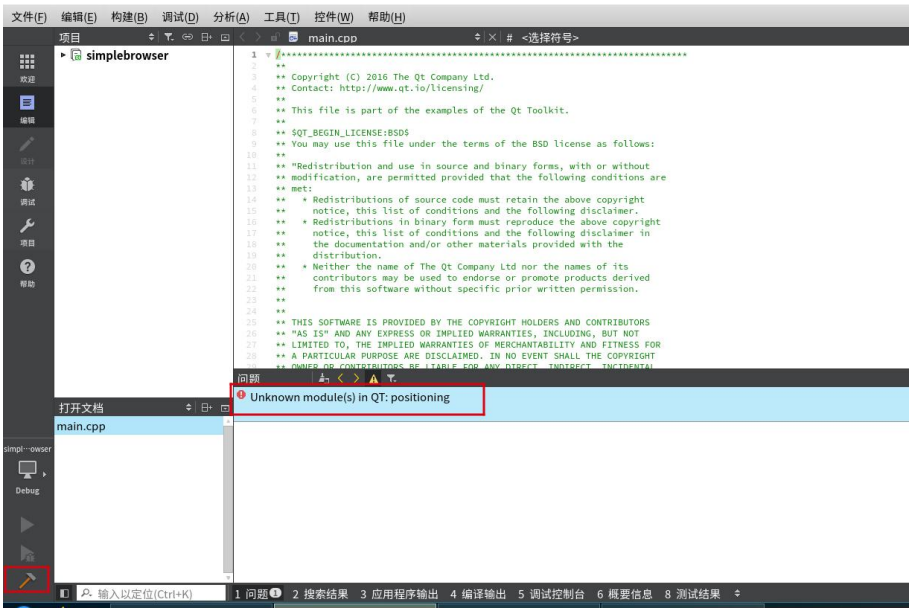


然后点击“配置项目”



8) 构建项目

点击左下角的“构建”按钮进行构建，此时会有如下报错，是由于缺少相关的依赖库导致，安装相应的依赖库即可。

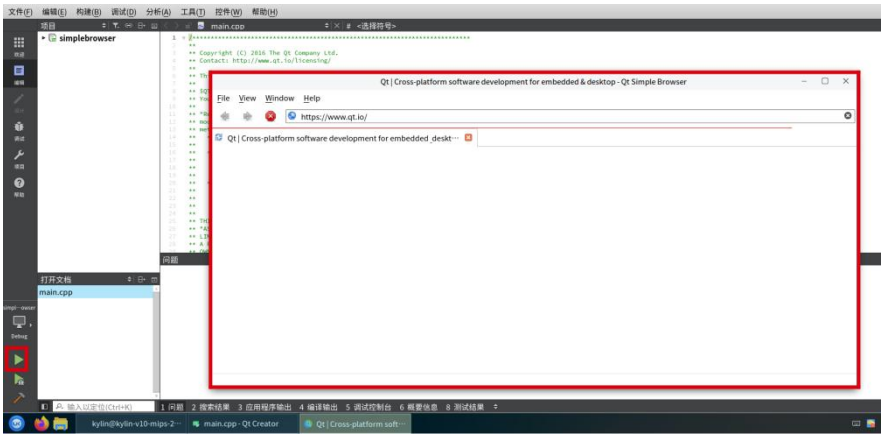



```
$ sudo apt install qtpositioning5-dev
```

安装完成后，点击“构建”重新进行构建，此时没有报错（右下角绿色）即表示构建完成，如果构建完成后左侧的“运行”和“开始调试”按钮为灰色，点击 Debug 切换到 Release 后，重新切换回来即可。

9) 运行

点击“运行”按钮来运行项目，此时可以看到一个简易的浏览器已经打开了。



3.3.4 编译 Qt 程序提示缺少 positoning

3.3.4.1 系统版本

适用系统：V10、V10（SP1）

适用架构：X86、ARM、MIPS

其他版本和架构可作参考。

3.3.4.2 问题描述

Project ERROR: Unknown module(s) in QT: positioning

3.3.4.3 问题分析

系统中缺少相关开发库。

3.3.4.4 解决方案

安装相关工具：

```
$ sudo apt install qtpositioning5-dev
```

3.3.5 编译 Qt 程序提示缺少 quick

3.3.5.1 系统版本

适用系统：V10、V10（SP1）

适用架构：X86、ARM、MIPS

其他版本和架构可作参考。

3.3.5.2 问题描述

error: Unknown module(s) in QT: qml quick - Qt Creator.

3.3.5.3 问题分析

系统中缺少相关开发库。

3.3.5.4 解决方案

安装相关工具：

```
$ sudo apt install qtdeclarative5-dev
```

3.3.6 MIPS 架构如何安装 electron

3.3.6.1 系统版本

适用系统：V10、V10（SP1）

适用架构：MIPS

具体请参考麒麟生态网站-文档帮助-《银河麒麟桌面操作系统 V10Electron 包开发者指南》

3.3.6.2 问题描述

在银河麒麟桌面操作系统 V10MIPS 架构上执行 npm install electron 报错，无法进行 electron 安装，那么如何在银河麒麟桌面操作系统 V10 架构上安装 electron 呢?

3.3.6.3 问题分析

npm 包都是从 npm 源中进行下载，但源中没有 MIPS 架构的 electron 包（或者说在 npmjs 这个网站上龙芯版本的 electron 不是这个包名）。实际包名：loongson-electron，版本有 4.1.3 6.1.7 10.1.0，使用 npm 下载时的命令为：

```
$ npm install loongson-electron@***
```

例如：$ npm install loongson-electron@6.1.7

建议安装 6.1.7 版本，测试发现 4.1.3 版本在 V10 系统存在键盘输入界面闪退的情况。

还有一个相关的包，包名：electron-mips，版本都是 10.1.0 以上的版本。

如果网络不好的情况下，会存在在线包失败的情况，我们可以通过离线安装的方式来安装 electron，下面就来介绍如何离线安装 electron。

注：

npm 官方网站：https://www/npmjs.com/

npm 官方源：https://registry.npm.com/binary.html

npm 淘宝源：https://registry.npmmirror.com/binary.html

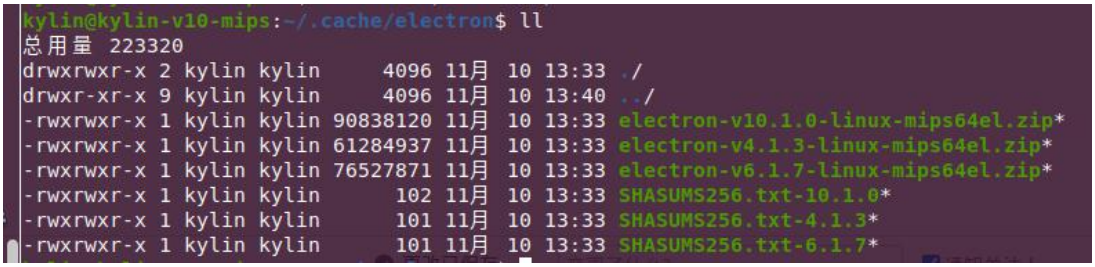
3.3.6.4 解决方案

1) 将离线安装包放在~/.cache/electron/目录下

首先需要手动创建~/.cache/electron/目录，执行如下命令：

```
$ mkdir -p ~/.cache/electron/
```

然后将离线包拷贝到上面的目录。



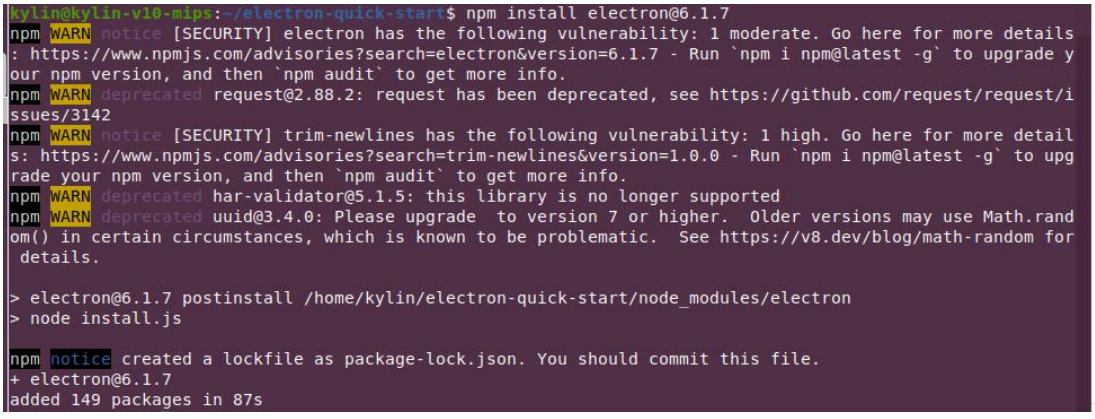
2) 克隆 electron-quick-start 工程项目（用此项目主要是为了验证 electron 是否安装成功）

```
$ git clone https://gitee.com/zzf35/electron-quick-start.git
$ cd electron-quick-start
$ git checkout remotes/origin/6-x-y
```

3) 安装 electron

在工程目录下（一般是 packages.json 文件的同级目录下）执行 npm install electron@6.1.7

```
$ cd ~/electron-quick-start
$ npm install electron@6.1.7
```



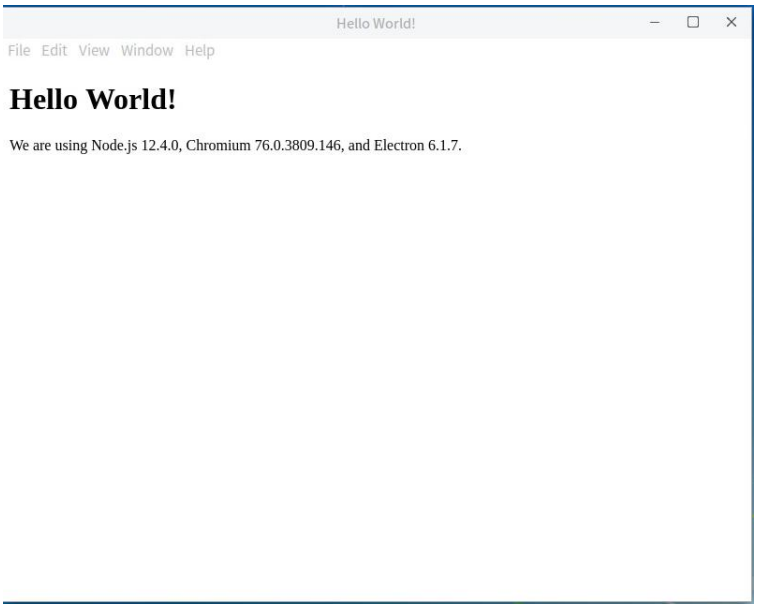
4) 测试 electron 是否正确安装

```
$ cd ~/electron-quick-start
$ npm start
```

electron6.1.7 版本执行此步骤的时候会有报错，按照报错提示进行修改：

```
$ sudo chown 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
```

执行之后，会在本地弹出页面窗口，并显示当前的 electron 版本。



表明 electron 已经正确安装。

3.3.7 系统同时存在 Qt5 和 Qt4，使用 qmake 命令行编译时如何指定 Qt 版本

3.3.7.1 系统版本

适用系统：V10、V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.3.7.2 问题描述

当系统中同时存在 Qt5 和 Qt4 时，直接使用 qmake 编译可能导致报错。

3.3.7.3 问题分析

当系统中同时存在 Qt5 和 Qt4 时，使用 qmake 编译时需要确认当前默认的 qmake 版本，如果不是想要使用的版本，需要通过配置文件来进行修改。

3.3.7.4 解决方案

1） 手动设置 qmake 版本可以从

/usr/lib/loongarch64-linux-gnu/qt-default/qtchooser/default.conf 配置

文件中修改：

```
$ vim /usr/lib/loongarch64-linux-gnu/qt-default/qtchooser/default.conf
```

将其中 qmake 的版本改成需要的版本。

2） 然后使用 qmake X.pro -o Makefile 自动生成 Makefile 文件：

```
$ qmake X.pro -o Makefile
```

3） 安装 libqt-dev 等开发工具包及 g++和 gcc 等编译器进行编译。

3.3.8 如何安装高版本 nodejs 开发环境

3.3.8.1 系统版本

适用系统：V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.3.8.2 解决方案

使用如下命令从系统源中直接安装

```
$ sudo apt update
$ sudo apt install node npm
```

系统源中仅维护了一个版本,其它版本可以通过 nodejs 官网下载 tar 包后通过指定环境变量的方式来进行使用，但维护不够方便。

推荐使用 nodejs 管理工具来维护 nodejs 环境

1）使用 n 模块安装 npm 和 node 方法

此种方法需要先从外网源中安装 npm，然后使用 npm 安装 n 模块，再使用 n 模块来管理 nodejs 环境。

```
$ sudo apt update
```

```
$ sudo apt install -y npm
$ npm config set registry https://registry.npm.taobao.org/
$ sudo npm i -g n
$ sudo n 16.17.1
$ sudo npm i -g pnpm
```

2）nvm 模块安装 npm 和 node 方法

此种方法无需从外网源中安装软件，安装 nvm 后直接使用 nvm 来管理 nodejs 环境。

nvm 模块通常需要联网进行安装，由于网络原因，可能会出现无法正常安装的情况，可以直接使用我们打好的 deb 包 nvm_0.39.3_all.deb 进行安装。

```
$ sudo   dpkg -i nvm_0.39.3_all.deb
```

#然后新开一个终端执行

```
$ nvm install 16.17.1
$ npm config set registry https://registry.npm.taobao.org/
$ npm i -g pnpm
```

3.3.9 如何搭建 Go 应用开发环境

3.3.9.1 系统版本

适用系统：V10(SP1)

适用架构：X86、ARM

其他版本和架构可作参考。

3.3.9.2 解决方案

麒麟外网源中下载安装：

麒麟系统外网源中有 go 环境，但版本相对较低，通过以下命令安装即可，默认安装的是 go1.13

```
$ sudo apt install golang
```

很多应用编译时需要高版本的 golang 版本，可以通过从官网中下载 tar 包进行安装。

参考：https://go.dev/doc/install

下载官方提供的 tar 包，官方仅提供了 amd64 和 arm64 架构的 tar 包，龙芯架构需要自行编译或去龙芯生态社区 http://www.loongnix.cn/zh/toolchain/Golang/获取

此处我们以在 x86_64 架构上安装 go1.20.4 为例

```
#下载
$ wget https://go.dev/dl/go1.20.4.linux-amd64.tar.gz
#解压
$ tar xvf go1.20.4.linux-amd64.tar.gz
#放到指定目录，此处我们以放到/usr/local/目录下为例
$ mv go /usr/local/
#添加环境变量
$ export PATH=$PATH:/usr/local/go/bin
#验证是否安装成功
$ go vesion
```

国内使用 go 拉取资源可能会存在问题，建议使用国内的 go 源【参考 3.3.10】

3.3.10 配置 go 加速镜像源

3.3.10.1 系统版本

使用系统：V10(SP1)

使用架构：全架构

3.3.10.2 问题描述

使用 go 进行编译程序时，拉取依赖会很慢，甚至拉取失败

3.3.10.3 问题分析

github 源下载比较慢，可以使用国内加速镜像源

3.3.10.4 解决方案

```
# 启用 Go Modules 功能
$ go env -w GO111MODULE=on

# 配置 GOPROXY 环境变量，以下三选一
```

```
# 1. 七牛 CDN
$ go env -w  GOPROXY=https://goproxy.cn,direct

# 2. 阿里云
$ go env -w GOPROXY=https://mirrors.aliyun.com/goproxy/,direct

# 3. 官方
$ go env -w  GOPROXY=https://goproxy.io,direct
```

查询

```
$ go env | grep GOPROXY GOPROXY="https://goproxy.cn"
```

参考：https://learnku.com/go/wikis/38122

3.3.11 如何搭建 Tauri 应用开发环境

3.3.11.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.3.11.2 解决方案

rustc 和 cargo 安装

系统源中可以安装 rustc 和 cargo，但版本不是最新的，编译时可能会有报错，建议使用脚本安装最新的 rustc 和 cargo

```
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
$ source $HOME/.cargo/env
```

cargo 拉取慢解决方法

进入~/.cargo/目录,创建 config 文件

```
[source.crates-io]
registry = "https://github.com/rust-lang/crates.io-index"

replace-with = 'tuna'
[source.tuna]
registry = "https://mirrors.tuna.tsinghua.edu.cn/git/crates.io-index.git"

#replace-with = 'ustc'
#[source.ustc]
```

```
#registry = "git://mirrors.ustc.edu.cn/crates.io-index"

[net]
git-fetch-with-cli = true
```

安装完成后，查看版本

```
$ cargo -V
cargo 1.69.0 (6e9a83356 2023-04-12)
$ rustc -V
rustc 1.69.0 (84c898d65 2023-04-16)
```

上面的安装方法只针对 x86_64 和 arm64 架构。

龙芯架构需要使用龙芯源中的版本。

http://www.loongnix.cn/zh/toolchain/Rust/

http://pkg.loongnix.cn:8080/loongnix/pool/main/r/rustc/

nodejs 和包管理工具安装

软件编译还需要 nodejs 和包管理工具，包管理工具一般有 npm、yarn、pnpm，编译不同程序时需要根据实际需要安装。

想要安装包管理器工具，一般都是先安装 npm 和 node，安装 npm 和 node 可以使用两种安装方式，即 n 模块安装或 nvm 模块安装

使用 n 模块安装 npm 和 node 方法

```
$ sudo apt update
$ sudo apt install -y npm
$ npm config set registry https://registry.npm.taobao.org/
$ sudo npm i -g n
$ sudo n 16.17.1
#根据实际需求安装，此处以安装 yarn 为例
$ sudo npm i -g yarn
```

nvm 模块安装 npm 和 node 方法

nvm 模块通常需要联网进行安装，由于网络原因，可能会出现无法正常安装的情况，可以直接使用我们打好的 deb 包 nvm_0.39.3_all.deb 进行安装。

```
$ sudo  dpkg -i nvm_0.39.3_all.deb
#然后新开一个终端执行
$ nvm install 16.17.1
$ npm config set registry https://registry.npm.taobao.org/
#根据实际需求安装，此处以安装 yarn 为例
$ npm i -g yarn
```

3.3.12 Pip 如何更换国内源

3.3.12.1 系统版本

适用系统：V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.3.12.2 问题描述

使用 pip 或 pip3 安装包时特别慢或者直接安装失败

3.3.12.3 问题分析

默认情况下 pip 使用的是国外的镜像源，在下载的时候速度非常慢，建议在使用 pip 安装包时切换为国内源

3.3.12.4 解决方案

可以直接在 pip 命令中使用 -i 参数来指定镜像地址，例如：

```
$ pip3 install numpy -i https://pypi.tuna.tsinghua.edu.cn/simple
```

以上命令使用清华镜像源安装 numpy 包。

这种只对当前安装的命令有用，如果需要全局修改，则需要修改配置文件。

Linux 环境中，配置文件位置在 ~/.config/pip/pip.conf（如果不存在创建该目录和文件）：

打开配置文件 ~/.config/pip/pip.conf，修改如下：

```
[global]
```

```
index-url = https://pypi.tuna.tsinghua.edu.cn/simple

[install]

trusted-host = https://pypi.tuna.tsinghua.edu.cn
```

查看 镜像地址：

```
$ pip3 config list

global.index-url='https://pypi.tuna.tsinghua.edu.cn/simple'

install.trusted-host='https://pypi.tuna.tsinghua.edu.cn'
```

或使用如下命令来进行配置

```
# 清华源
$ pip3 config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple

# 或：
# 阿里源
$ pip3 config set global.index-url https://mirrors.aliyun.com/pypi/simple/
# 腾讯源
$ pip3 config set global.index-url http://mirrors.cloud.tencent.com/pypi/simple
# 豆瓣源
$ pip3 config set global.index-url http://pypi.douban.com/simple/
# 百度源
$ pip3 config set global.index-url https://mirror.baidu.com/pypi/simple
```

注：

需要注意 pip 的 版本，使用 pip -V 确认 pip 的版本，确认是哪个 python 版本的 pip。

龙芯需要使用龙芯 pip 源

```
$ cat ~/.config/pip/pip.conf
[global]
timeout = 60
index-url = https://pypi.loongnix.cn/loongson/pypi
extra-index-url = https://pypi.tuna.tsinghua.edu.cn/simple
[install]
trusted-host =
```

```
pypi.loongnix.cn
pypi.tuna.tsinghua.edu.cn
```

参考：http://docs.loongnix.cn/python/python.html

3.3.13 hadoop 安装配置及启动参考文档

3.3.13.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.3.13.2 解决方案

hadoop 下载链接：

http://archive.apache.org/dist/hadoop/core

解压安装

```
$ sudo tar -zxvf   hadoop-3.3.0.tar.gz -C /usr/local
$ cd /usr/local
$ sudo mv   hadoop-3.3.0     hadoop
$ sudo chmod 777 -R hadoop
```

配置环境变量

```
$ vim ~/.bashrc
# hadoop
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_CLASS_PATH=$HADOOP_CONF_DIR
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
$ source ~/.bashrc
```

修改 jkd 路径并验证

```
$ cd /usr/local/hadoop/etc/hadoop
$ sudo vim hadoop-env.sh
$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
$ cd /usr/local/hadoop/bin
$ ./hadoop version
```

修改配置文件

```
$ sudo vim core-site.xml
<configuration>
```

```
    <property>
        <name>hadoop.tmp.dir</name>
        <value>file:/usr/local/hadoop/tmp</value>
        <description>Abase for other temporary directories.</description>
    </property>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>

$ sudo vim hdfs-site.xml
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/usr/local/hadoop/tmp/dfs/name</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/usr/local/hadoop/tmp/dfs/data</value>
    </property>
</configuration>
```

配置密钥

```
$ ssh-keygen -t rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 600 ~/.ssh/authorized_keys
$ chmod 700 ~/.ssh
```

执行 namenode 格式化并启动 namenode 和 datanode 进程

```
$ ./bin/hdfs namenode -format
$ ./sbin/start-dfs.sh
$ ./sbin/stop-dfs.sh
```

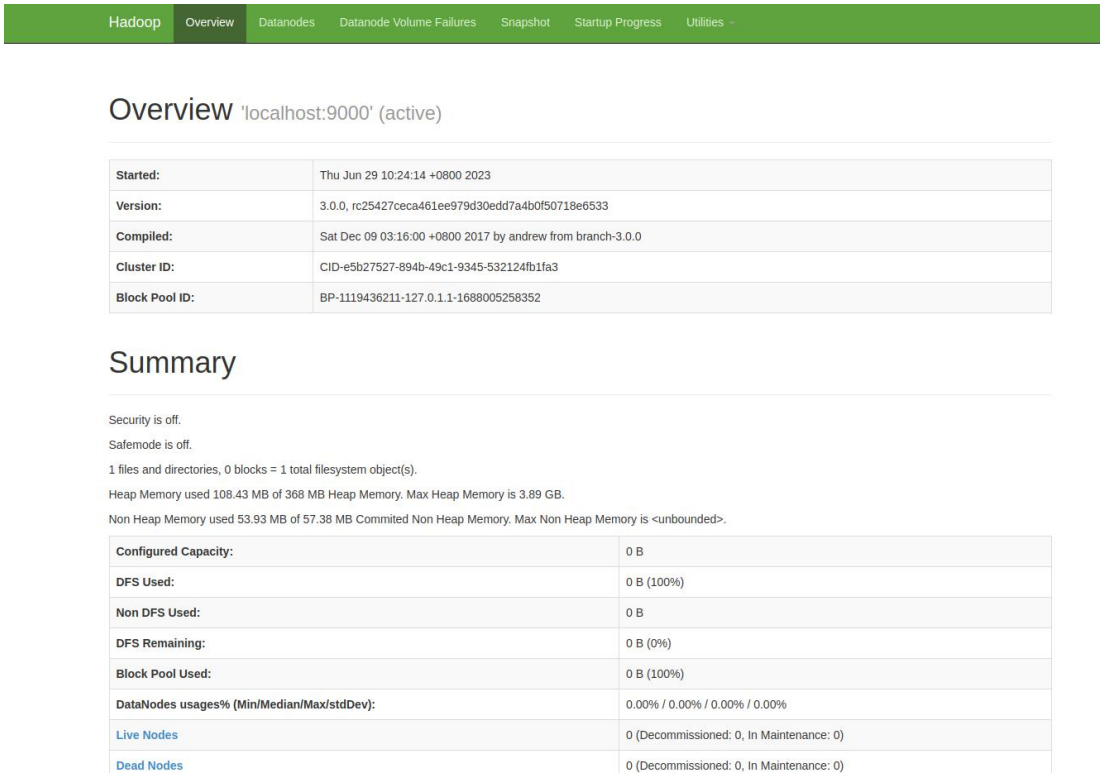
查看启动结果及 web 页面登录

查看节点已启动

```
$ jps
```

```
kylin@kylin-gw001m1aftf:/usr/local/hadoop$ jps
988637 DataNode
988524 NameNode
989067 Jps
988849 SecondaryNameNode
```

访问 http://localhost:9870



3.3.14 将 jar 包转换成可执行文件

3.3.14.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.3.14.2 解决方案

创建脚本

```
$ vim stub.sh
#!/bin/sh
MYSELF=`which "$0" 2>/dev/null`
[ $? -gt 0 -a -f "$0" ] && MYSELF="./$0"
java=java
```

```
if test -n "$JAVA_HOME"; then
    java="$JAVA_HOME/bin/java"
fi
exec "$java" $java_args -jar $MYSELF "$@"
exit 1
```

将 app.jar 转换成 app.run

```
$ cat stub.sh app.jar > app.run && chmod +x app.run
```

然后即可直接运行

```
$ ./app.run
```

运行 java 程序如果需要添加 VM 参数，可在 stub.sh 脚本第一行加上

```
java_args="-Xmx20m"
```

3.3.15 编译 Qt 后无法切换中文输入法

3.3.15.1 系统版本

适用系统：V10、V10（SP1）

适用架构：X86、ARM、Loongarch

其他版本和架构可作参考。

3.3.15.2 问题描述

在进行 Qt 程序打包时，部分应用会选择自包含方案，自己编译 Qt 使用，但打包后出现程序无法切换中文输入法导致无法输入中文的情况。

3.3.15.3 问题分析

银河麒麟桌面操作系统输入法框架采用的是 fcitx，而不是 ibus，Qt 默认只支持 ibus 输入法框架。在 Qt/5.15.2/gcc_64/plugins/platforminputcontexts/路径下可以看到，只有 libibusplatforminputcontextplugin.so，而没有 libfcitxplatforminputcontextplugin.so。解决该问题需要编译出 libfcitxplatforminputcontextplugin.so 动态库,使其支持 fcitx 输入法框架，从而支持输入中文。

3.3.15.4 解决方案

1）下载源码

```
$ git clone https://github.com/fcitx/fcitx-qt5
```

2）安装依赖

```
$ sudo apt install extra-cmake-modules
```

3）设置 Qt 环境变量

设置 Qt 的环境变量，使用自己编译的 Qt 即可，例如

```
export QT_HOME=/home/kylin/Qt5.15.12
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${QT_HOME}/lib
export QT_PLUGIN_PATH=$QT_PLUGIN_PATH:${QT_HOME}/plugins
export QML2_IMPORT_PATH=$QML2_IMPORT_PATH:${QT_HOME}/qml
export PATH=${QT_HOME}/bin:$PATH
```

4）Qt6 修改配置文件

如果编译的是 Qt6，需要修改 fcitx-qt5 的 CMakeLists.txt 文件，将 ENABLE_QT5 改为 off，ENABLE_QT6 改为 on

```
option(ENABLE_QT4 "Enable Qt 4" On)
option(ENABLE_QT5 "Enable Qt 5" Off)
option(ENABLE_QT6 "Enable Qt 6" On)
option(BUILD_ONLY_PLUGIN "Build only plugin" Off)
option(BUILD_STATIC_PLUGIN "Build plugin as static" Off)
option(WITH_FCITX_PLUGIN_NAME "Enable plugin name with fcitx" On)
```

5）编译

```
$ mkdir build
$ cd build
$ cmake .. -DENABLE_LIBRARY=false
$ make
```

编译完成的 libfcitx5platforminputcontextplugin.so 插件动态库在 build 目录下的 qt5/platforminputcontext 文件夹内。

3.4 系统设置问题

3.4.1 系统如何通过 interfaces 配置文件修改系统 IPv4 和 IPv6

3.4.1.1 系统版本

适用系统：V10

适用架构：X86、ARM、MIPS

其他版本和架构可作参考。

3.4.1.2 问题描述

V10 系统如何通过 interfaces 设置文件设置系统 IPv4 和 IPv6 地址？

3.4.1.3 问题分析

有些系统可能不方便从桌面来配置网络，需要通过配置文件来配置网络。

3.4.1.4 解决方案

1) 首先需要确认网卡名称

使用 ip a 或 ip link 查看：

```
$ ip a
```

或

```
$ ip link
```

```
kylin@kylin-v10-mips-2107:~/桌面$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:e1:4a:3d brd ff:ff:ff:ff:ff:ff
    inet [redacted] brd [redacted] scope global dynamic ens33
       valid_lft 1184sec preferred_lft 1184sec
    inet6 fe80::5434:e4fb:e7b5:e386/64 scope link
       valid_lft forever preferred_lft forever
kylin@kylin-v10-mips-2107:~/桌面$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:e1:4a:3d brd ff:ff:ff:ff:ff:ff
kylin@kylin-v10-mips-2107:~/桌面$
```

2) 修改 IP

```
$ sudo vim /etc/network/interfaces
```

添加以下内容

```
auto ens33
iface ens33 inet static
address XXX.XXX.XXX.XXX
netmask XXX.XXX.XXX.XXX
gateway XXX.XXX.XXX.XXX
dns-nameservers 114.114.114.114
dns-nameservers XXX.XXX.XXX.XXX
iface ens33 inet6 static
address 2a04:f80:0754:168:225:218:171:0/112
gateway 2a04:f80:0754::1
```

```
dns-nameservers 2001:4860:4860::8888
dns-nameservers 2001:4860:4860::8844
```

其中，XXX.XXX.XXX.XXX 根据实际情况填写。

3) 重启网络

```
$ sudo /etc/init.d/networking restart
```

在这里可能会出现两个**错误**：

a. failed to start raise network interfaces，提示可以通过 journalctl -xe 命令查看错误日志。

解决办法：

将 IP 地址修改的内容中 auto eth0 改为 allow-hotplug eth0

修改之后重新运行重启命令，修改成功。

b. Ubuntu Server: job is already running networking

在修改 Ubuntu Server 的网络配置后，重启网络

（命令:sudo /etc/init.d/networking restart）时出现错误:

stop: Job failed while stopping

start: Job is already running: networking

解决方法：

跟踪一下网络设置的日志：

命令：

```
$ sudo tail -f /var/log/upstart/networking.log
```

可以看到日志的内容：

```
Stoping or restarting the networking job is not supported.
    Use ifdown & ifup to reconfigure desired interface.
```

也就是说不支持 Stop 和 Restart 这样的方式来重启网络，使用 ifdown 和 ifup 来重启，命令如下：

```
$ sudo ifdown eth0 && sudo ifup eth0
```

3.4.2 如何通过修改 grub 文件来调整虚拟机分辨率

3.4.2.1 系统版本

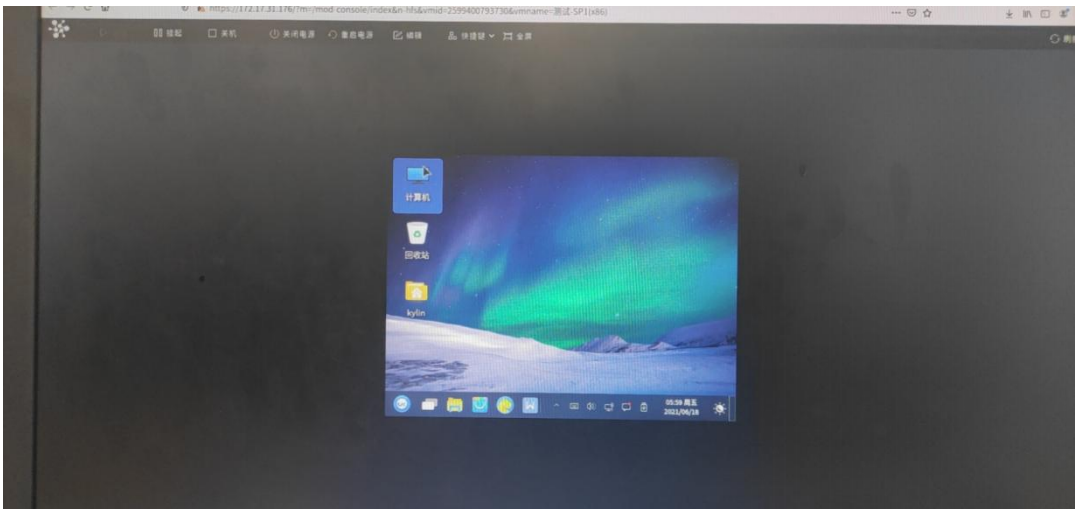
适用系统：V10、V10(SP1)

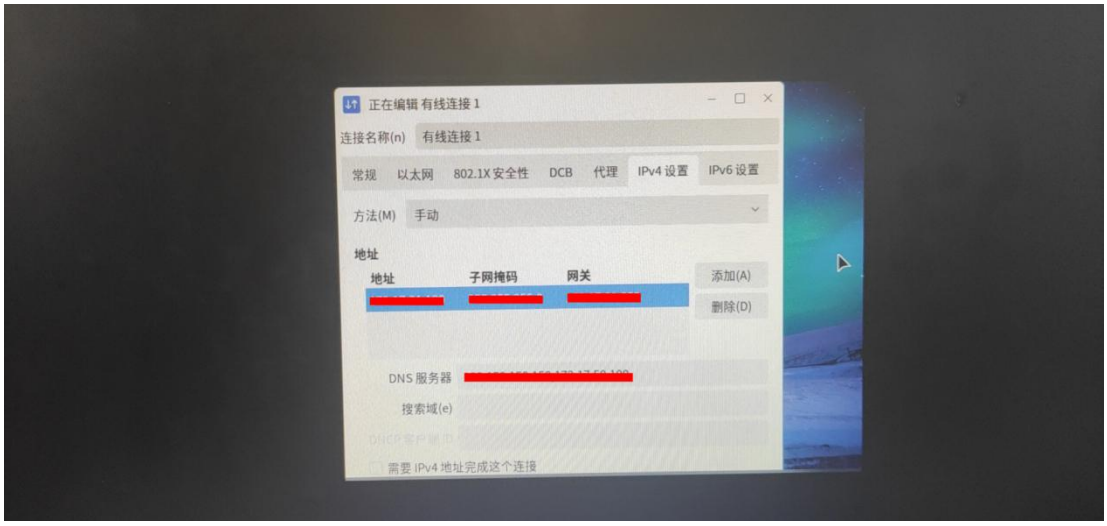
适用架构：X86、ARM、MIPS

其他版本和架构可作参考。

3.4.2.2 问题描述

在云桌面安装虚拟机后，分辨率过小且无法调整，导致系统部分功能无法使用。（如下例：因分辨率太低，网络设置窗口的保存按钮无法点击）配置 xrandr 和 cvt 均无效。





3.4.2.3 问题分析

可以从 grub 文件中进行修改分辨率。

3.4.2.4 解决方案

修改 grub 文件：

1) 在虚拟机系统中打开 grub 文件

```
$ vim /etc/default/grub
```

2) 搜索#GRUB_GFXMODE=640x480

3) 编辑：640x480 改成你想要的分辨率，并取消前面的#

例如：GRUB_GFXMODE=1024x768

4) 更新：

```
$ sudo update-grub
```

5) 最后重启虚拟机

3.4.3 系统如何进入单用户模式

3.4.3.1 系统版本

适用系统：V10（SP1）

适用架构：X86、ARM

其他版本和架构可作参考。

3.4.3.2 问题描述

麒麟桌面系统如何进入单用户模式，对系统文件进行修改？

3.4.3.3 问题分析

麒麟操作系统提供了单用户模式（类似 Windows 安全模式），可以在最小环境中进行系统维护。在单用户模式（运行级别 1）中，Linux 引导进入根 shell，网络被禁用，只有少数进程运行。单用户模式可以用来修改文件系统损坏、还原配置文件、移动用户数据等。

麒麟系统遇到以下情况，这时候需修改某些文件让系统正常启动，如果直接进入 recovery 模式，默认是文件权限只读，无法修改文件。这时我们需要进入 recovery 的单用户模式，获得修改文件的权限。

1) 机器卡死，hang 住；

2) 一直重启无法进系统修改和配置；

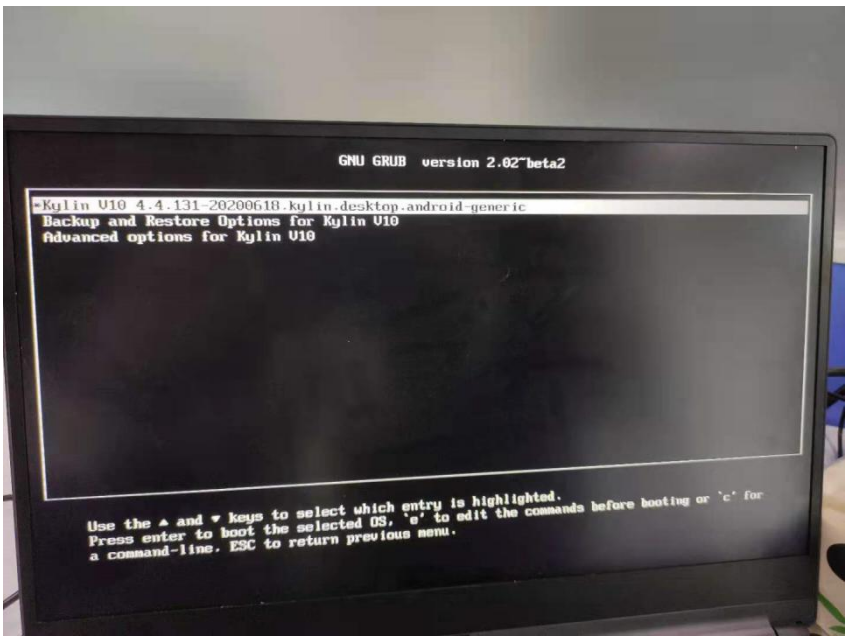
3) 密码忘记需要修改；

4) 文件系统被损坏；

等等。

3.4.3.4 解决方案

1) 启动系统到以下图片内容的时候按 e 键，进入 grub 菜单：



2) 修改里面的 linux 行，在最后面加上“single”；

3) 按 ctrl+x 或 F10 保存后进入单用户模式；

4) 根据以上场景进行实际修改。

3.4.4 如何禁用系统 IPv6

3.4.4.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.4.4.2 问题描述

如何禁用系统 IPV6？

3.4.4.3 问题分析

通过修改 grub 文件。

3.4.4.4 解决方案

修改 grub。

1) 通过配置 grub 配置文件禁用：

```
$ sudo vim /etc/default/grub
```

2) 查找包含"GRUB_CMDLINE_LINUX"的行，并如下编辑：

```
GRUB_CMDLINE_LINUX="ipv6.disable=1"
```

然后，按下“ESC”，再输入“:wq”保存并关闭文件。

3) 重新生成 grub 文件

```
$ sudo update-grub2
```

4) 重启电脑

现在 IPV6 就禁用了。

3.4.5 系统安装应用提示：“检测到不合法来源应用试图安装，是否允许”

3.4.5.1 系统版本

适用系统：V10(SP1)

适用架构：X86、ARM

其他版本和架构可作参考。

3.4.5.2 问题描述

系统安装应用的时候会反复提示“安全授权认证”，如何才能取消呢?

3.4.5.3 问题分析

V10(SP1) 6 月之后的版本增加了“麒麟安全授权认证”，此程序会检查应用程序的来源，阻止未知来源的应用程序安装；当应用没有经过麒麟签名认证就会有此警告提示；

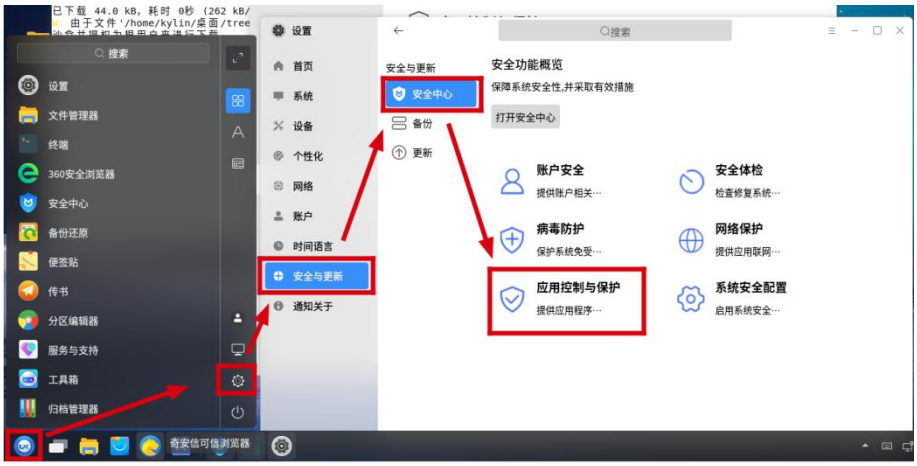
建议从麒麟软件商店进行安装已经签名的软件或通过麒麟开发者平台申请证书后对应签名再进行安装；（推荐）

系统默认处于“警告”级别状态，检测到未知来源的应用就会提示用户进行授权，如果只是临时进行软件测试可以通过调整“检测应用程序来源”的级别状态为“关闭”来去除反复提示的“安全授权认证”。

3.4.5.4 解决方案

修改“检测应用程序来源”级别状态。

1) 点击“开始”-“设置”-“安全与更新”-“安全中心”，选择“应用控制与保护”



2) 修改为“关闭”

3.4.6 V10 系统如何在菜单栏添加目录

3.4.6.1 系统版本

适用系统：V10

适用架构：X86

其他版本和架构可作参考。

3.4.6.2 问题描述

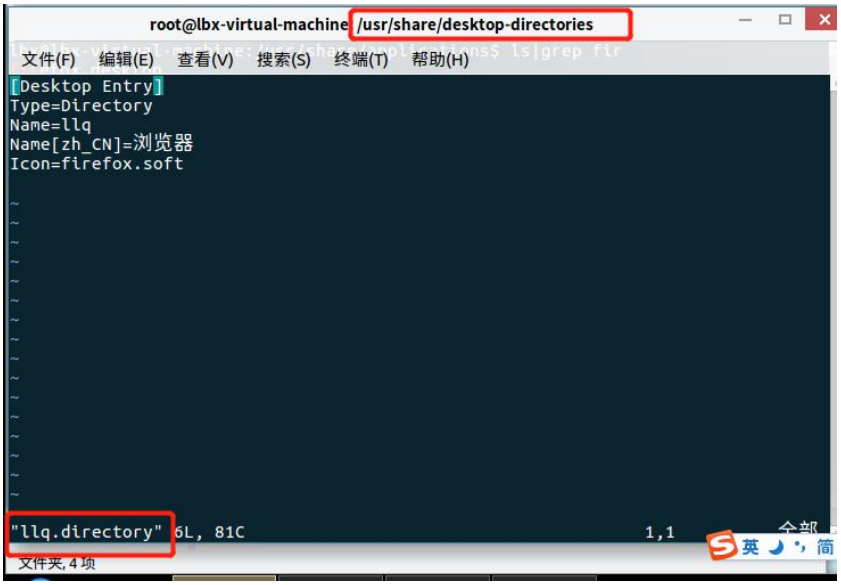
V10 系统如何在菜单栏添加目录?

3.4.6.3 问题分析

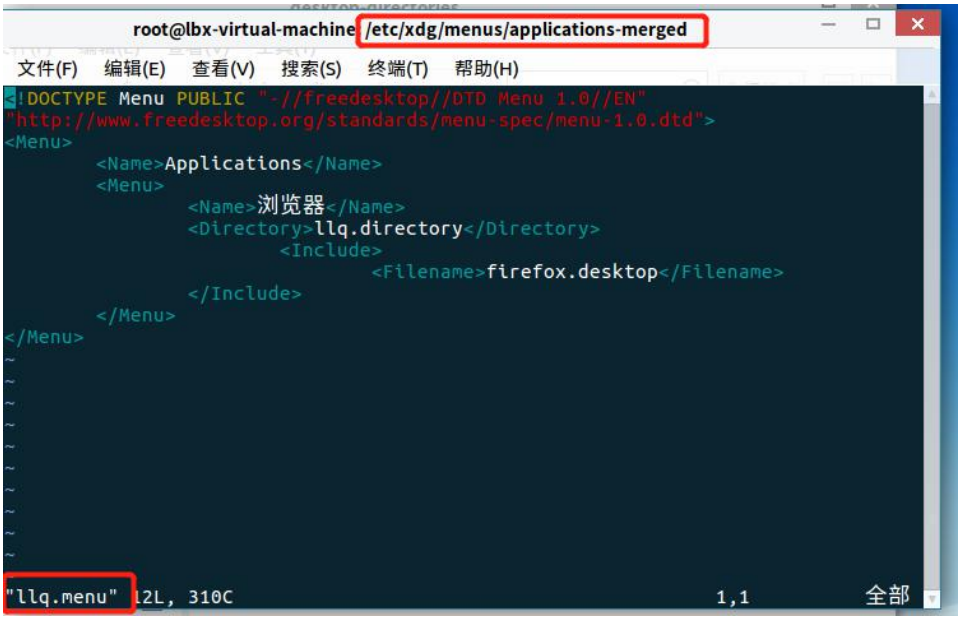
主要是修改配置文件。

3.4.6.4 解决方案

1) 首先在/usr/share/desktop-directories 目录下，创建后缀名为：“.directory”的文件，文件名在第二步会用到，文件格式如下图所示：



2) 然后在/etc/xdg/menus/applications-merged 目录下（注意，如果没有 applications-merged 这个路径，必须利用 mkdir 命令创建此目录），创建后缀名为：“.menu”的文件，其中文件格式如下图所示：



3) 将以上文件修改后，保存退出，即可在菜单栏看到如下效果：

3.4.7 如何设置 Swappiness 的值

3.4.7.1 系统版本

适用系统：V10（SP1）

适用架构：全架构

其他版本和架构可作参考。

3.4.7.2 问题描述

麒麟系统如何设置 Swappiness 的值?

3.4.7.3 问题分析

Swappiness 参数决定了 Swap 分区如何使用，当 Swappiness = 0 时，表示最大限度使用物理内存，然后才使用 Swap，当 Swappiness = 100 时，表示最大限度使用 Swap，然后才使用物理内存，这个值的初始值是 60，用 Swap 比较多，性能会差些，最好改成 10，尽量先用物理内存。

3.4.7.4 解决方案

1) 查看 Swappiness

```
$ cat /proc/sys/vm/swappiness
```

2) 更改 Swappines

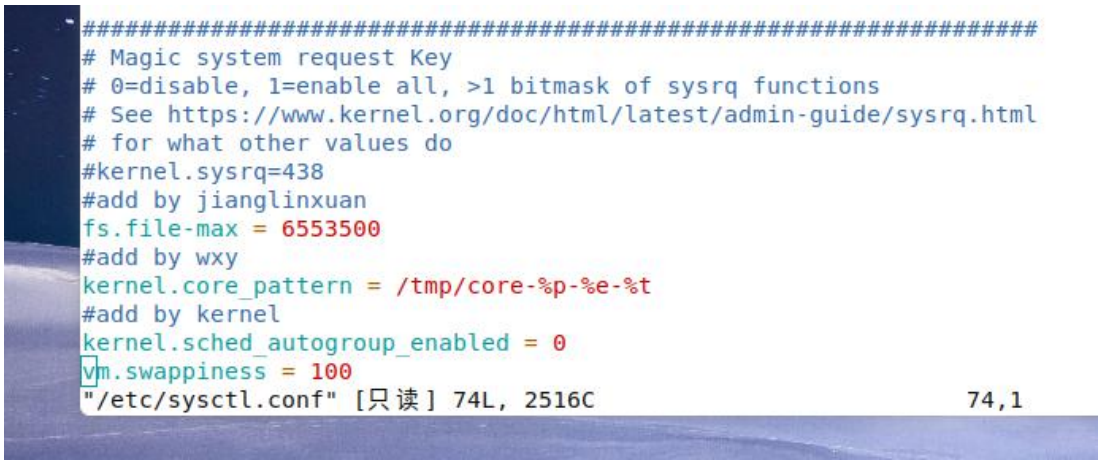
```
$ sudo sysctl vm.swappiness=10
```

3) 使开机生效

```
$ sudo vi /etc/sysctl.conf
```

4) 在最后一行加上

```
vm.swappiness = 10
```



3.4.8 如何修改网卡名称

3.4.8.1 系统版本

适用系统：V10（SP1）

适用架构：X86、ARM

其他版本和架构可作参考。

3.4.8.2 问题描述

麒麟系统如何修改网卡名称?

3.4.8.3 问题分析

可以通过修改/etc/default 目录下的 grub 文件和/etc/network 目录下的 interfaces 文件。

3.4.8.4 解决方案

可见如下将网卡名称 enp2s0 修改为 eth0。

1) 首先，查看原来的网卡名称

```
$ ifconfig
```

```
kylin@kylin-pc:~/桌面$ ifconfig
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet [redacted]  netmask [redacted]  broadcast [redacted]
        inet6 fe80::a09e:3f8:e6cd:1482  prefixlen 64  scopeid 0x20<link>
        ether 4e:56:2b:f6:ba:6b  txqueuelen 1000  (Ethernet)
        RX packets 218063  bytes 78383873 (78.3 MB)
        RX errors 0  dropped 3672  overruns 0  frame 0
        TX packets 15842  bytes 1242839 (1.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet [redacted]  netmask [redacted]
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2694  bytes 454949 (454.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2694  bytes 454949 (454.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

可见，原网卡名为“enp2s0”

2) 修改 grub 文件

```
$ sudo vim /etc/default/grub
```

在“GRUB_CMDLINE_LINUX”位置处不改变之前的原有信息，在其基础上追加信息“net.ifnames=0 biosdevname=0”

然后，按下 Esc，再输入:wq 保存退出。

```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_DISTRIBUTOR_RELEASE=`lsb_release -d -s | awk -F" " '{print $2 " " $3}' 2> /
dev/null || echo ""`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX="net.ifnames=0 biosdevname=0"
GRUB_CMDLINE_LINUX_SECURITY="audit=0 security=kysec"

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console
```

3) 更新配置

```
$ sudo update-grub
```

```
kylin@kylin-pc:~$ sudo update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
正在生成 grub 配置文件 ...
找到主题: /usr/share/grub/themes/UKUI/theme.txt
找到 Linux 镜像: /boot/vmlinuz-5.4.18-55-generic
找到 initrd 镜像: /boot/initrd.img-5.4.18-55-generic
找到 initrd 镜像: /boot/initrd.img-5.4.18-55-generic
完成
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
正在生成 grub 配置文件 ...
找到主题: /usr/share/grub/themes/UKUI/theme.txt
找到 Linux 镜像: /boot/vmlinuz-5.4.18-55-generic
找到 initrd 镜像: /boot/initrd.img-5.4.18-55-generic
找到 initrd 镜像: /boot/initrd.img-5.4.18-55-generic
完成
```

4) 编辑网络接口文件

```
$ sudo vim /etc/network/interfaces
```

在文中添加两行信息

```
auto eth0
iface eth0 inet dhcp
```

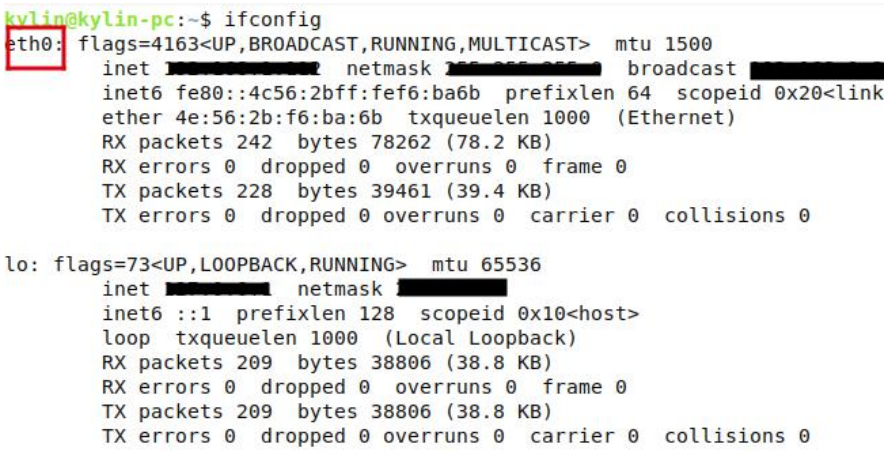
其中，**eth0** 为修改后的网卡名称。

然后，按下 Esc，再输入:wq 保存退出。

```
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
auto eth0
iface eth0 inet dhcp
~
```

5) 终端输入 reboot 重启系统，并查看网卡名称是否修改

```
$ reboot
$ ifconfig
```



3.5 应用安装卸载运行问题

3.5.1 npm 包下载慢的问题

3.5.1.1 系统版本

适用系统：V10、V10（SP1）

适用架构：全架构

其他版本和架构可作参考。

3.5.1.2 问题描述

NPM 是随同 NodeJS 一起安装的包管理工具，经常使用它来下载第三方包到本地。但在使用 NPM 过程很多人都知道，在国内下载第三方包的速度极其之慢。

3.5.1.3 问题分析

可以切换 npm 官方源为淘宝源。

3.5.1.4 解决方案

推荐使用淘宝 NPM 镜像，它是一个完整 npmjs.org 镜像，也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。

淘宝 NPM 镜像地址：https://npmmirror.com/

1)

```
$ npm install webpack-cli --registry=https://registry.npmmirror.com
```

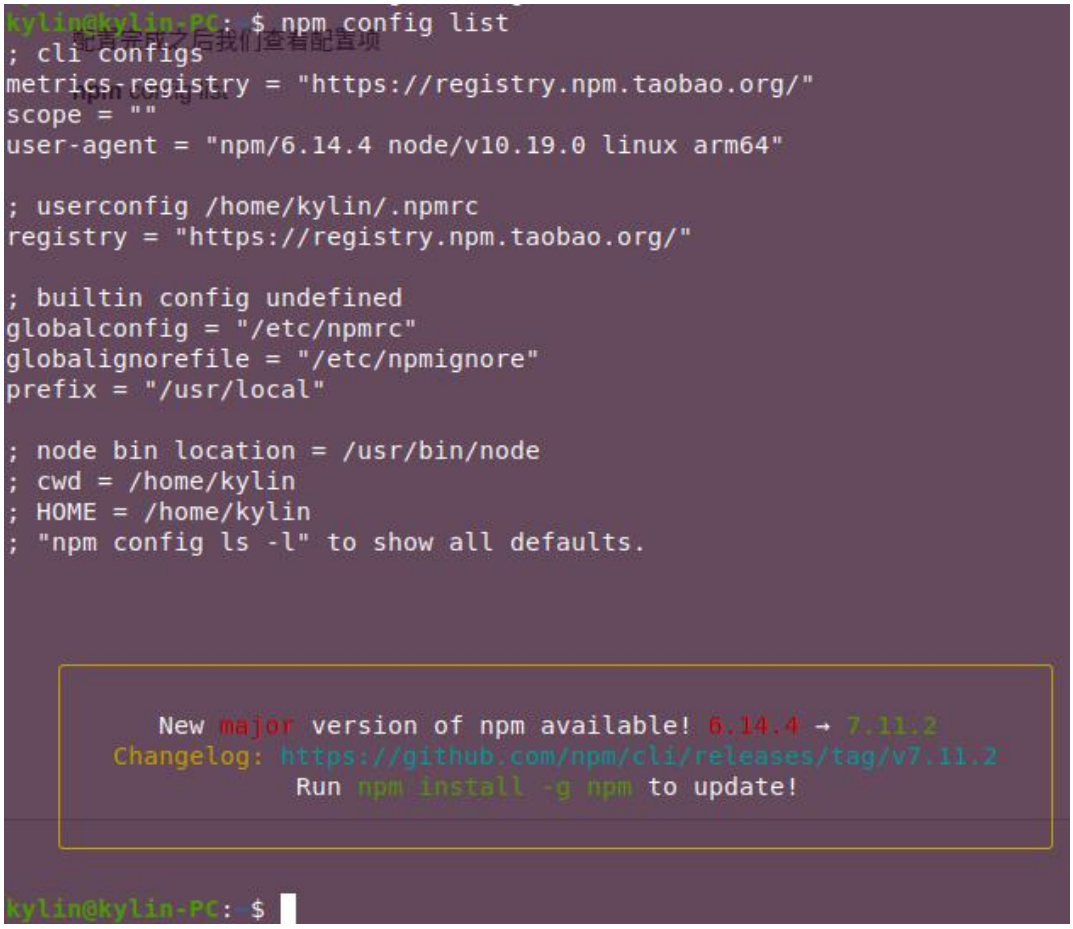
如果觉得每次都需要添加淘宝的网址麻烦的话，可以把它加入配置文件中。

2) 将镜像源加入配置文件

```
$ npm config set registry https://registry.npmmirror.com
```

3) 配置完成之后我们查看配置项

```
$ npm config list
```



4) 同时也可以使用淘宝 NPM 镜像定制的 cnpm (gzip 压缩支持) 命令行工具代替默认的 npm:

```
$ npm install -g cnpm --registry=https://registry.npmmirror.com
```

5) 这样就可以使用 cnpm 命令来安装模块了：

```
$ cnpm install [name]
```

3.5.2 V10 系统双击 deb 包无法安装应用

3.5.2.1 系统版本

适用系统：V10-0710 之前的版本

适用架构：X86、ARM、MIPS

其他版本或架构可做参考。

3.5.2.2 问题描述

V10-0710 之前的版本双击 deb 包无法安装应用

3.5.2.3 问题分析

双击安装 deb 包主要是麒麟包管理器进行处理，V10-20200710 之前的版本默认没有预装麒麟包管理器，需要从源里下载安装。

3.5.2.4 解决方案

安装麒麟包管理器：kylin-installer

安装 kylin-installer 的方法：

```
$ sudo apt update
$ sudo apt install kylin-installer
$ sudo apt install libqt5-ukui-style1
```

3.5.3 使用 apt 命令安装应用报错：Sub-process /usr/bin/dpkg returned an error code(1)

3.5.3.1 系统版本

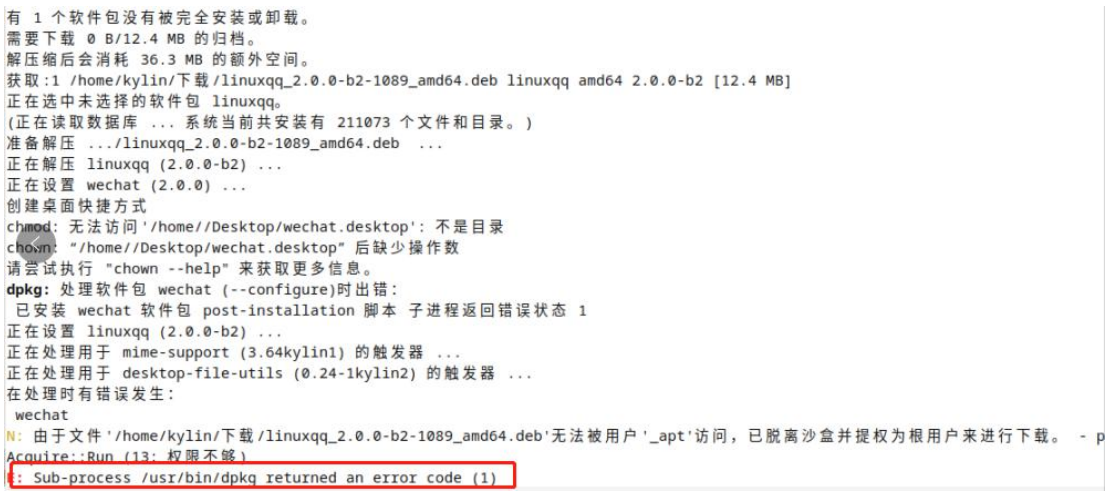
适用系统：V10、V10（SP1）

适用架构：全架构

其他版本或架构可做参考。

3.5.3.2 问题描述

使用 apt 安装源中的包时，如果受到网络影响造成安装失败，再次使用 apt 可能出现 Sub-process /usr/bin/dpkg returned an error code (1)错误。



3.5.3.3 问题分析

需要更新源恢复 apt 命令。

3.5.3.4 解决方案

1)

```
$ cd /var/lib/dpkg/
```

2) 改名做备份

```
$ sudo mv info/ info_bak
```

3) 新建 info 目录

```
$ sudo mkdir info
```

4) 更新源

```
$ sudo apt update
```

5) 修复源

```
$ sudo apt -f install
```

6) 执行完上一步操作后会在新的 Info 文件夹下生成一些文件，现将这些文件全部移动到 info_bak 文件夹下

```
$ sudo mv info/* info_bak/
```

7) 把自己新建的 info 目录删掉

```
$ sudo rm -rf info
```

8) 把备份的 info 目录名改回

```
$ sudo mv info_bak info
```

3.5.4 安装应用报错-“dpkg-deb：错误粘贴子进程被信号（断开的管道）终止了”

3.5.4.1 系统版本

适用系统：V10（SP1）

适用架构：X86

其他版本或架构可做参考。

3.5.4.2 问题描述

使用 apt 或 dpkg 安应用时报错：dpkg-deb: 错误: 粘贴子进程被信号(断开的管道) 终止了。

```
wyx@wyx-QiTianM435-N000:~/tmp$ sudo apt install ./com.iflytek.iflyime_2.0.15_amd64.deb
[sudo] wyx 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
注意，选中 'com.iflytek.iflyime' 而非 './com.iflytek.iflyime_2.0.15_amd64.deb'
下列软件包是自动安装的并且现在不需要了：
  libgsound0
使用'sudo apt autoremove'来卸载它(它们)。
下列【新】软件包将被安装：
  com.iflytek.iflyime
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 2 个软件包未被升级。
需要下载 0 B/35.9 MB 的归档。
解压缩后会消耗 168 MB 的额外空间。
(正在读取数据库 ... 系统当前共安装有 245006 个文件和目录。)
准备解压 .../com.iflytek.iflyime_2.0.15_amd64.deb  ...
正在解压 com.iflytek.iflyime (2.0.15) ...
dpkg: 处理归档 /var/cache/apt/archives/com.iflytek.iflyime_2.0.15_amd64.deb (--unpack)时出错：
 正试图覆盖 /usr/lib/x86_64-linux-gnu/libcares.so.2，它同时被包含于软件包 libc-ares2:amd64 1.15.0-1build
1
dpkg-deb: 错误：粘贴 子进程被信号(断开的管道) 终止了
在处理时有错误发生：
 /var/cache/apt/archives/com.iflytek.iflyime_2.0.15_amd64.deb
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

3.5.4.3 问题分析

从报错提示信息“正试图覆盖 /usr/lib/x86_64-linux-gnu/libcares.so.2，它同时被包含于软件包 libc-ares2:amd64 1.15.0-1build1”，基本可以看出应该是应用安装时和系统的库文件存在冲突；

此问题一般是由于应用打包不规范，将库文件直接放置在了系统库目录（/usr/lib/x86_64-linux-gnu），当依赖库未在系统安装时不会出现此问题，当系统已经安装此依赖库就会有如上报错。

3.5.4.4 解决方案

应用打包时不能将库文件直接放置在系统库文件目录下，需要通过 control 文件来增加依赖。

3.5.5 卸载时报错：软件包***需要重新安装，但无法找到相应的安装文件

3.5.5.1 系统版本

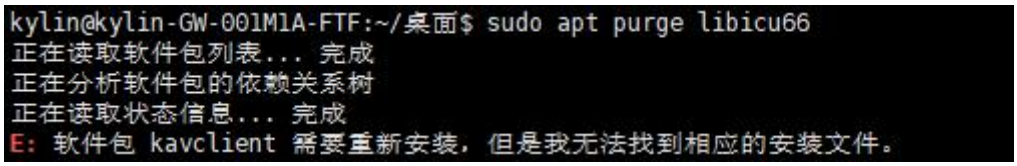
适用系统：V10、V10（SP1）

适用架构：全架构

其他版本或架构可做参考。

3.5.5.2 问题描述

卸载某些应用时报错：E：软件包***需要重新安装，但无法找到相应的安装文件；当出现此问题时，由于***应用无法卸载，会导致 apt 源不能使用的问题，包括使用 apt 源进行更新、安装、卸载、修复都无法正常使用。



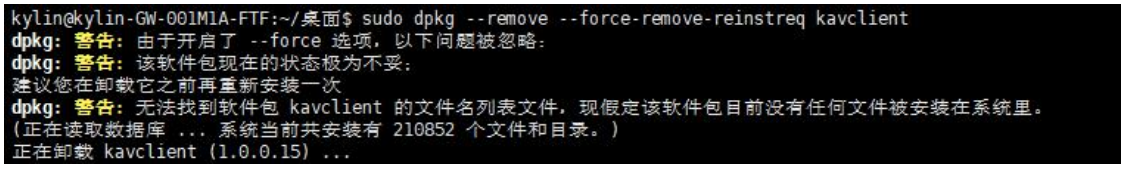
3.5.5.3 问题分析

某些应用没有正常卸载，导致 dpkg 数据库异常，需要将出现问题的应用彻底删除。

3.5.5.4 解决方案

1) 可以用以下命令将出现问题的应用彻底删除。

```
$ sudo rm -rf /var/lib/dpkg/info/kavclient.*
$ sudo dpkg --remove --force-remove-reinstreq kavclient
```



2) 之后就可以正常使用 apt 源进行安装、卸载、更新的操作了。

```
kylin@kylin-GW-001M1A-FTF:~/桌面$ sudo apt install libicu57
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列软件包是自动安装的并且现在不需要了:
  libyaml-cpp0.6 localechooser-data user-setup
使用'sudo apt autoremove'来卸载它(它们)。
下列【新】软件包将被安装:
  libicu57
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 74 个软件包未被升级。
需要下载 7,596 kB 的归档。
解压缩后会消耗 30.8 MB 的额外空间。
获取:1 http://archive.kylinos.cn/kylin/KYLIN-ALL 10.1/main arm64 libicu57 arm64 57.1-6kylin0.3 [7,596 kB]
获取:1 http://archive.kylinos.cn/kylin/KYLIN-ALL 10.1/main arm64 libicu57 arm64 57.1-6kylin0.3 [7,596 kB]
已下载 6,129 kB，耗时 1分 36秒 (64.1 kB/s)
正在选中未选择的软件包 libicu57:arm64。
(正在读取数据库 ... 系统当前共安装有 210852 个文件和目录。)
准备解压 .../libicu57_57.1-6kylin0.3_arm64.deb  ...
正在解压 libicu57:arm64 (57.1-6kylin0.3) ...
正在设置 libicu57:arm64 (57.1-6kylin0.3) ...
正在处理用于 libc-bin (2.31-0kylin9.1k20.3) 的触发器 ...
```

3.5.6 使用 yarn install 时报错：00h00m00s 0/0::ERROR:[Errno 2] No such fileor directory:’install’.

3.5.6.1 系统版本

适用系统：V10（SP1）

适用架构：X86

其他版本或架构可做参考。

3.5.6.2 问题描述

使用 yarn install 时报错：00h00m00s 0/0::ERROR:[Errno 2] No such fileor directory:’install’。

3.5.6.3 问题分析

系统在带的 yarn 有问题，需要通过 npm 的方式从全局安装 yarn。

3.5.6.4 解决方案

执行以下命令:

```
$ sudo apt remove cmdtest
$ sudo apt remove yarn
$ sudo npm install -g yarn
```

执行上述命令后需要重启终端再输入：

```
$ yarn install
```

3.5.7 应用上架后应用商店搜索不到的排查流程

3.5.7.1 系统版本

适用系统：V10（SP1）

适用架构：全架构

其他版本或架构可做参考。

3.5.7.2 问题描述

应用上架银河麒麟桌面操作系统应用商店后搜索不到。

3.5.7.3 问题分析

软件商店的问题较复杂，上架后在商店搜索不到的情况下，可以参考以下解决方法和排查步骤。

3.5.7.4 解决方案

一、可通过更新源和重启软件商店解决大部分问题

终端输入以下命令：

1） 清空本地缓存

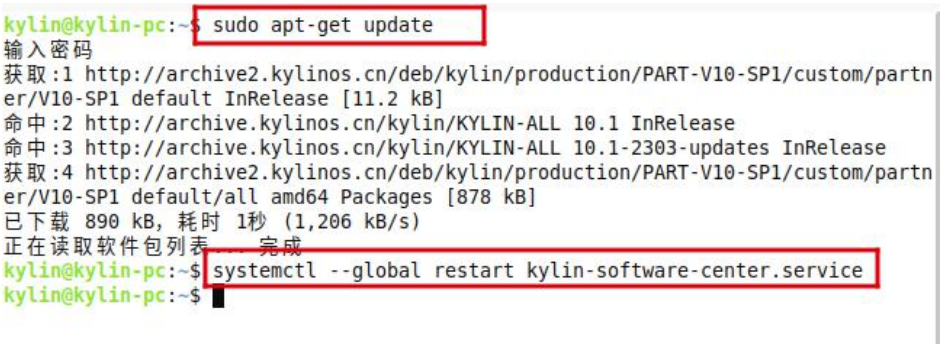
```
$ sudo rm -rf /var/lib/apt/lists/*
```

2） 更新源

```
$ sudo apt-get update
```

3） 重启软件商店

```
$ systemctl --global restart kylin-software-center.service
```

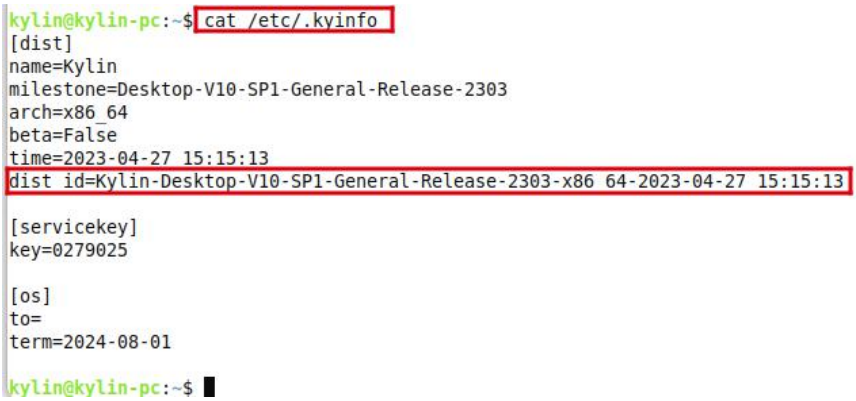


二、如果通过以上方法无法解决，可以参考下面的步骤进行排查出错环节

1）查询系统镜像版本

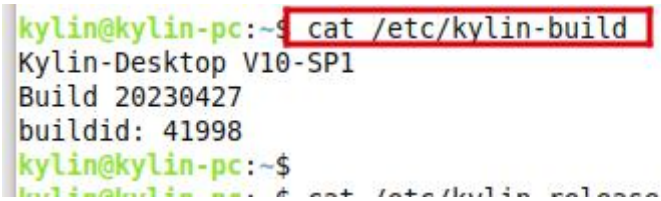
排查使用的系统镜像是不是出库版本，以及软件是否在该系统版本进行上架。

```
$ cat /etc/.kyinfo
```



注：图例所示的系统版本是【桌面 V10-SP1-release-2303-X86】。

如果用以上命令查不到系统镜像版本，可以使用’cat /etc/kylin-build’命令进行查询。



注：图例所示的系统版本是【桌面 V10-SP1】，系统镜像小版本号是【20230427】。

2）查询源地址

排查源地址是否正确，源不正确的情况下无法同步及更新。

```
$ cat /etc/apt/sources.list
```

注：系统对应的商业源地址如下，以下两个源地址对应其一即可（区别：上面的及时生效，下面的有一定延迟）

①V10 系统商业源地址：

```
deb https://archive2.kylinos.cn/DEB/KYLIN_DEB V10 main all
deb
```

```
http://archive2.kylinos.cn/deb/kylin/production/PART-V10/custom/partner/V10 default all
```

②V10SP1 系统商业源地址：

```
deb https://archive2.kylinos.cn/DEB/KYLIN_DEB V10-SP1 main all
deb http://archive2.kylinos.cn/deb/kylin/production/PART-V10-SP1/custom/partner/V10-SP1 default all
```

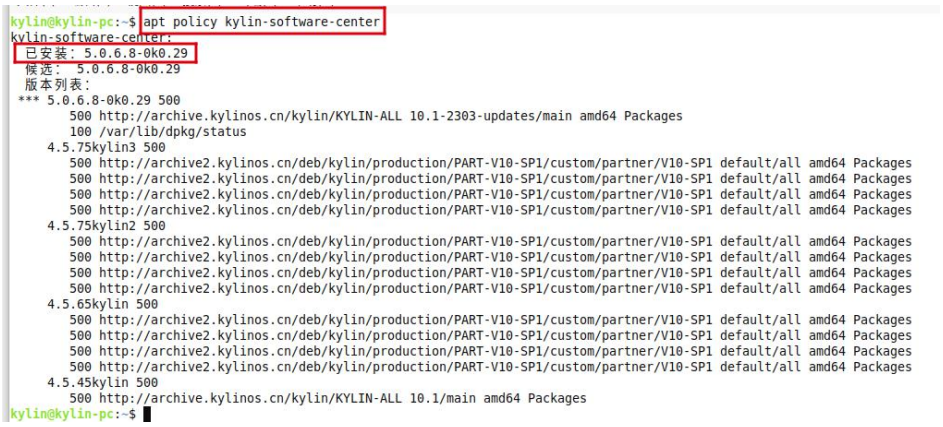
③海思系统商业源地址：

```
deb https://archive2.kylinos.cn/DEB/KYLIN_DEB 9a0 main all
deb https://archive2.kylinos.cn/deb/kylin/production/PART-10_1-kirin9a0/custom/partner/10_1-kirin9a0 default all
```

3）查询软件商店版本号

排查软件商店是否安装最新版本。

```
$ apt policy kylin-software-center
```



注：图例所示安装版本已是最新版本。

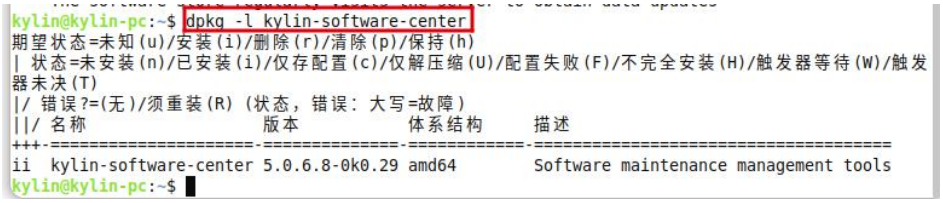
如果不是最新版本需将软件商店更新至最新版本。

更新软件商店命令：

```
$ sudo apt-get install kylin-software-center
```

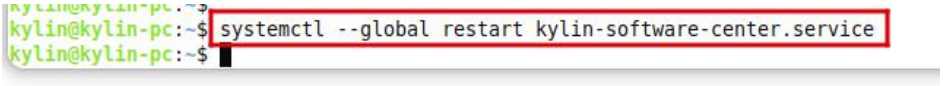
查看软件商店安装状态命令：

```
$ dpkg -l kylin-software-center
```



安装后重启软件商店服务同步数据：

```
$ systemctl --global restart kylin-software-center.service
```



4）查询仓库源里是否有软件包

根据包名查看仓库源是否已有相应软件包。

```
$ apt policy +包名
```

注：图例所示 weixin 在仓库源已有软件包。

若仓库源里有软件包则等待数据同步或手动重启软件商店同步数据，仓库源里没有软件包则需联系软件商店管理员上传软件包至仓库。

如果需查看仓库源中软件各个版本的详细信息可用命令’apt-cache show + 包名’查看。

```
kylin@kylin-pc:~$ apt-cache show weixin
Package: weixin
Version: 2.1.8
Installed-Size: 339739
Maintainer: weixin, Inc <weixin@tencent.com>
Architecture: amd64
Depends: libgtk-3-0, libnotify4, libnss3, libxss1, libxtst6, xdg-utils, libatspi2.0-0, libuuid1, libsecret-1-0
Recommends: libappindicator3-1
Description: 微信桌面版
Description-md5: 9a39b2798d7f733e44889451e5244bcb
Homepage: https://./
Section: default
Priority: optional
License: ISC
Vendor: weixin, Inc <weixin@tencent.com>
Filename: pool/all/weixin_2.1.8_amd64.deb
Size: 91731190
MD5sum: eb1050d60ea0a7996b6afb6547f00403
SHA1: 3f83d7565426b4920df8a1169acac0b36a050965
SHA256: dddf28ec148eab3ddd3023b67cfb535e55d65f12f5a00402d9b3c6e200c9a483
cert_subject_cn: 麒麟软件有限公司
cert_subject_ou: DS120G321040301
cert_subject_o: 麒麟软件有限公司

Package: weixin
Version: 2.1.4-1
Installed-Size: 429964
Maintainer: weixin, Inc <weixin@tencent.com>
Architecture: amd64
Depends: libgtk-3-0, libnotify4, libnss3, libxss1, libxtst6, xdg-utils, libatspi2.0-0, libuuid1, libsecret-1-0
Recommends: libappindicator3-1
Description: 微信桌面版
Description-md5: 9a39b2798d7f733e44889451e5244bcb
Homepage: https://./
Section: default
Priority: optional
License: ISC
Vendor: weixin, Inc <weixin@tencent.com>
Filename: pool/all/weixin_2.1.4-1_amd64.deb
```

5）查看数据库最后同步时间

①终端输入命令

```
$ sqlite3 ~/.cache/uksc/uksc.db
```

②输入搜索指令

```
> select * from dict where key="appinfo_updatetime";
```

```
kylin@kylin-pc:~$ sqlite3 ~/.cache/uksc/uksc.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> select * from dict where key="appinfo_updatetime";
3|appinfo_updatetime|2023-10-19 17:16:33
sqlite>
```

注：图例所示最近一次同步时间是 2023 年 10 月 19 日 17:16:33。

如果时间非当前时间 ，根据软件商店版本取日志。

5.*版本

```
~/.log/kylin-software-center 开头的日志
~/.cache/uksc/uksc.db
~/.cache/uksc/kylin-software-center.conf
```

4.*版本

```
~/.cache/uksc/logs
~/.cache/uksc/uksc.db
~/.cache/uksc/kylin-software-center.conf
```

检查~/.cache/uksc/kylin-software-center.conf 的服务端地址配置，如下地址配置为软件商店公网地址：

若地址变化，则需要用户自行排查解决；

若地址无变化，浏览器输入 https://api.kylinos.cn/api/v1/application/getSyncFrequency，若不能打开，则是用户网络策略限制；若能打开，则提供日志协助排查解决。

若经过上述排查，仍然无法解决，请联系我们处理。

3.5.8 DNS 解析失败导致软件商店使用异常

3.5.8.1 系统版本

适用系统：V10（SP1）

适用架构：全架构

其他版本或架构可做参考。

3.5.8.2 问题描述

软件商店已经更新，但是一直让重新更新。

3.5.8.3 问题分析

可能是 DNS 解析失败。

3.5.8.4 解决方案

1) 查看~/.cache/uksc/log/kylin-software-center-error.log 里面日志，如果出现大量 DNS 解析失败的报错或 DNS 异常的信息，则初步判断为 DNS 存在问题；
2) 如果是公网环境，浏览器尝试访问常规域名看是否能访问，如 https://www.baidu.com ，再尝试访问 https://softwareadmin.kylinos.cn,若不能访问,基本确定是 DNS 问题；
3) 如果浏览器可以访问，则在终端使用 curl www.baidu.com 和 curl softwareadmin.kylinos.cn 看能否正常输出 html 信息，如果不行，则考虑是否网络需要走代理，浏览器做了代理，系统没有；
4) 如果是私网环境，则 2）3）步骤改成私网的某个域名尝试；
5) 公网环境，DNS 地址中增加 114.114.114.114 进行尝试，如果配有多个 DNS 服务地址，将 114 放在最前面；私网环境，需要私网 DNS 服务器的管理员进行处理。

3.5.9 如何将 jar 包转换成可直接执行文件

3.5.9.1 系统版本

适用系统：V10（SP1）

适用架构：全架构

其他版本或架构可做参考。

3.5.9.2 问题描述

部分功能单独一个 jar 包即可实现，执行时用 java -jar 命令调用，但其本身无法作为执行程序打包成 deb 安装或上架商店。

3.5.9.3 问题分析

可以将 jar 包的执行命令转换成脚本，讲该脚本作为可执行程序运行。

3.5.9.4 解决方案

编写脚本 stub.sh

```
$ cat stub.sh
```

内容如下：

```
#!/bin/sh
MYSELF=`which "$0" 2>/dev/null`
[ $? -gt 0 -a -f "$0" ] && MYSELF="./$0"
java=java
if test -n "$JAVA_HOME"; then
    java="$JAVA_HOME/bin/java"
fi
exec "$java" $java_args -jar $MYSELF "$@"
exit 1
```

假如可通过 java -jar 执行的 jar 包名为 app.jar，将命令封装到 app.run：

```
$ cat stub.sh app.jar > app.run && chmod +x app.run
```

则直接执行 app.run 就可运行 java 程序：

```
$ ./app.run
```

如需要 VM 参数，可以在脚本第一行添加：

```
java_args="-Xmx20m"
```

3.5.10 如何配置 wireshark 中的 USB 接口选项

3.5.10.1 系统版本

适用系统：V10（SP1）

适用架构：全架构

其他版本或架构可做参考。

3.5.10.2 问题描述

使用 wireshark 时，无法直接显示并配置 USB 接口。

3.5.10.3 问题分析

需手动挂载 USB 模块。

3.5.10.4 解决方案

1）安装 wireshark

```
kylin@kylin-pc:~/桌面$ sudo apt install wireshark
输入密码
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
wireshark 已经是最新版 (3.2.3-1)。
下列软件包是自动安装的并且现在不需要了：
  archdetect-deb fcitx-libs gnome-desktop3-data libdebian-installer4
  libgeoclue-2-0 libgnome-desktop-3-19 libgrpc++1 libgrpc6 libzip5
  localechooser-data user-setup
使用'sudo apt autoremove'来卸载它(它们)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 464 个软件包未被升
级。
```

2）在/etc/udev/rules.d/路径下创建 99-usbmon.rules 文件并赋权

```
kylin@kylin-pc:~/桌面$ sudo touch /etc/udev/rules.d/99-usbmon.rules
kylin@kylin-pc:~/桌面$ sudo chmod 777 /etc/udev/rules.d/99-usbmon.rules
```

3）在新建的文件中写入规则

```
kylin@kylin-pc:~/桌面$ sudo echo 'SUBSYSTEM=="usbmon", GROUP="usbmon", MODE="640"' > /etc/udev/rules.d/99-usbmon.rules
```

4）添加用户组

```
kylin@kylin-pc:~/桌面$ sudo addgroup usbmon
addgroup: "usbmon"组已经存在。
```

5）指定要管理的工作组（此处以 kylin 为例，实际应用根据所需）

```
kylin@kylin-pc:~/桌面$ sudo gpasswd -a kylin usbmon
正在将用户“kylin”加入到“usbmon”组中
```

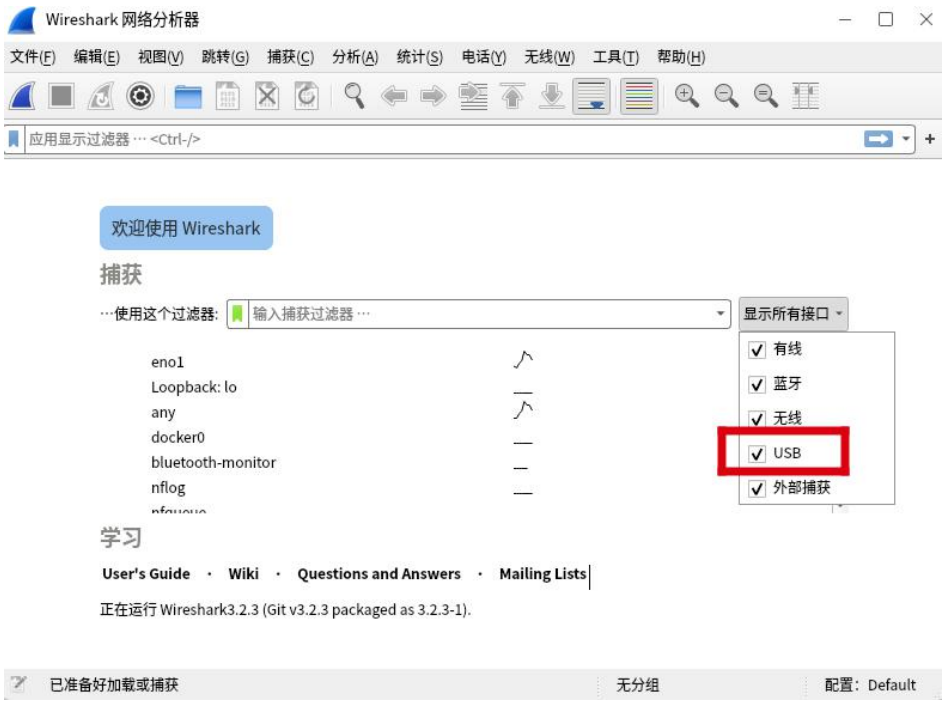
6）载入 usbmon 模块

```
kylin@kylin-pc:~/桌面$ sudo modprobe usbmon
```

7）检查是否挂载成功，如下图所示即为成功

```
kylin@kylin-pc:~/桌面$ lsmod | grep usbmon
usbmon                 28672  0
kylin@kylin-pc:~/桌面$  tcpdump -D
1.eno1 [Up, Running]
2.lo [Up, Running, Loopback]
3.any (Pseudo-device that captures on all interfaces) [Up, Running]
4.docker0 [Up]
5.bluetooth-monitor (Bluetooth Linux Monitor) [none]
6.nflog (Linux netfilter log (NFLOG) interface) [none]
7.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
8.usbmon0 (Raw USB traffic, all USB buses) [none]
9.usbmon1 (Raw USB traffic, bus number 1)
10.usbmon2 (Raw USB traffic, bus number 2)
11.wlx502b73040ea4 [none]
```

8）使用 root 用户打开 wireshark，查看 USB 接口已可配置



3.6 应用打包问题

具体打包可参考《银河麒麟桌面操作系统 V10-DEB 包打包规范》和《银河麒麟桌面操作系统人 V10-DEB 包开发者指南》

3.6.1 应用缺少依赖重新打包

3.6.1.1 系统版本

适用系统：V10(SP1)

适用架构：X86、ARM

其他版本和架构可作参考。

3.6.1.2 问题描述

应用安装后运行提示缺少依赖包，无法正常运行；而用户又是内网环境，无法通过外网源安装依赖时如何解决?

3.6.1.3 问题分析

查看缺少的依赖包，下载依赖包后解压，拷贝需要的库文件到解压后的原 deb 包对应目录中，然后重新打包。

3.6.1.4 解决方案

1) 下载应用包，进行解压：

```
$ dpkg-deb -R ***.deb ***
```

2) 然后找到应用的可执行文件，使用 ldd 查看缺少的依赖有哪些。

```
$ ldd
```

3) 从源里下载缺少的依赖包

```
$ sudo apt downlaod libc-ares2 libgrpc++1 libgrpc6 libportaudio2 libzip5
```

4) 然后再分别解压上面的包

```
$ dpkg-deb -R ***.deb ***
```

5) 解压后，分别进入到每个解压出来的目录，将.so 动态库文件拷贝到 /opt/apps/com.iflytek.iflyime/files/bin/目录下。

```
$ cp -rp *  /opt/apps/com.iflytek.iflyime/files/bin/目录
```

6) 再使用下列命令修改可执行文件的 rpath，要修改 rpath，需要 patchelf 工具，首先安装 patchelf 工具。

a. 安装 patchelf 工具

```
$ sudo apt install patchelf
```

b. 修改 rpath

```
$ patchelf --set-rpath /opt/apps/com.iflytek.iflyime/files/bin/ iflyime-hw
$ patchelf --set-rpath /opt/apps/com.iflytek.iflyime/files/bin/ iflyime-sett
******
```

7) 最后重新进行打包即可

```
$ dpkg-deb -b *** .
```

3.6.2 安装其他的 Qt 版本

3.6.2.1 系统版本

适用系统：V10、V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.6.2.2 问题描述

应用开发的时候需要其他版本 Qt，如何进行安装？

3.6.2.3 问题分析

可以从 Qt 官网下载安装包或源码编译安装，并配置环境变量。

3.6.2.4 解决方案

此处以安装 5.9.0 for Linux 为例

1） 使用从

https://mirrors.tuna.tsinghua.edu.cn/qt/archive/qt/5.9/5.9.0/qt-opensource-linux-x64-5.9.0.run

下载下来的安装文件，更改权限为可执行，然后执行安装程序。不要用管理员权限安装，直接安装到 home 目录即可，例如~/Qt5.9.0。

2） 添加环境变量：

```
$ vim ~/.bashrc

export QT_HOME=/home/kylin/Qt5.9.0/5.9/gcc_64
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${QT_HOME}/lib
export QT_PLUGIN_PATH=$QT_PLUGIN_PATH:${QT_HOME}/plugins
export QML2_IMPORT_PATH=$QML2_IMPORT_PATH:${QT_HOME}/qml
export PATH=${QT_HOME}/bin:$PATH
```

3.6.3 V10(SP1)系统适配为知笔记打包方法

3.6.3.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.6.3.2 问题描述

V10(SP1)系统适配为知笔记如何进行打包？

3.6.3.3 问题分析

有两种方案：

1） AppImage 转 deb 包;

2） 源码打包。

3.6.3.4 解决方案

1） 安装 Qt 相关应用及依赖库

```
$ sudo apt update
$ sudo apt install -y qt5-default qtcreator qttools5-dev-tools qttools5-dev
qtwebengine5-dev libqt5websockets5-dev libqt5svg5-dev
```

2） 安装 Git 等编译环境

```
$ sudo apt-get install -y git build-essential cmake zlib1g-dev patchelf
```

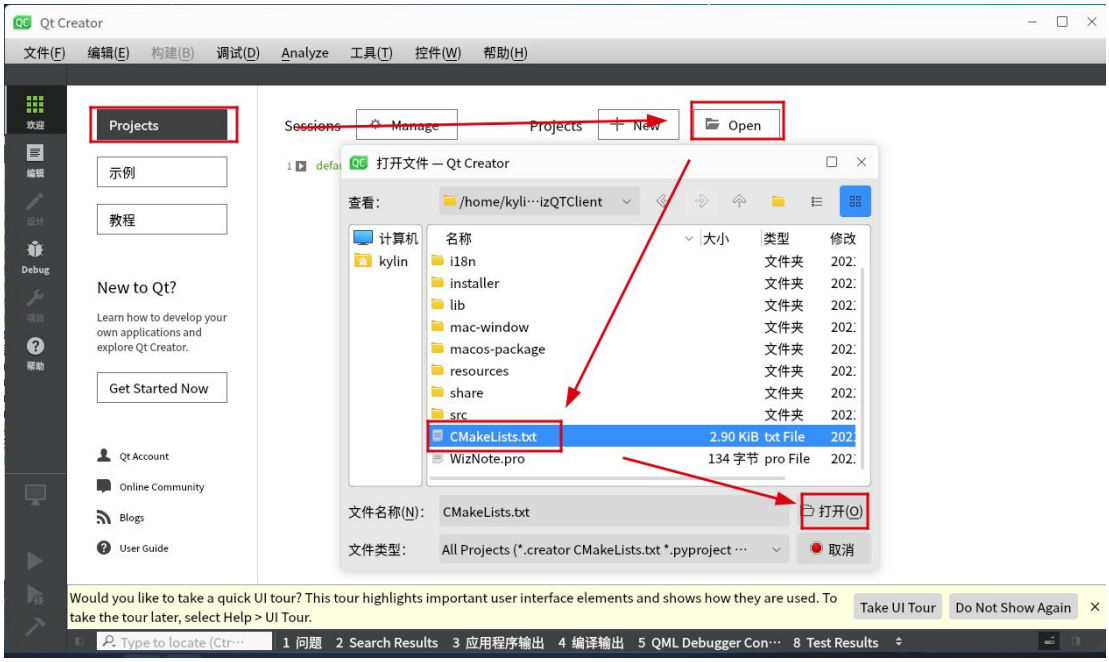
3） Clone 为知笔记源代码

```
$ cd ~
$ mkdir WizTeam
$ cd WizTeam
$ git clone https://hub.fastgit.org/WizTeam/WizQTClient.git
（原始地址 https://github.com/WizTeam/WizQTClient.git）
```

4） 编译源代码

a. 打开项目

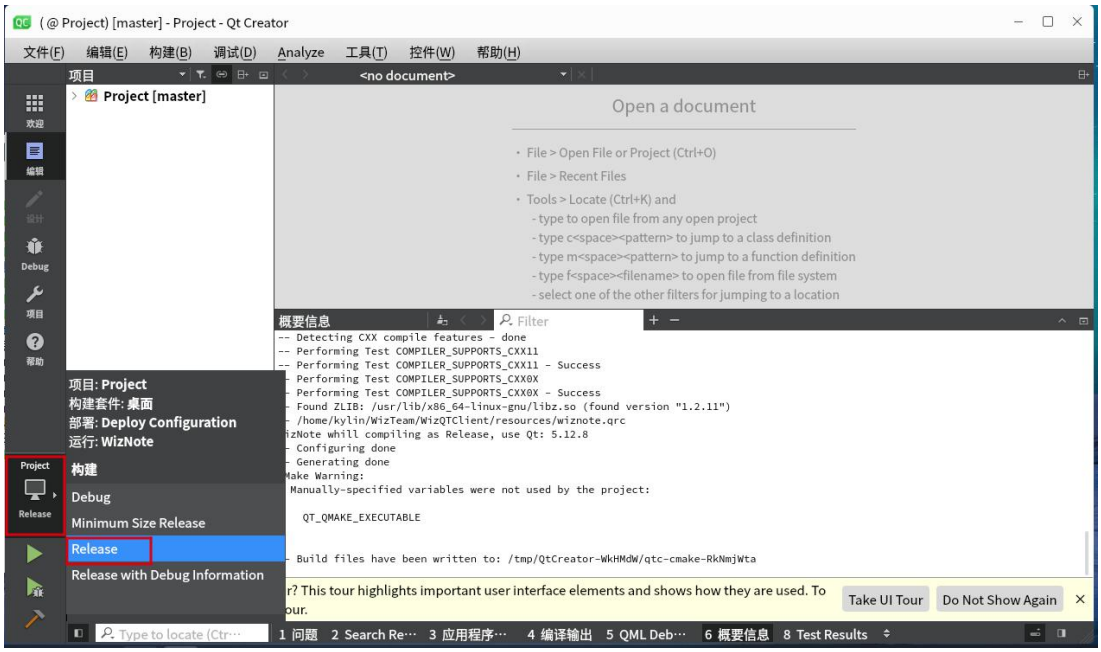
运行 QtCreator，选择打开~/WizTeam/WizQTClient/CMakeLists.txt 这个文件。



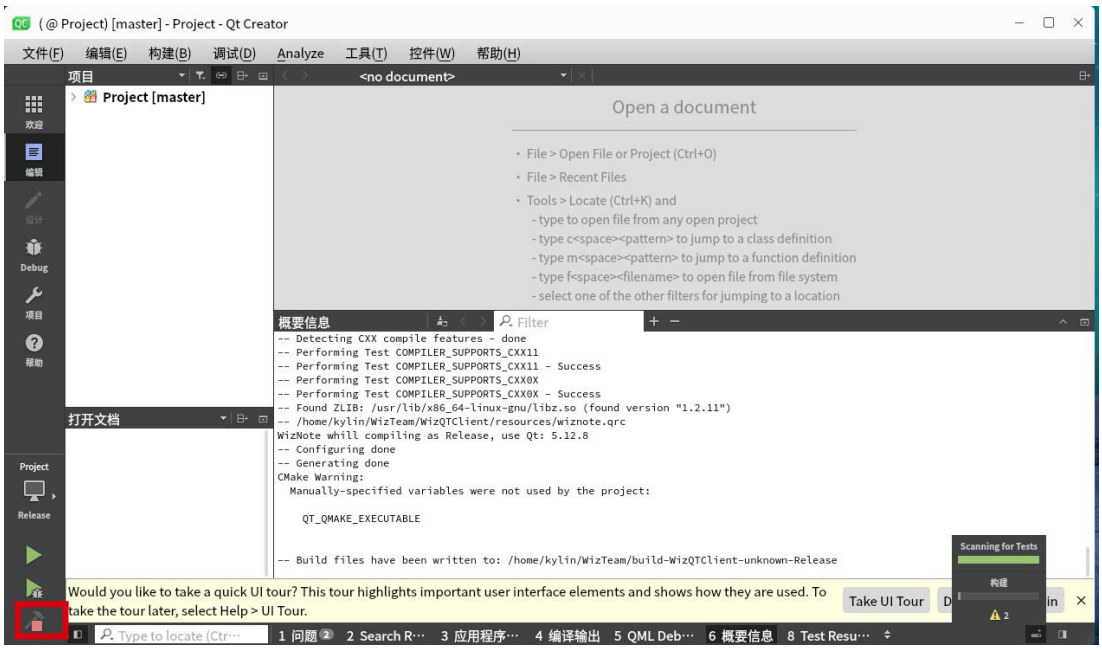
b. 配置项目

c. 构建项目

选择构建 Release 版本



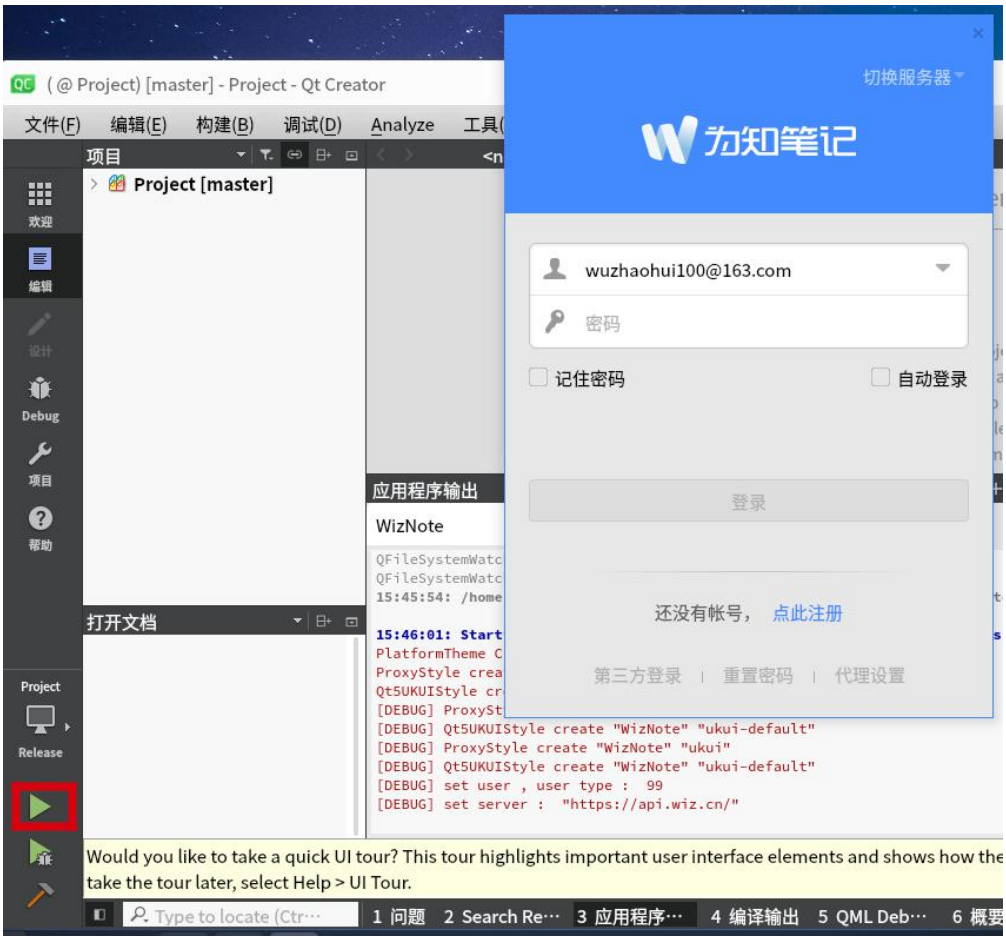
进行项目构建



等待构建完成，问题中没有报错，右下角都是绿色，即代表构建成功。

5) 打包

系统构建完成后，可以点击运行来查看应用是否能够正常运行，也可以直接进行打包。



打包脚本使用的工具为 linuxdeployqt，这个工具会根据生成的可执行文件，自动搜索依赖。

6) 修改打包脚本

打包前我们需要对脚本 linux-package.sh 进行修改

```
$ cd WizQTClient
$ vim linux-package.sh
```

```
# 编译可执行程序
mkdir ../WizQTClient-Release-Linux
rm -rf ../WizQTClient-Release-Linux/*

cd ../WizQTClient-Release-Linux
# 此处需要修改为正确的 Qt 版本
```

```
# cmake -DCMAKE_BUILD_TYPE=Release  ../WizQTClient && \
cmake -DCMAKE_BUILD_TYPE=Release ../WizQTClient && \
make -j2
# 创建 DEBIAN 目录及 control 文件
cd ..
rm -rf wiznote
mkdir wiznote
cd wiznote
rm -rf DEBIAN
mkdir DEBIAN
cd DEBIAN
# 编写 control 文件
cat>control<<EOF
Package: wiznote
Version: 2.8.7
Maintainer: kylin<kylin@kylinos.cn>
Architecture: amd64
Depends:
Description: Wiznote is a note taking software based on cloud storage. With wiznote, you can record inspiration, write documents, organize materials and review diaries at any time; You can also quickly import data and capture web pages, screenshots and text fragments.
Priority: optional
EOF

# 创建应用目录
cd ..
rm -rf opt/apps/wiznote
mkdir -p opt/apps/wiznote
cd opt/apps/wiznote

# 拷贝可执行程序及应用文件到指定目录
mkdir bin
cd bin
cp ../../../../../WizQTClient-Release-Linux/src/WizNote ./wiznote
cp -R ../../../../../WizQTClient/share/qtwebengine_dictionaries ./

cd ..
cp -R ../../../../WizQTClient-Release-Linux/share ./

cd ../../../
# 创建图标文件目录并拷贝图标
mkdir -p usr/share/icons
```

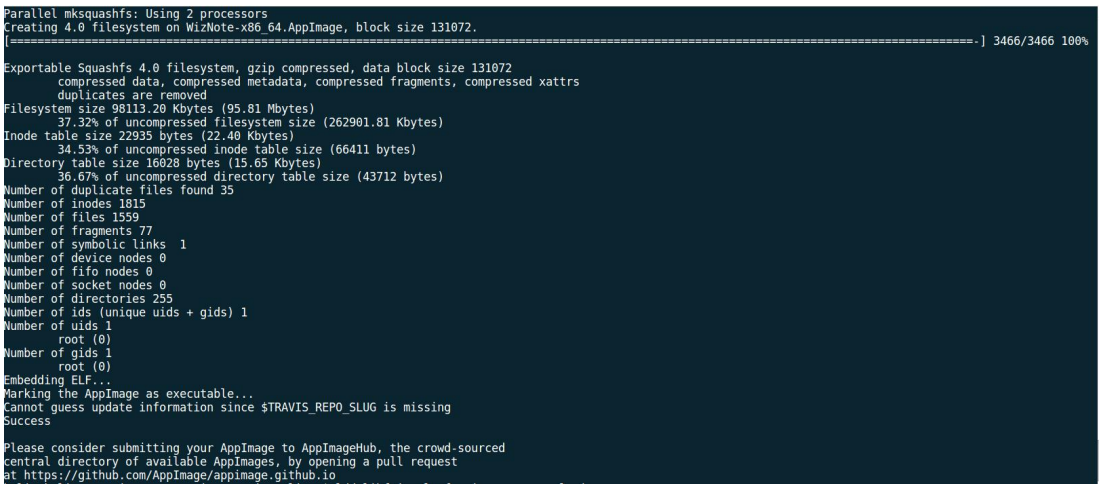
```
cd usr/share/icons
mkdir hicolor
cd hicolor
mkdir -p 16x16/apps
mkdir -p 32x32/apps
mkdir -p 64x64/apps
mkdir -p 128x128/apps
mkdir -p 256x256/apps
mkdir -p 512x512/apps

cp ../../../../../WizQTClient/build/common/logo/wiznote16.png
16x16/apps/wiznote.png
cp ../../../../../WizQTClient/build/common/logo/wiznote32.png
32x32/apps/wiznote.png
cp ../../../../../WizQTClient/build/common/logo/wiznote64.png
64x64/apps/wiznote.png
cp ../../../../../WizQTClient/build/common/logo/wiznote128.png
128x128/apps/wiznote.png
cp ../../../../../WizQTClient/build/common/logo/wiznote256.png
256x256/apps/wiznote.png
cp ../../../../../WizQTClient/build/common/logo/wiznote512.png
512x512/apps/wiznote.png

cd ../../../../
mkdir -p usr/share/applications
cd usr/share/applications
cp ../../../../WizQTClient/build/common/wiznote2.desktop ./wiznote.desktop
sed -i "4c Exec=/opt/apps/wiznote/bin/wiznote" wiznote.desktop
echo 'Name[zh_CN]=为知笔记' >> wiznote.desktop

cd ../../../
mkdir -p opt/apps/wiznote/lib

cd ..

# 此处需要修改为正确的 Qt 版本,由于上面已经添加环境变量，-qmake 直接配置成
qmake 即可
#    ./WizQTClient/linuxdeployqt    ./Package/wiznote.desktop    -verbose=1
-appimage -qmake=../Qt5.9.3/5.9.3/gcc_64/bin/qmake
./WizQTClient/linuxdeployqt ./Package/wiznote.desktop -verbose=1 -appimage
-qmake=qmake
```

7) 打包成 AppImage

a. 在 WizQTClient 目录里面运行

```
$ ./linux-package.sh
```

b. 完成后，如果显示一下信息表示打包完成。



c. 成功运行后可以生成一个 AppImage 文件，双击即可运行，使用 linuxdeployqt 只可以打出一个 AppImage 的包，如果想打成 deb 的包，需要按照 deb 包的格式重新进行打包。

3.6.4 .net 程序打 deb 包

3.6.4.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.6.4.2 问题描述

如何将.net 程序进行打包

3.6.4.3 问题分析

将项目上的.net 程序进行打包，使用 detnet-deb 工具

3.6.4.4 解决方案

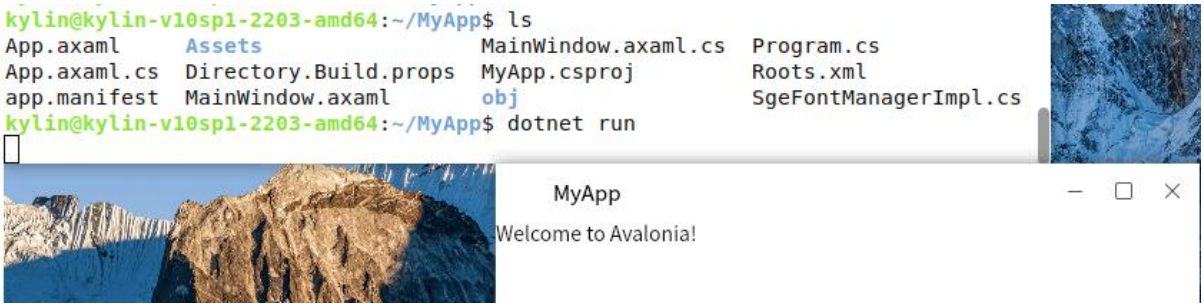
此处我们以 Avalonia 的一个 Demo 程序项目为例，将程序打包成 deb。

首先我们需要将 Demo 程序解压，然后运行 dotnet run 查看程序是否能够正常运行。

```
$ tar xvf AvaloniaDemo-MyApp.tar.gz
```

```
$ cd MyApp
$ dotnet run
```

如下，表示程序可以正常运行，可以进行后续的打包操作。



打包

将项目打包成 deb 安装包，需要对项目做一些设置，设置应用程序生成快捷方式 desktop 文件和应用程序图标。

项目添加 desktop 文件和图标

```
$ cat MyApp.desktop
[Desktop Entry]
Name=MyApp
Type=Application
Exec=Mypp
Icon=MyApp
```

新增 MyApp.desktop 和 MyApp.svg 文件

```
$ ls
App.axaml     MyApp.csproj    Program.cs
App.axaml.cs  Directory.Build.props  MyApp.desktop  Roots.xml
app.manifest  MainWindow.axaml     MyApp.svg     SgeFontManagerImpl.cs
Assets        MainWindow.axaml.cs
```

文件配置

在 MyApp.csproj 中新增 desktop 和图标的安装位置代码

```
<ItemGroup>
        <Content
Include="MyApp.svg" CopyToPublishDirectory="PreserveNewest">
<LinuxPath>/usr/share/icons/hicolor/scalable/apps/MyApp.svg</LinuxPath>
        </Content>
        <Content
Include="MyApp.desktop" CopyToPublishDirectory="PreserveNewest">
```

```
<LinuxPath>/usr/share/applications/MyApp.desktop</LinuxPath>
  </Content>
</ItemGroup>
```

安装.net 打包 deb 工具

对.net 程序进行打包，可以采用 dotnet-packaging 工具。

执行 dotnet tool install --global dotnet-deb 命令安装 dotnet-deb 工具

```
$ dotnet tool install --global dotnet-deb
```

由于刚安装了 .NET SDK，因此在运行安装的工具之前，需要注销或重启会话。

可使用以下命令调用工具:

```
$ dotnet-deb
已成功安装工具“dotnet-deb”(版本“0.1.220”)。
```

如果是刚安装了.NET SDK，需要先注销或重启后，再进行后面的操作，否则会有如下提示

```
$ dotnet deb install
```

无法执行，因为找不到指定的命令或文件。

可能的原因包括:

*内置的 dotnet 命令拼写错误。

*你打算执行 .NET 程序，但 dotnet-deb 不存在。

*你打算运行全局工具，但在路径上找不到具有此名称且前缀为 dotnet 的可执行文件

在项目目录下，添加 deb 工具

```
$ cd MyApp
$ dotnet deb install
dotnet deb (0.1.220+3f7bd3c61a)
Successfully installed dotnet deb. Now run 'dotnet deb' to package your application as a Debian/Ubuntu installer package
```

打包

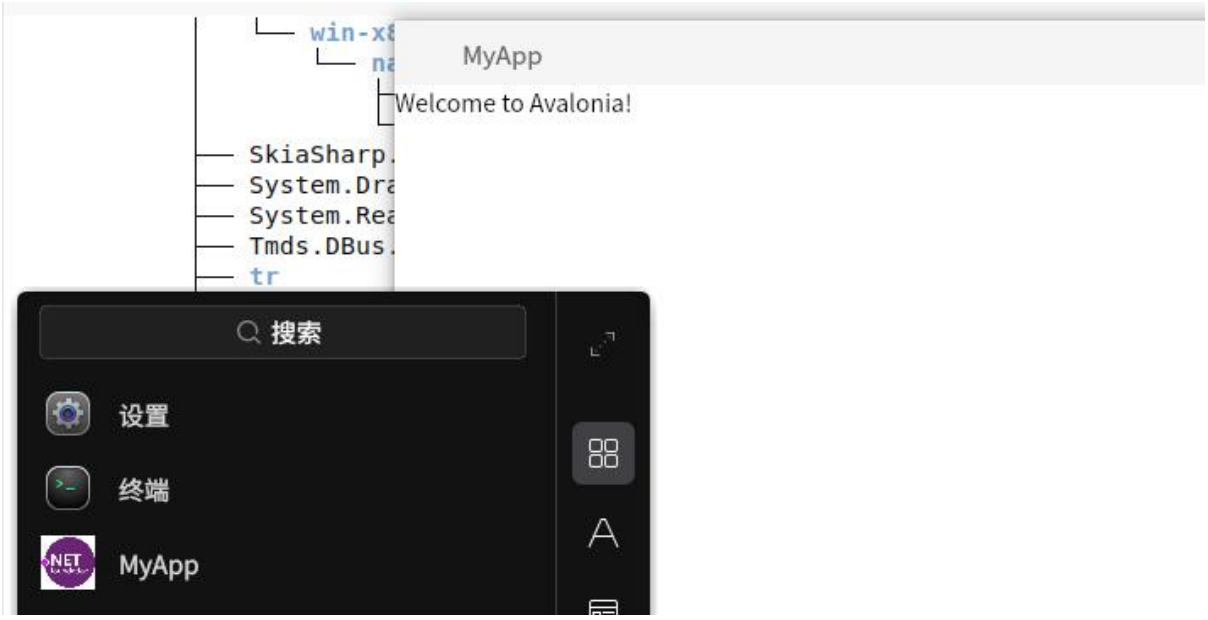
在项目目录下，执行 dotnet deb 进行打包

```
$ dotnet deb -c Release
dotnet deb (0.1.220+3f7bd3c61a)
MSBuild version 17.3.2+561848881 for .NET
    GenerateAvaloniaResourcesTask -> Root: /home/kylin/MyApp, 2 resources, Output:obj/Release/net6.0//Avalonia/resources
    avares -> name:/App.axaml, path: /home/kylin/MyApp/App.axaml, size:255, ItemSpec:App.axaml
    avares -> name:/MainWindow.axaml, path: /home/kylin/MyApp/MainWindow.axaml, size:416, ItemSpec:MainWindow.axaml
    MyApp -> /home/kylin/MyApp/bin/Release/net6.0/MyApp.dll
    MyApp -> /home/kylin/MyApp/bin/Release/net6.0/publish/
    Creating
DEB package '/home/kylin/MyApp/bin/Release/net6.0/MyApp.1.0.0.deb' from folder 'bin/Release/net6.0/publish/'
Created
DEB package '/home/kylin/MyApp/bin/Release/net6.0/MyApp.1.0.0.deb' from folder 'bin/Release/net6.0/publish/'
```

打包完成后，可以看到 deb 包生成到 /home/kylin/MyApp/bin/Release/net6.0/MyApp.1.0.0.deb。

验包

```
$ sudo dpkg -i MyApp.1.0.0.deb
正在选中未选择的软件包 myapp。
(正在读取数据库 ... 系统当前共安装有 196228 个文件和目录。)
准备解压 MyApp.1.0.0.deb ...
正在解压 myapp (1.0.0) ...
正在设置 myapp (1.0.0) ...
正在处理用于 desktop-file-utils (0.24-1kylin2) 的触发器 ...
正在处理用于 bamfdaemon (0.5.3+18.04.20180207.2-0kylin2) 的触发器 ...
Rebuilding /usr/share/applications/bamf-2.index...
正在处理用于 mime-support (3.64kylin1) 的触发器 ...
正在处理用于 hicolor-icon-theme (0.17-2) 的触发器 ...
```



3.6.5 git@github.com:Permission denied(publickey)报错解决方法

3.6.5.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.6.5.2 问题描述

使用 npm i 安装依赖的时候有如下报错：

```
npm verb stack Error: command failed
npm verb stack at ChildProcess.<anonymous>
(/home/kylin/.nvm/versions/node/v16.5.0/lib/node_modules/npm/node_modul
es/@npmcli/promise-spawn/index.js:64:27)
npm verb stack at ChildProcess.emit (node:events:394:28)
npm verb stack at maybeClose (node:internal/child_process:1067:16)
npm verb stack at Process.ChildProcess._handle.onexit
(node:internal/child_process:301:5)
npm verb cwd /home/kylin/electerm
npm verb Linux 5.4.18-55-generic
npm verb
argv "/home/kylin/.nvm/versions/node/v16.5.0/bin/node" "/home/kylin/.nvm/v
ersions/node/v16.5.0/bin/npm" "i" "--verbose"
npm verb node v16.5.0
npm verb npm v7.19.1
npm ERR! code 128
npm ERR! command failed
```

```
npm ERR! command git --no-replace-objects ls-remote ssh://git@github.com/vscode-icons/vscode-icons.git
npm ERR! Warning: Permanently added 'github.com,20.205.243.166' (ECDSA) to the list of known hosts.
npm ERR! git@github.com: Permission denied (publickey).
npm ERR! fatal: 无法读取远程仓库。
npm ERR!
npm ERR! 请确认您有正确的访问权限并且仓库存在。
```

3.6.5.3 问题分析

秘钥设置的有问题

3.6.5.4 解决方案

首先需要有 github 账号，没有需要注册。

检查有没有在 GitHub 的 https://github.com/settings/keys 上添加你本机的 SSH key。注意换了电脑是要重新添加的，每台都不一样。

添加 SSH key 的方法：

（1）在用户主目录下，看看有没有.ssh 目录，如果有，再看看这个目录下有没有 id_rsa 和 id_rsa.pub 这两个文件，如果已经有了，可直接跳到下一步。如果没有，打开终端，创建 SSH Key：

```
$ ssh-keygen -t rsa -C "wuzhaohui100@163.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kylin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kylin/.ssh/id_rsa
Your public key has been saved in /home/kylin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:bz2TwOS8t6KHgffGtSO8Ereu3/Ae8dDWmaSoZ7Njqo8
    wuzhaohui100@163.com
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|                 |
|         .    . |
|        =   ..o.o|
```

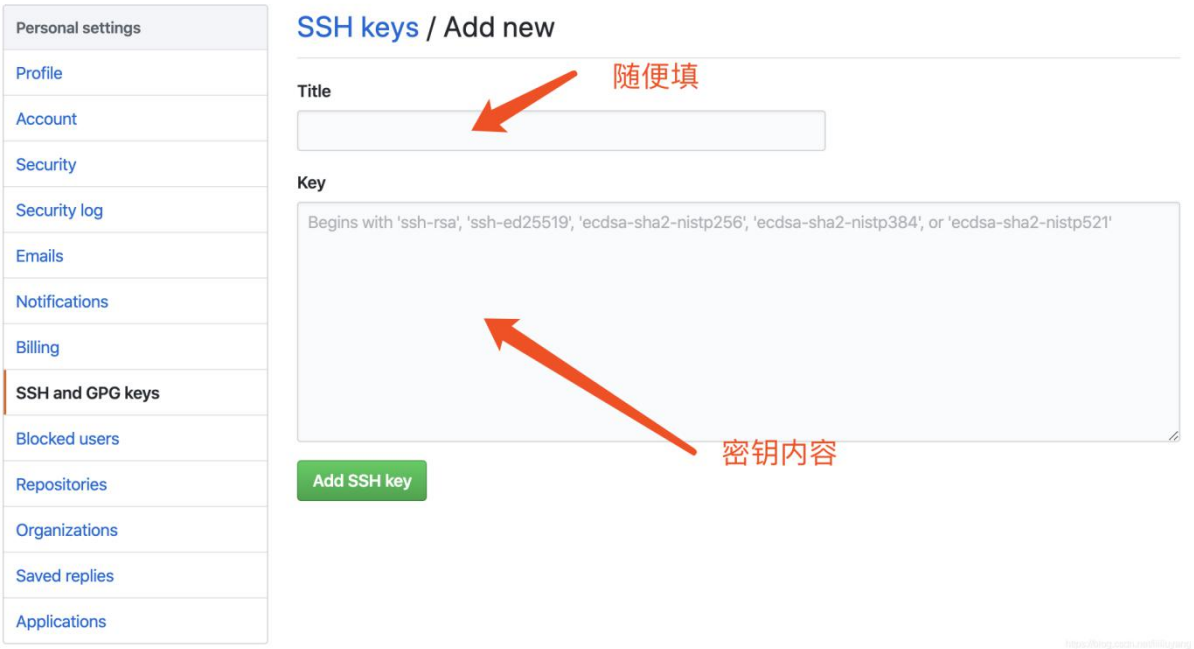
```
|       .S =.o.oo.|
|      . +.o+.*   |
|       . X=*B..  |
|        +.%B==   |
|       E=@***.   |
+----[SHA256]-----+
```

把邮件地址换成你自己注册 github 的邮件地址，然后一路回车，使用默认值即可。

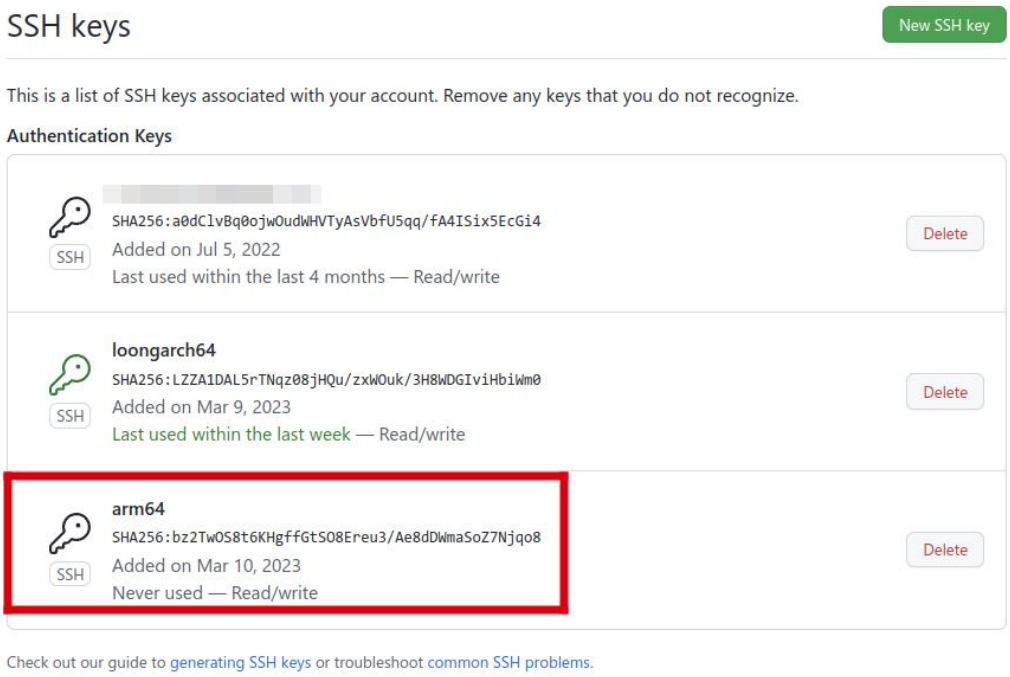
如果一切顺利的话，可以在用户主目录里找到.ssh 目录，里面有 id_rsa 和 id_rsa.pub 两个文件，这两个就是 SSH Key 的秘钥对，id_rsa 是私钥，不能泄露出去，id_rsa.pub 是公钥，可以放心地告诉任何人。

（2）登录 GitHub，打开“Account settings”，“SSH Keys”页面：

然后，点“Add SSH Key”，填上任意 Title，在 Key 文本框里粘贴 id_rsa.pub 文件的内容：



点“Add Key”，你就应该看到已经添加的 Key：



添加后重新执行命令获取依赖即可。

3.6.6 npm yarn 设置代理，解决依赖拉取失败问题

3.6.6.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.6.6.2 问题描述

electron 应用打包时，会用到 npm 或 yarn 来拉取依赖，但总出现依赖拉取不成功的问题，报错如下：

```
npm ERR! code 1
npm ERR! path /home/wyx/code/electron-quick-start-master/node_modules/electron
npm ERR! command failed
npm ERR! command sh -c node install.js
npm ERR! ReadError: The server aborted pending request
npm ERR!     at IncomingMessage.<anonymous> (/home/wyx/code/electron-quick-start-master/node_modules/got/dist/source/core/index.js:809:31)
npm ERR!     at Object.onceWrapper (node:events:641:28)
```

```
npm ERR!     at IncomingMessage.emit (node:events:539:35)
npm ERR!     at IncomingMessage.origin.emit
(/home/wyx/code/electron-quick-start-master/node_modules/@szmarczak/htt
p-timer/dist/source/index.js:43:20)
npm ERR!     at IncomingMessage._destroy (node:_http_incoming:179:10)
npm ERR!     at _destroy (node:internal/streams/destroy:102:25)
npm ERR!     at IncomingMessage.destroy
(node:internal/streams/destroy:64:5)
npm ERR!     at TLSSocket.socketCloseListener (node:_http_client:414:11)
npm ERR!     at TLSSocket.emit (node:events:539:35)
npm ERR!     at node:net:709:12
RequestError: connect ETIMEDOUT 20.205.243.166:443
                                    at ClientRequest.<anonymous>
(/home/wyx/code/joplin/packages/app-desktop/node_modules/@electron/get/
node_modules/got/source/request-as-event-emitter.js:178:14)
   at Object.onceWrapper (node:events:642:26)
   at ClientRequest.emit (node:events:539:35)
                                    at ClientRequest.origin.emit
(/home/wyx/code/joplin/packages/app-desktop/node_modules/@electron/get/
node_modules/@szmarczak/http-timer/source/index.js:37:11)
   at TLSSocket.socketErrorListener (node:_http_client:454:9)
   at TLSSocket.emit (node:events:527:28)
   at emitErrorNT (node:internal/streams/destroy:157:8)
   at emitErrorCloseNT (node:internal/streams/destroy:122:3)
   at processTicksAndRejections (node:internal/process/task_queues:83:21)
```

3.6.6.3 问题分析

使用 npm 或 yarn 拉取依赖时，基本都是从 Github 或者 npm 官方源中拉取，容易由于网络因素出现这种问题。

3.6.6.4 解决方案

我们可以采取如方法规避：

使用替代工具 cnpm

可以使用淘宝 NPM 镜像定制的 cnpm (gzip 压缩支持) 命令行工具代替默认的 npm:

```
$ npm install -g cnpm --registry=https://registry.npmmirror.com
```

这样就可以使用 cnpm 命令来安装模块了：

```
$ cnpm install [name]
```

设置代理；

下面介绍如何设置代理：

（1）设置 registry

通过将 registry 设置为淘宝源，基本可以解决大部分问题

npm 设置 registry:

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

yarn 设置 registry

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

如果使用的是高版本的 yarn，执行上面的命令会有报错：

```
Usage Error: Couldn't find a configuration settings named "registry"
```

原因是高版本的 yarn 设置 registry 的名称变了，可以使用 yarn config 查看，需要使用如下命令设置

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

参考：https://yarnpkg.com/configuration/yarnrc

注：龙芯架构需要使用龙芯自己搭建的源

设置 registry 为龙芯源

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

（2）设置 electron 镜像源

npm 设置 electron_mirror 为淘宝源:

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

npm 设置 electron_mirror 为龙芯源：

#mips64el 架构

```
$ npm config set electron_mirror
```

```
http://ftp.loongnix.cn/os/loongnix/1.0/electron/releases/mips/
#loongarch64 架构
$ npm config set electron_mirror http://ftp.loongnix.cn/electron/LoongArch/
```

yarn 设置 electron_mirror 为淘宝源:

```
$ yarn config set registry https://registry.npm.taobao.org/
$ yarn config set disturl https://npm.taobao.org/dist
$ yarn config set electron_mirror https://npm.taobao.org/mirrors/electron/
```

其他不常用的还有

```
$ npm install phantomjs --phantomjs_cdnurl=http://npm.taobao.org/mirrors/phantomjs
$ npm install chromedriver --chromedriver_cdnurl=http://npm.taobao.org/mirrors/chromedriver
$ npm install operadriver --operadriver_cdnurl=http://npm.taobao.org/mirrors/operadriver
$ npm config set sass_binary_site https://npm.taobao.org/mirrors/node-sass/
$ npm config set phantomjs_cdnurl https://npm.taobao.org/mirrors/phantomjs/
```

```
$ yarn config set sass_binary_site https://npm.taobao.org/mirrors/node-sass/
$ yarn config set phantomjs_cdnurl https://npm.taobao.org/mirrors/phantomjs/
$ yarn config set chromedriver_cdnurl https://cdn.npm.taobao.org/dist/chromedriver
$ yarn config set operadriver_cdnurl https://cdn.npm.taobao.org/dist/operadriver
$ yarn config set fse_binary_host_mirror https://npm.taobao.org/mirrors/fsevents
```

当然，如果有可用的代理服务器也可以直接设置代理，通过代理服务器来拉取依赖，就不需要设置淘宝源。

删除镜像源设置：

```
$ unset $ELECTRON_MIRROR
```

或直接修改~/.npmrc 文件，删除 electron_mirror 字段。

或配置 electron_mirror=https://npmmirror.com/mirrors/electron/

（3）设置 proxy

NPM 设置代理：

```
$ npm config set proxy="<http_proxy>"
$ npm config set https-proxy="<https_proxy>"
```

NPM 删除代理：

```
$ npm config delete proxy
$ npm config delete https-proxy
```

YARN 设置代理:

```
$ yarn config set proxy <http_proxy>
$ yarn config set https-proxy <https_proxy>
```

YARN 删除代理:

```
$ yarn config delete proxy
$ yarn config delete https-proxy
```

3.6.7 yarn 安装方法

3.6.7.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.6.7.2 解决方案

安装 npm，安装 yarn

```
$ sudo apt update
$ sudo apt install npm
$ sudo npm install -g yarn --registry=https://registry.npm.taobao.org
```

查看 yarn 版本

```
$ yarn -v
```

某些系统上使用上面的命令安装 yarn 后会有如下提示，显示 yarn 没有安装成功

```
$ sudo npm install -g yarn --registry=https://registry.npm.taobao.org

> yarn@1.22.19 preinstall /usr/local/lib/node_modules/yarn
> :; (node ./preinstall.js > /dev/null 2>&1 || true)

$ yarn -v
```

```
Could not find command-not-found database. Run 'sudo apt update' to populate it.
yarn：未找到命令
```

需要按照提示手动做一个链接即可

```
$ sudo ln -s /usr/local/lib/node_modules/yarn/bin/yarn /usr/local/bin/yarn
```

验证：

```
$ yarn -v
1.22.19
```

3.6.8 使用 electron-builder 打包报 fpm 问题解决方法

3.6.8.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.6.8.2 问题描述

使用 electron-builder 进行打包时，有时会出现 fpm 拉取失败或如下不支持某些架构导致的报错：

```
$ 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=10.1.5 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-bin
```

aries/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 libuuid1 -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 'XX XXXXXXX <XXXXXXXXX@kylinos.cn>' --url 'https://github.com/electron/electron-quick-start#readme' --vendor 'XXX XXXXXXX <XXXXXXXXX@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/share/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/share/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/share/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/share/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$
```

3.6.8.3 问题分析

使用 electron-builder 打包时默认会从 github 拉取 fpm 包，如果无法连接 github 就会出现无法打包成功的情况。另外 fpm 包 npm 仓库源中仅有 X86_64 架构的包，没有其他架构的包，所以也会有上述报错。

3.6.8.4 解决方案

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"
```

如果使用 gem 安装 fpm 时没有响应，可以通过替换 gem 源后重新安装：

```
$ sudo gem sources --add https://mirrors.tuna.tsinghua.edu.cn/rubygems/ --remove https://rubygems.org/
```

3.6.9 如何将图片由 PNG 格式转换成 SVG 格式

3.6.9.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.6.9.2 问题描述

如何将图片由 PNG 格式转换成 SVG 格式。

3.6.9.3 问题分析

可以通过 Inkscape 软件实现，或者通过命令行实现

3.6.9.4 解决方案

1)使用 Inkscape 软件

安装 inkscape 软件

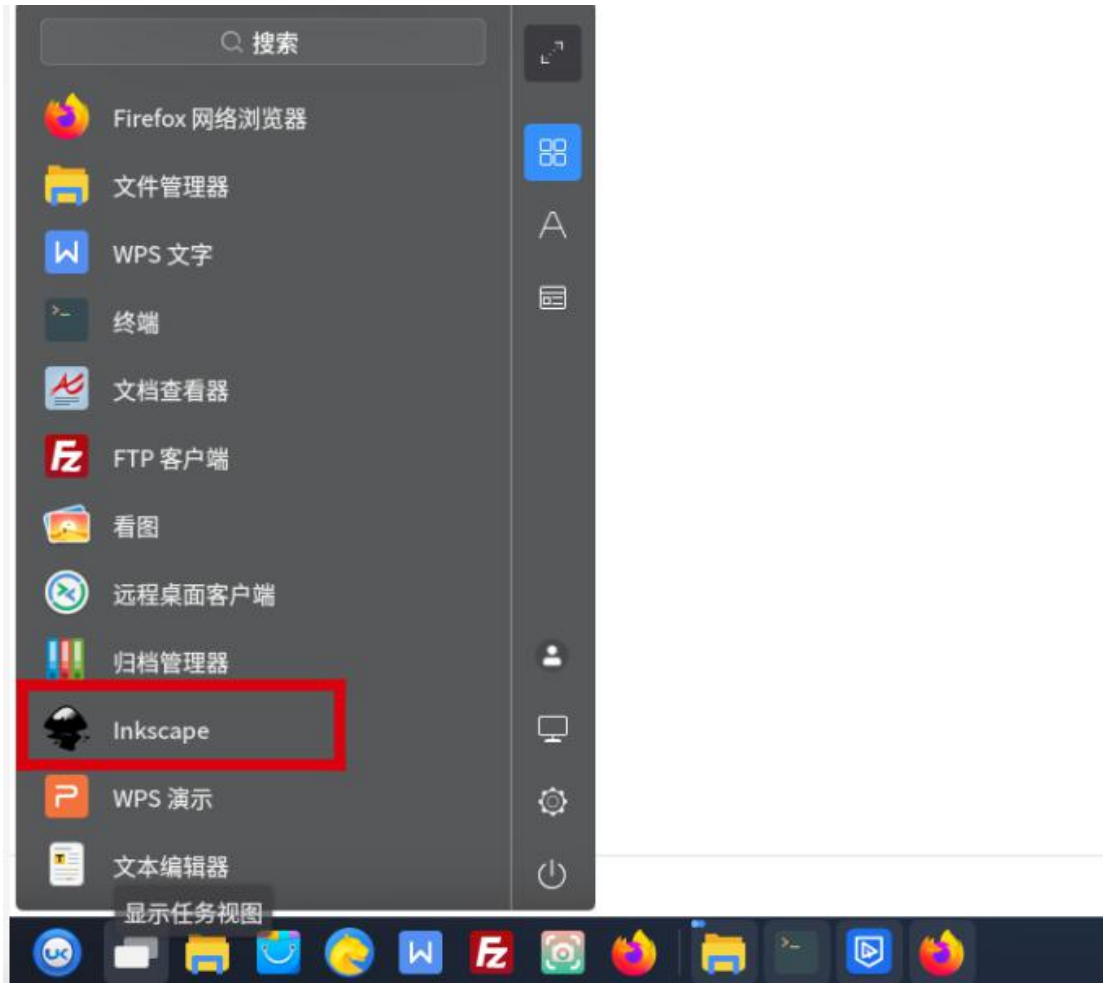
```
$ sudo apt install inkscape
```

安装 python-lxml 依赖包

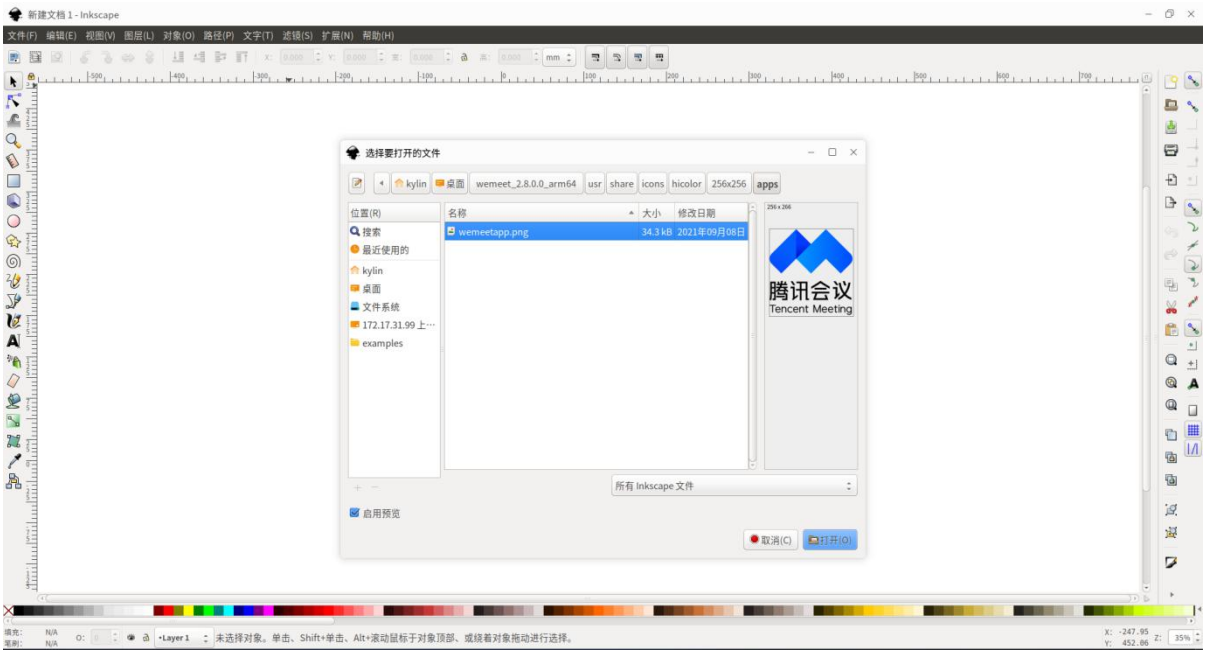
```
$ sudo apt install python-lxml
```

使用 inkscape 软件打开 PNG 图像；

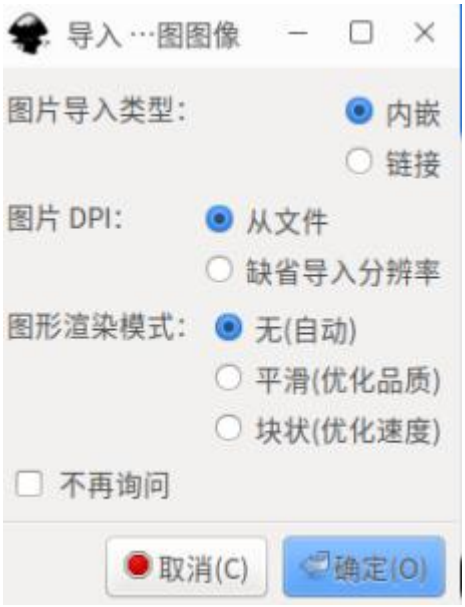
在开始菜单栏中打开 inkscape 软件程序；



点击【文件】——【打开】，选择一个 PNG 格式图片

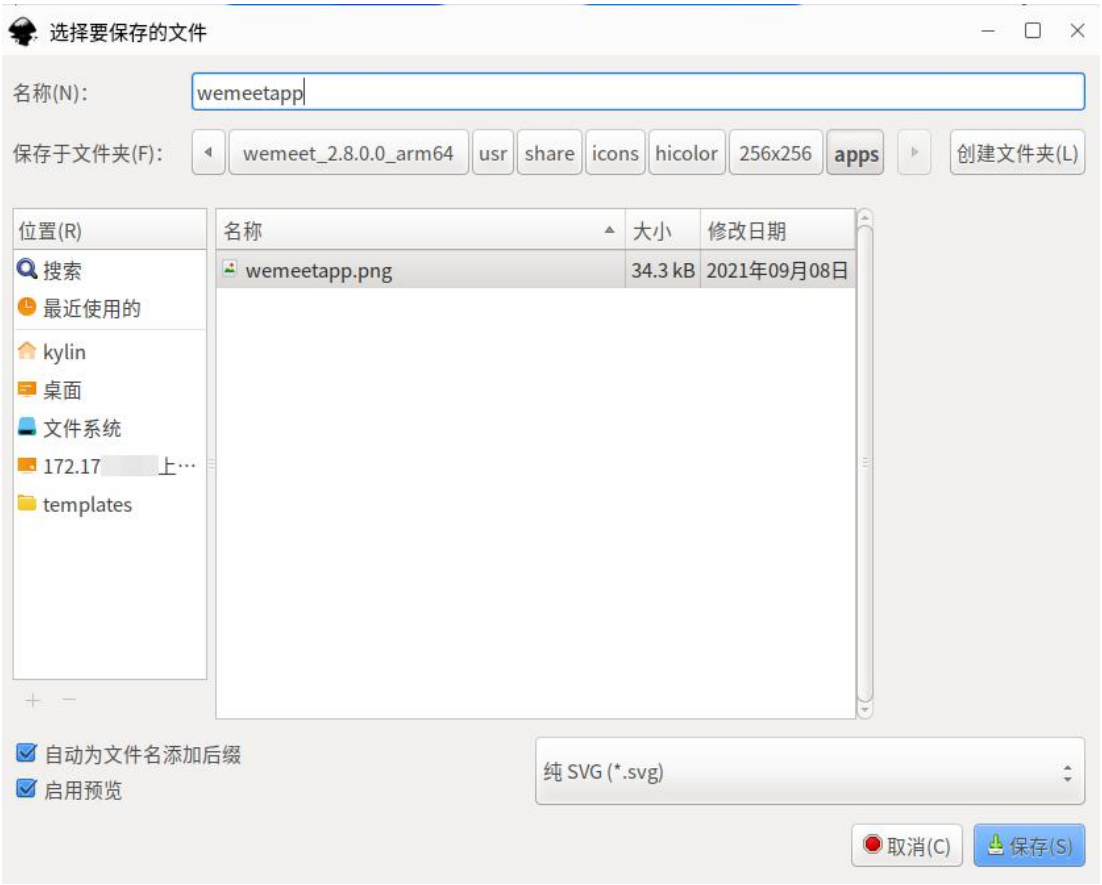


点中图像，选择命令 效果->图像->嵌入所有图像



保存图片

选择文件→另存为



2)使用命令行对图片格式转换

png 转 icns

```
$ sudo npm i -g make-icns --unsafe-perm

$ mk-icns *.png .
```

https://askubuntu.com/questions/223215/how-can-i-convert-a-png-file-to-icns

PNG 转 SVG

```
$ inkscape jeff.png --export-plain-svg=jeff.svg -z

$inkscape
512x512/apps/mqttx.png  --export-plain-svg=scalable/apps/mqttx.svg
```

类型转换

```
$ convert old.jpg new.png     # 进行图片格式转换，改一下后缀名即可
```

转换为 jpg 图片会比 png 图片小

SVG 转 PNG

尝试使用以下命令将 in.svg 转换为 1000×1000 png 图像：

```
$ inkscape -z -e out.png -w 1000 -h 1000 in.svg
```

图片缩放和压缩

```
## 按尺寸缩放：缩放为 80x60 的尺寸大小
# 可以使用 -sample 和 -resize：
# 其中：-sample 使用一个简单的算法生成缩略图，速度快，画质较低，适合生成 100x100 以下的图片
# -resize 画质好
convert -sample 80x60 input.jpg output.jpg   # 画质较低
convert -resize 80x60 input.jpg output.jpg # 画质好
convert -resize 80x60\! input.jpg   output.jpg   # 强制按给定的宽高缩放

## 按比例缩放：缩放后图像保持原来的长宽比例
convert -resize 50%x50% input.jpg output.jpg  # 将图像缩小为原来的 50%*50%
convert -sample 25%x25% input.jpg output.jpg # 将图像缩小为原来的 25%*25%
```

```
$ convert -resize 256x256 512x512/apps/mqttx.png 256x256/apps/mqttx.png
$ convert -resize 128x128 512x512/apps/mqttx.png 128x128/apps/mqttx.png
```

```
$ convert -resize   64x64 512x512/apps/mqttx.png 64x64/apps/mqttx.png
$ convert -resize   32x32 512x512/apps/mqttx.png 32x32/apps/mqttx.png
$ convert -resize   16x16 512x512/apps/mqttx.png 16x16/apps/mqttx.png
```

```
$ convert -quality 70% input.jpg output.jpg # -quality 降低图片的质量，值 0-100，越大质量越好，默认值为 75%
# 去除图片多余 exif 信息，比如日期、相机型号、GPS 使用 -strip
$ convert -strip input.jpg output.jpg
```

3.6.10 如何使用桌面快捷方式启动需要 sudo 权限的应用

3.6.10.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.6.10.2 问题描述

很多应用需要 sudo 权限才能启动，但在 desktop 中加上 sudo 后依然无法启动

3.6.10.3 问题分析

没有赋予桌面快捷方式 sudo 权限，无法获取 sudo 权限，导致应用无法启动

3.6.10.4 解决方案

给桌面快捷方式添加 sudo 权限

比如 xampp.desktop

```
$ cat xampp.desktop
[Desktop Entry]
Version=1.0
Type=Application
Name=XAMPP
Exec=sudo /opt/lampp/manager-linux-x64.run
Icon=/opt/lampp/htdocs/favicon.ico
Terminal=false
StartupNotify=false
```

给桌面快捷方式添加 sudo 权限

```
$ sudo visudo
#在文件的最后添加如下内容
your-user ALL = NOPASSWD: /opt/lampp/manager-linux-x64.run
```

注意：不要忘记将 your-user 替换为您在系统上用于运行 XAMPP 的当前用户。

3.6.11 NxShell 编译打包

3.6.11.1 系统版本

适用系统：V10(SP1)

适用架构：ALL

其他版本和架构可作参考。

3.6.11.2 解决方案

安装 nodejs

```
$ sudo apt install npm
```

由于系统源中的 node 版本较低，需要使用 n 模块或 nvm 模块来安装高版本的 nodejs，此处我们使用 node 16.5.0。

X86_64 和 arm64 安装方法

```
$ sudo npm i -g n
$ sudo n 16.5.0
```

loongarch64 安装方法

参考 http://www.loongnix.cn/zh/nodejs/2022/06/08/Node.js-v16.5.0-loongarch64%E7%89%88%E6%9C%AC%E5%8F%91%E5%B8%83/

设置 npm 代理

x86_64 和 arm64

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

loongarch64，参考 3.3.6

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

拉取代码

```
$ git clone https://github.com/nxshell/build.git
$ cd build/build
```

修改源码

由于当前版本需要 electron17.4.1，但 loongarch64 架构只有 17.4.0 版本，需要修改 package.json 文件中 electron 版本为 17.4.0

修改 artifactName 格式，保证输出的 deb 包名为规范的格式。

修改 linux 下打包的格式，只保留 deb 即可。

```
$ cat build.yml
appId: org.nxshell.shell
productName: NxShell
copyright: Copyright © 2020-2021 NxShell Team
artifactName: nxshell_${version}_${arch}.${ext}
electronDownload:
  mirror: https://npm.taobao.org/mirrors/electron/
extraResources:
  - from: ../pack/native/node_modules
    to: node_modules
  - from: ../resources/nxshell.png
    to: nxshell.png
files:
  - filter: "!native"
directories:
  buildResources: ../resources
  output: ../dist/apppackage
  app: ../pack
target:
  - target: zip
win:
  icon: ../resources/appx/Square512x512Logo.png
  target:
   - appx
   - nsis
appx:
  publisherDisplayName: nxshell
  identityName: 64354nxshell.nxshell
  publisher: CN=13CA10C8-3E4F-483D-844A-FB63CF8E93F7
```

```
  applicationId: nxshell.com
nsis:
  oneClick: false
  allowToChangeInstallationDirectory: true
linux:
  target:
   - deb
  icon: ../resources/icons
  category: Development
mac:
  target: dmg
  icon: ../resources/mac/icon-128.icns
  category: public.app-category.developer-tools
afterPack: ./afterPackHook.js
```

由于打 deb 包时会用到 fpm，除 x86_64 架构外都会有报错，需要参考 3.6.8 章节

拉取依赖：

```
$ npm i
```

编译打包：

```
$ node build.js
```

loongarch64 架构，源码拉取后，源码 core/package.json 中的 electron 的版本也需要进行修改为 17.4.0。

构建包将位于 build/dist。

3.7 应用设置问题

3.7.1 pango 相关依赖库版本不匹配问题

3.7.1.1 系统版本

适用系统：V10(SP1)

适用架构：X86、ARM

其他版本和架构可作参考。

3.7.1.2 问题描述

部分软件安装后无法打开，比如：蚂蚁笔记，深信服 VPN，EasyConnect，钉钉。

3.7.1.3 问题分析

不是由于安装包缺失依赖，而是依赖包版本不符。将软件需要的正确依赖包版本放置到该运行程序所在目录。

3.7.1.4 解决方案

X86：

1） libpango-1.0-0_1.42.4-7_amd64.deb

下载链接：

https://launchpad.net/ubuntu/+source/pango1.0/1.42.4-7/+build/17448184/+files/libpango-1.0-0_1.42.4-7_amd64.deb

2） libpangocairo-1.0-0_1.42.4-7_amd64.deb

下载链接：

https://launchpad.net/ubuntu/+source/pango1.0/1.42.4-7/+build/17448184/+files/libpangocairo-1.0-0_1.42.4-7_amd64.deb

3） libpangoft2-1.0-0_1.42.4-7_amd64.deb

下载链接：

https://launchpad.net/ubuntu/+source/pango1.0/1.42.4-7/+build/17448184/+files/libpangoft2-1.0-0_1.42.4-7_amd64.deb

ARM:

1） libpango-1.0-0_1.42.4-7_arm64.deb

下载链接：

https://launchpad.net/ubuntu/+source/pango1.0/1.42.4-7/+build/17448185/+files/libpango-1.0-0_1.42.4-7_arm64.deb

2） libpangocairo-1.0-0_1.42.4-7_arm64.deb

下载链接：

https://launchpad.net/ubuntu/+source/pango1.0/1.42.4-7/+build/17448185/+files/libpangocairo-1.0-0_1.42.4-7_arm64.deb

3) libpangoft2-1.0-0_1.42.4-7_arm64.deb

下载链接：

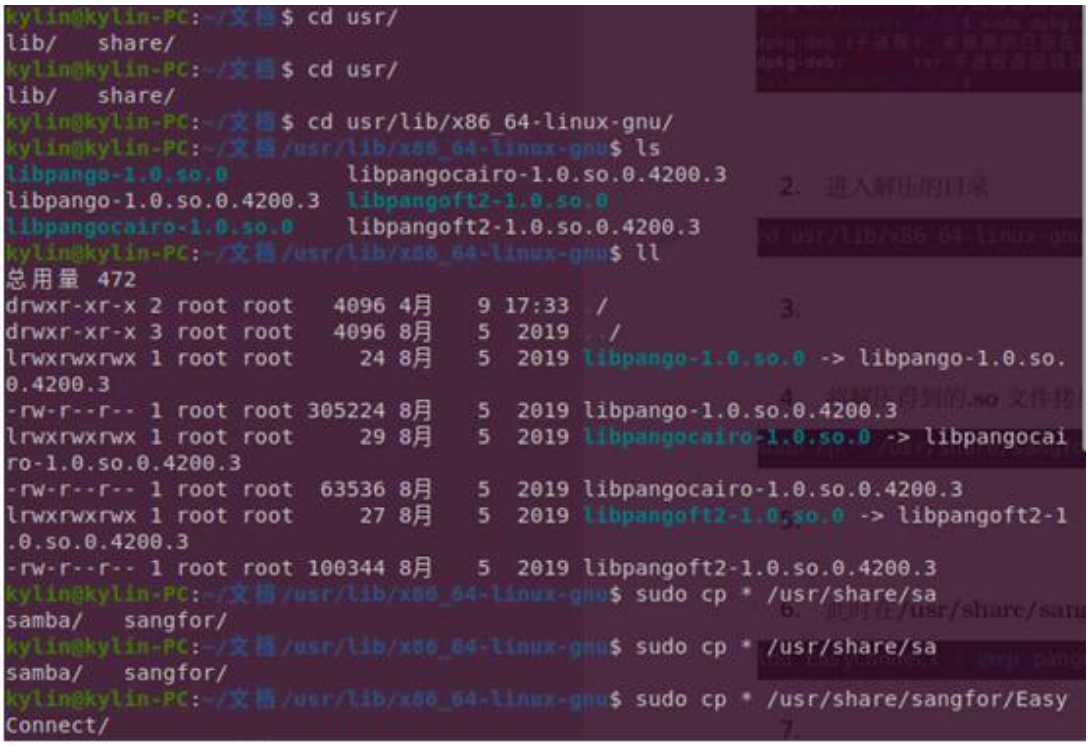
https://launchpad.net/ubuntu/+source/pango1.0/1.42.4-7/+build/17448185/+files/libpangoft2-1.0-0_1.42.4-7_arm64.deb

1) 从终端进入下载的三个包所在的目录，用命令解压下载的三个.deb 包。

```
$ sudo dpkg-deb -R libpango*.deb
```

2) 进入解压的目录。

```
$ cd usr/lib/x86_64-linux-gnu
```

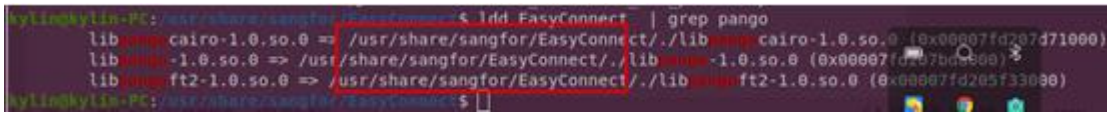


3) 将解压得到的.so 文件拷贝到 easyconnect 目录下

```
$ sudo cp * /usr/share/sangfor/EasyConnect
```

4) 此时在/usr/share/sangfor/EasyConnect 目录下输入命令：

```
$ ldd EasyConnect | grep pango
```



5) 再次打开相关软件可正常打开。

3.7.2 V10(SP1)如何将应用“固定到任务栏”

3.7.2.1 系统版本

适用系统：V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.7.2.2 问题描述

如何将应用固定到任务栏上。

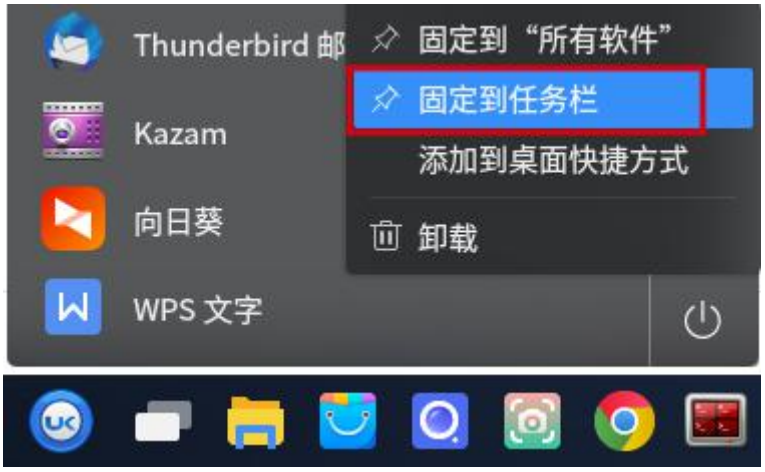
3.7.2.3 问题分析

可以用图形界面直接操作，也可以用命令行进行操作。

3.7.2.4 解决方案

1) 图形界面操作方式

点击“开始”，找到应用程序，点击“右键”，选择“固定到任务栏”即可将应用固定到任务栏。



2) 命令操作方式

a. 判断应用是否固定在任务栏上

```
dbus-send --session --type=method_call --print-reply --dest=org.ukui.panel /
com.ukui.panel.desktop.CheckIfExist
string:"/usr/share/applications/firefox.desktop"
```

b. 将应用添加到任务栏上

```
c.dbus-send --session --type=method_call --print-reply --dest=org.ukui.panel /
com.ukui.panel.desktop.AddToTaskbar
string:"/usr/share/applications/firefox.desktop"
```

c. 从任务栏取消固定

```
dbus-send --session --type=method_call --print-reply --dest=org.ukui.panel /
com.ukui.panel.desktop.RemoveFromTaskbar
string:"/usr/share/applications/firefox.desktop"
```

3.7.3 如何对已安装的应用添加“桌面快捷方式”

3.7.3.1 系统版本

适用系统：V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.7.3.2 问题描述

如何对已安装的应用添加桌面快捷方式。

3.7.3.3 问题分析

对于已有 desktop 文件的应用可以通过图形界面添加或通过命令行拷贝的方式来添加；对于没有 desktop 文件的应用需要手动编写 desktop。

3.7.3.4 解决方案

1) 应用已有 desktop 文件（几乎所有文件都有此文件）

如果应用有 desktop 文件的话，将对应的 desktop 文件拷贝到桌面上，然后给予其可执行权限，就可以了。

a. **图形界面操作方式：**

点击“开始”，找到应用程序，点击“右键”，选择“添加到桌面快捷方式”即可将应用添加桌面快捷方式。

b. **命令行方式：**

以生物特征管理工具为例，其 desktop 文件在/usr/share/applications。所以只需执行下面两条命令：

```
$ cp /usr/share/applications/biometric-manager.desktop ~/桌面
$ chmod +x ~/桌面/biometric-manager.desktop
```

2) 没有 desktop 文件（需要编写 desktop 文件）

如果应用没有 desktop 文件。则需要自己创建一个 desktop 文件在桌面，再给予可执行权限。

下面还是以生物特征管理工具为例子。创建一个如下的文件。之后给予其可执行权限即可。Name 代表程序的名字，Icon 代表显示的图标，Exec 后面跟程序启动命令，后面可以跟随参数。

```
[Desktop Entry]
Name=Biometric Manager
Name[zh_CN]=生物特征管理工具
Comment=Biometric Manager
Comment[zh_CN]=生物特征管理工具
Icon=biometric-manager
Exec=/usr/bin/biometric-manager
Keywords=Settings
Terminal=false
Type=Application
Categories=Qt;System;Utility;Core;
```

下面介绍一个 desktop 文件一些关键字的含义。

Key	描述	Value 类型	是否必须?
Type	Application (type1), Link (type2), Direcory (type3)	string	Yes
Version	版本，例如 1.1	string	No
Name	应用程序的特定名称，例如“Mozilla”。	localestring	Yes
Name[zh_CN]	应用程序的中文名称，例如“火狐浏览器”	localestring	Yes
GenericName	应用程序的通用名称，例如“Web 浏览器”。	localestring	No
NoDisplay	不在菜单中显示，但可以与 MIME 类型相关联	boolean	No
Comment	应用描述	ocalestring	No
Icon	要么绝对路径，要么符合图标主题规范	localestring	Yes
Hidden	是否隐藏，等同于不存在的文件	boolean	No
OnlyShowIn，NotShowIn	一般不用此字段	boolean	No
DBusActivatable	DBus 激活，默认 false。参阅 D-Bus 激活。应包含 Exec 行，实现兼容	boolean	No

TryExec	如果文件不存在，则忽略该文件，并不在菜单中出现	string	No
Exec	执行路径 。参阅 Exec Key	string	Yes
Path	当 Type=Application 时，程序运行的目录	string	No
Terminal	程序是否在终端窗口中运行。	boolean	No
Actions	为其他组提供接口，比如 Action=Gallery;，那么其他组就为 [Desktop Action Gallery]	string(s)	No
MimeType	此应用支持打开的类型，具体类型可以百度	string(s)	No
Categories	参阅桌面菜单规范	string(s)	No
Implements	默认情况下，桌面文件不实现任何接口。参阅接口	string(s)	No
Keywords	用于搜索，不应该是 Name 或者为多余的值 GenericName。	localestring(s)	No
StartupNotify	如果不存在，则合理的处理取决于实现（假设为 false，使用 StartupWMClass 等）。参阅启动通知协议规范	boolean	No
StartupWMClass	一般跟 Name 相同即可	string	No
URL	只适用于 Type=Link	string	No

当应用程序需要参数时，可以在 Exec 关键字后面添加参数，比如 pluma 打开文件时需要指定参数，就可以写成 Exec=pluma %U，参数会通过%U 传递给 pluma。

exec 的关键字解释见下表。

Code	描述
%f	%f 指向临时文件。用于不了解 URL 语法的程序。
%F	文件列表。用于可以一次打开多个本地文件的应用程序。每个文件作为单独的参数传递给可执行程序。
%u	单一的 URL 或者本地文件
%U	%u 的复数
%i	如果 Icon 为空，不应该填写此参数。一般也不用
%c	Name 键中的已翻译名称。
%k	桌面文件的位置 要么为 URL，要么是本地文件名，要么是为空，一般不用

命令行最多可包含一个%f，%u，%F 或%U 字段代码。如果应用程序不应打开任何文件，则必须从命令行中删除%f，%u，%F 和%U 字段代码并将其

忽略。

当需要删除快捷方式时，只需要删除该文件即可。

3.7.4 如何添加应用到右键“打开方式”

3.7.4.1 系统版本

适用系统：V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.7.4.2 问题描述

如何添加应用到“打开方式”。

3.7.4.3 问题分析

可以通过在 desktop 中修改 Exec 后面的参数和 MimeType 的类型来进行添加。

3.7.4.4 解决方案

可以使用 Exec=命令%U 添加。

其中，%U 作用是传递多个 URL，本地文件可以作为文件 URL 或作为文件路径传递。

具体以 pluma 举例子：

文件：/usr/share/applications/pluma.desktop 里面的字段 Exec=pluma %U，MimeType=text/plain，则表示 text/plain 类型的文件默认右键“打开方式”有 pluma 应用。

如果修改 Exec=pluma，则新打开的 text/plain 类型的文件默认右键“打开方式”没有 pluma 应用；

如果删除 MimeType，则新打开的 text/plain 类型的文件默认右键“打开方式”没有 pluma 应用，需要点击“更多应用”才能看到此应用。

3.7.5 如何添加应用开机自启动

3.7.5.1 系统版本

适用系统：V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.7.5.2 问题描述

如何添加应用开机自启动。

3.7.5.3 问题分析

主要有两种方法：

1) 打包时添加：系统开机进入桌面之后会执行/etc/xdg/autostart/文件中的脚本，如果想设置开机启动需要将脚本添加到/etc/xdg/autostart/目录下；

2) 通过系统设置添加。

3.7.5.4 解决方案

系统开机进入桌面之后会执行/etc/xdg/autostart/文件中的脚本，如果想设置开机启动需要将脚本添加到/etc/xdg/autostart/目录下。

1) 对于已经安装的包没有添加自启动，可以手动进行添加

如果是想设置开机启动已有的桌面应用，可以通过在“开始菜单”-“设置”-“系统”-“开机启动”，点击添加自启动程序进行添加软件自启动，这种方式属于桌面操作的方式。

或者，通过将/usr/share/applications/目录下的*.desktop 文件拷贝到/etc/xdg/autostart/目录下注销系统即可。

2) 对于第三方厂商如果想在安装软件时实现软件的开机启动，可以将*.desktop 文件同时放在/usr/share/applications/目录下和/etc/xdg/autostart/目录下。

脚本

对于非桌面应用，可以直接在/etc/xdg/autostart/目录下直接添加*.desktop 启动文件，文件中的 exec 写上脚本的绝对路径地址。

脚本示例如下：

```
$ sudo  vi  /etc/xdg/autostart/autowork.desktop
[Desktop Entry]
Name=switcher
Name[zh_CN]=自动编译及预编译
GenericName[zh_CN]=编译及预编译
Comment=Auto resume monitor mode, adjust preferred geometry.
Comment[zh_CN]=自动编译来自 gitlab 发来的合并请求
Exec=/usr/lib/autowork/autowork.sh
Terminal=false
```

```
Type=Application
Categories=System;Utility;
StartupNotify=false
X-GNOME-Autostart-Phase=Applications
X-GNOME-AutoRestart=false
X-GNOME-Autostart-Notify=false
X-KDE-autostart-after=panel
X-KDE-StartupNotify=false
```

3.7.6 系统如何安装 mysql5.7 及配置

3.7.6.1 系统版本

适用系统：V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.7.6.2 问题描述

如何在桌面系统中安装 mysql5.7 并进行配置。

3.7.6.3 问题分析

如解决方案中所示。

3.7.6.4 解决方案

1) 桌面安装 mysql5.7，推荐使用官网 Ubuntu 安装包，

a. 下载软件包，链接如下：

https://dev.mysql.com/downloads/mysql/5.7.html

b. 解压后执行 dpkg -i *.deb 报错会提示缺少依赖：

```
$ dpkg -i *.deb
```

c. 使用 apt 解决依赖关系

```
$ apt --fix-broken   install
```

d. 然后重新执行 dpkg -i *.deb：

```
$ dpkg -i *.deb
```

e. 安装后查看服务

```
$ systemctl status mysql
```

f. 进行登录测试

```
$ mysql -u root -p
```

2) 在生产环境中，设置远程访问

a. 可以配置/etc/mysql/mysql.conf.d/mysql.cnf 文件

```
$ vi /etc/mysql/mysql.conf.d/mysql.cnf
```

b. 修改 ip 地址从 127.0.0.1 → 0.0.0.0

c. 保存后重启服务

```
$ systemctl restart mysql
```

3) 配置忽略大小写

a. 配置/etc/mysql/mysql.conf.d/mysql.cnf 文件

```
$ vi /etc/mysql/mysql.conf.d/mysql.cnf
```

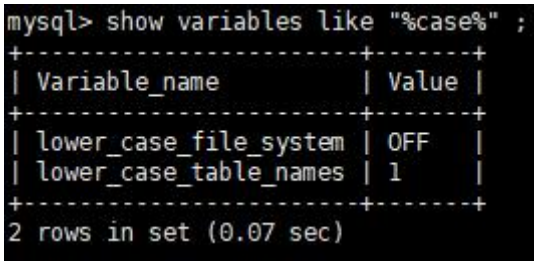
b. 在[mysqld]下添加 lower_case_table_mysql=1

c. 保存退出重启服务。

```
$ systemctl restart mysql
```

d. 然后登录数据库使用命令

```
$ show variables like '%case%';
```



value 值为 1 代表忽略大小写

4) mysql5.7 默认不支持 group by，如需修改，可以配置

/etc/mysql/mysql.conf.d/mysql.cnf 文件。

a. 打开/etc/mysql/mysql.conf.d/mysql.cnf 文件。

```
$ vi /etc/mysql/mysql.conf.d/mysql.cnf
```

b. 在[mysqld]下添加

```
sql_mode
=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIV
```

```
ISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
```

后保存退出

c. 重启服务。

```
$ systemctl restart mysql
```

3.7.7 deb 包安装后，应用图标显示异常

3.7.7.1 系统版本

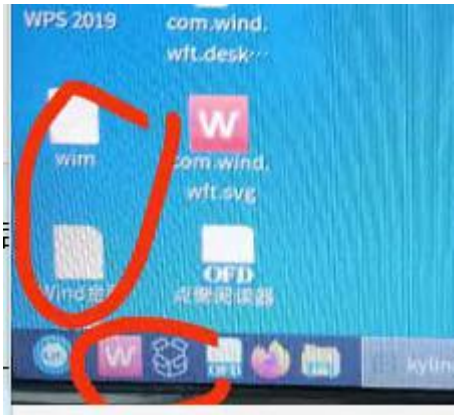
适用系统：V10

适用架构：全架构

其他版本和架构可作参考。

3.7.7.2 问题描述

第三方厂商应用安装后，应用图标显示异常，需要注销或者重启才能恢复正常。



3.7.7.3 问题分析

一般第三方应用安装后开始菜单中或任务栏或桌面快捷方式图标显示异常都是由于打包不规范导致，只需要按照打包规范重新进行打包即可。

要想知道软件包是否符合打包规范，我们可以通过 dpkg -c package.deb 来查看软件包的目录结构,此命令可以列出 package 的所有目录及文件，如果文件目录过多，不建议使用此命令。

或者通过 dpkg-deb -R package.deb package 来将 deb 包解压后查看

包的目录结构来确认软件包是否符合打包规范，建议使用此种方法。

下面以一个第三方应用为例来展示一下排查过程：

包：himirage_1.0.1.10604_amd64.deb

下载链接：https://pan.baidu.com/s/1NtQuekUyxSV2Z1ioMTAIOA

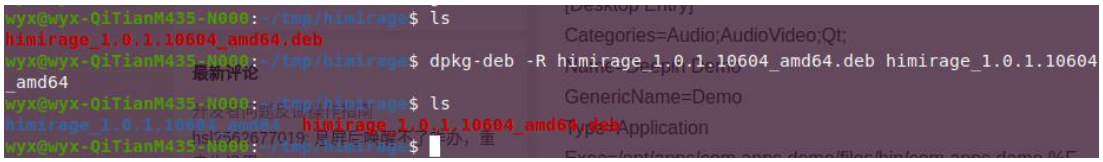
提取码：k4io

1) 解包

打开终端，进入软件包所在目录，然后执行如下命令进行解包：

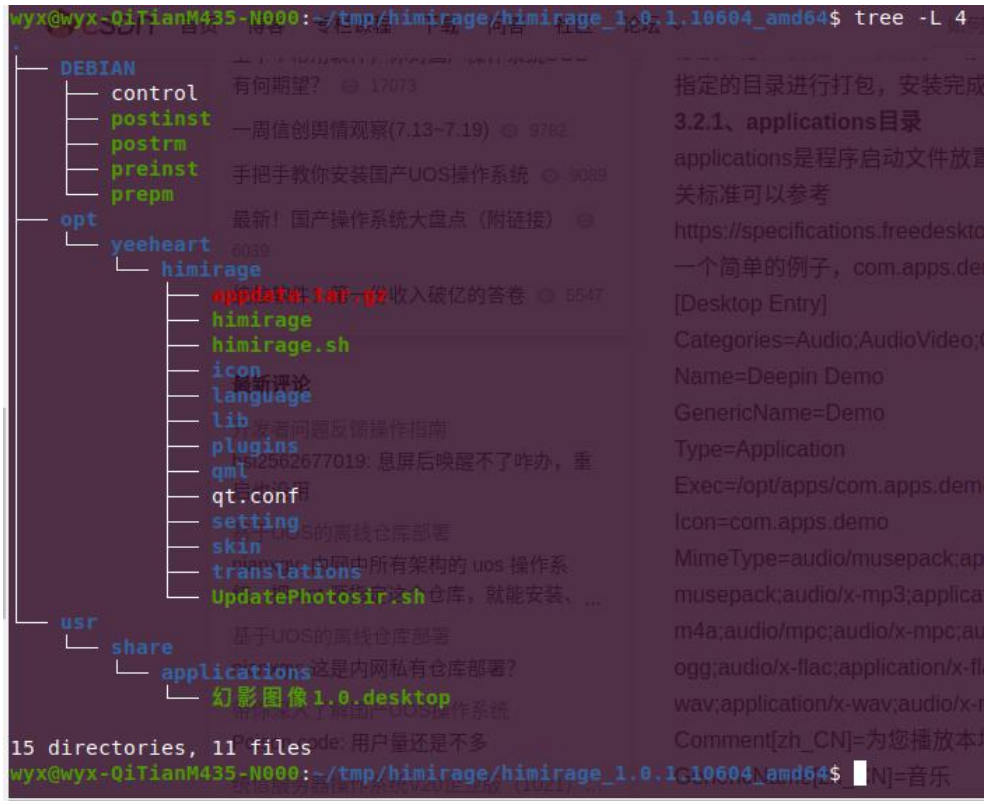
```
$ dpkg-deb -R himirage_1.0.1.10604_amd64.deb himirage_1.0.1.10604_amd64
```

解包完成后可以看到多了一个 himirage_1.0.1.10604_amd64 的目录。



2) 查看目录结构

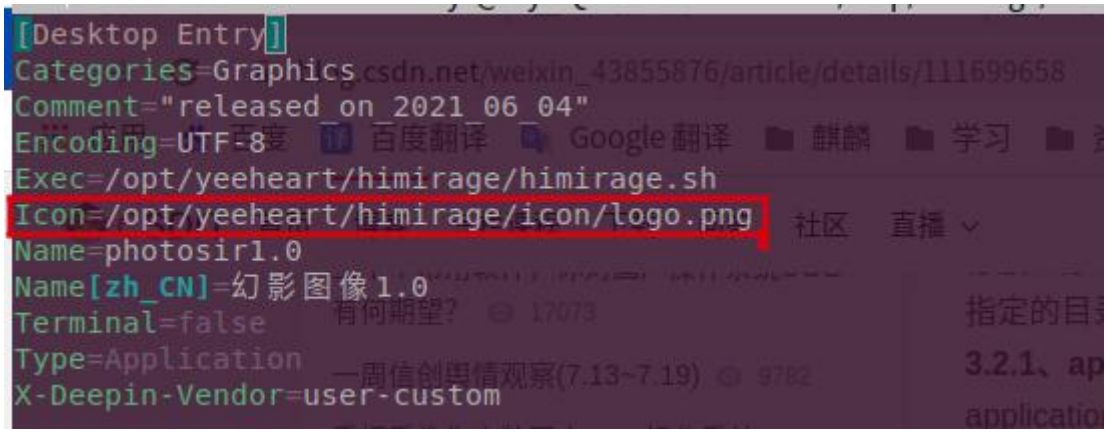
进入软件目录，使用 tree -L 4 查看 4 级目录结构或使用 tree 查看整个目录结构。



3) 查看图标放置路径是否符合打包规范

从上面查看软件目录结构可以看出，此软件包不存在/usr/share/icons/目录，所以厂商并没有按照打包规范要求进行打包。一般这种情况是厂商把 icon 图标放在了自己的软件目录（即/opt/package/目录下），此时我们需要通过查看*.desktop 文件来确定图标位置：

```
$ vim usr/share/applications/幻影图像 1.0.desktop
```



可以看出此软件包将图标放置在了自己的软件目录，desktop 中 icon 写的

是绝对路径，图标命名也不够规范，且只有一个分辨率的 png 格式的图标，这种就很容易出现图标异常的问题。

3.7.7.4 解决方案

一般由厂商来做修改：

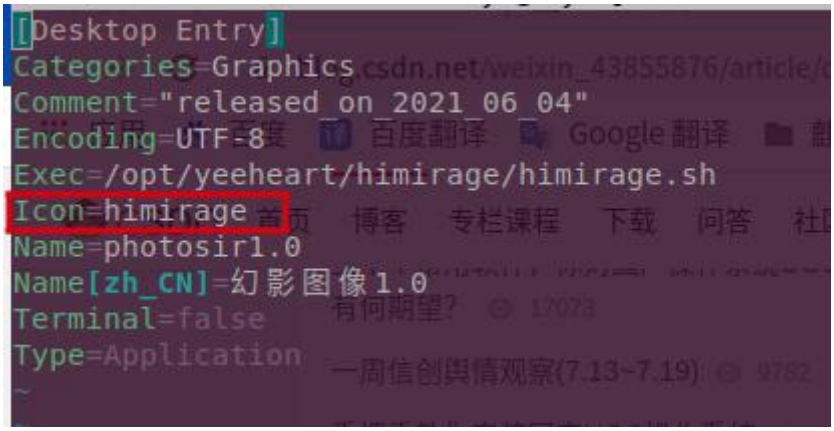
1) 找厂商要 svg 格式的矢量图标，一般都会有，如果实在没有就要全尺寸的 png 格式的非矢量图标。

2) 将从厂商要到的 svg 格式图标以应用英文名称命名。

3) 使用 mkdir -p /usr/share/icons/hicolor/scalable/apps/创建目录（如果存在请忽略），然后将图标放置在此目录下。

```
$ mkdir -p /usr/share/icons/hicolor/scalable/apps/
```

4) 修改/usr/share/applications/*.desktop 文件中 Icon 字段，后面写成步骤 2）中命名的图标名称，注意不用跟后缀。



5) 使用以下命令进行打包(不写打包名称会按照 control 文件自动进行命名打

包，很可能会出现将原包覆盖的情况)，如下：

```
$ dpkg-deb -b himirage_1.0.1.10604_amd64/ .
```

所以可以使用以下命令来重命名新包进行重新打包：

```
$ dpkg-deb -b
himirage_1.0.1.10604_amd64/ himirage_1.0.1.10604_amd64_new.deb
```

6) 打包完成后，使用以下命令安装新包来进行验证

```
$ dpkg -i himirage_1.0.1.10604_amd64_new.deb
```

注意：验证前需要将原来的包卸载，注销系统后再进行验证

```
$ sudo apt purge package（包名）
```

3.7.8 系统上如何安装多个版本的 JDK 或多个版本的 Python，并且能够快速实现切换

3.7.8.1 系统版本

适用系统：V10(SP1)

适用架构：全架构

其他版本和架构可作参考。

3.7.8.2 问题描述

系统上如何安装多个版本的 JDK 或多个版本的 python,并且能够快速实现切换。

3.7.8.3 问题分析

update-alternatives 命令用于处理 linux 系统中软件版本的切换，在各个 linux 发行版中均提供了该命令，命令参数略有区别，但大致是一样的。

3.7.8.4 解决方案

1) 注册软件（通过 apt 源安装的 jdk 已经自动注册，无需手动注册）

以 jdk 为例，安装了 jdk 以后，先要在 update-alternatives 工具中注册

```
$ sudo update-alternatives --install /usr/bin/java java
```

```
/usr/jdk1.8.0_291/bin/java 180
$ sudo update-alternatives --install /usr/bin/java java /usr/jdk-11.0.9/bin/java
1109
```

其中:

第一个参数--install 表示向 update-alternatives 注册服务名。

第二个参数是注册最终地址,成功后将会把命令在这个固定的目的地址做真实命令的软链，以后管理就是管理这个软链；

(--install link name path priority)

其中 link 为系统中功能相同软件的公共链接目录，比如/usr/bin/java(需绝对目录);name 为命令链接符名称,如 java path 为你所要使用新命令、新软件的所在目录 priority 为优先级，当命令链接已存在时，需高于当前值，因为当 alternative 为自动模式时,系统默认启用 priority 高的链接;# 整数 根据版本号设置的优先级（更改的优先级需要大于当前的）

第三个参数：服务名，以后管理时以它为关联依据。

第四个参数：被管理的命令绝对路径。

第五个参数：优先级，数字越大优先级越高。

2) 查看已注册列表

```
$ sudo update-alternatives --display java
```

3) 修改命令版本

注意--display 开关使用时第一行信息：

```
wyx@wyx-QiTianM435-N000:~/soft$ sudo update-alternatives --display java
java - 自动模式
  最佳链接版本为 /usr/jdk-11.0.9/bin/java
 链接目前指向 /usr/jdk-11.0.9/bin/java
  链接 java 指向 /usr/bin/java
/usr/jdk-11.0.9/bin/java - 优先级 1109
/usr/jdk1.8.0_291/bin/java - 优先级 180
```

下面手动修改为 jdk1.8.0_291：默认为自动版本，根据优先级，使用优先级高的。

a. 交互式修改

交互式会提示所有可用的列表，选择对应的索引确认。

修改为手动：

```
$ sudo update-alternatives --config java
```

```
wyx@wyx-QiTianM435-N000:~/soft$ sudo update-alternatives --config java
[sudo] wyx 的密码：
有 2 个候选项可用于替换 java (提供 /usr/bin/java)。

  选择       路径                          优先级  状态
------------------------------------------------------------
* 0            /usr/jdk-11.0.9/bin/java      1109      自动模式
  1            /usr/jdk-11.0.9/bin/java      1109      手动模式
  2            /usr/jdk1.8.0_291/bin/java    180       手动模式

要维持当前值[*]请按<回车键>，或者键入选择的编号： 2
update-alternatives: 使用 /usr/jdk1.8.0_291/bin/java 来在手动模式中提供 /usr/bin/java (java)
wyx@wyx-QiTianM435-N000:~/soft$ java -version
java version "1.8.0_291"
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)
```

查看模式

```
$ update-alternatives --display java
```

```
wyx@wyx-QiTianM435-N000:~/soft$ update-alternatives --display java
java - 手动模式
  最佳链接版本为 /usr/jdk-11.0.9/bin/java
 链接目前指向 /usr/jdk1.8.0_291/bin/java
  链接 java 指向 /usr/bin/java
/usr/jdk-11.0.9/bin/java - 优先级 1109
/usr/jdk1.8.0_291/bin/java - 优先级 180
```

可以看到当前状态变成了手动模式。

修改为自动：

```
$ sudo update-alternatives --auto java
```

```
wyx@wyx-QiTianM435-N000:~/soft$ sudo update-alternatives --auto java
update-alternatives: 使用 /usr/jdk-11.0.9/bin/java 来在自动模式中提供 /usr/bin/java (java)
wyx@wyx-QiTianM435-N000:~/soft$ update-alternatives --display java
java - 自动模式
  最佳链接版本为 /usr/jdk-11.0.9/bin/java
 链接目前指向 /usr/jdk-11.0.9/bin/java
  链接 java 指向 /usr/bin/java
/usr/jdk-11.0.9/bin/java - 优先级 1109
/usr/jdk1.8.0_291/bin/java - 优先级 180
```

又改为按照优先级高的了。

b. 立即修改

除了交互式修改，也可以使用一条命令直接修改，修改后立即生效。

```
$ sudo update-alternatives --set java /usr/jdk1.8.0_291/bin/java
```

```
wyx@wyx-QiTianM435-N000:~/soft$ sudo update-alternatives --set java /usr/jdk1.8.0_291/bin/java
update-alternatives: 使用 /usr/jdk1.8.0_291/bin/java 来在手动模式中提供 /usr/bin/java (java)
wyx@wyx-QiTianM435-N000:~/soft$ update-alternatives --display java
java - 手动模式
  最佳链接版本为 /usr/jdk-11.0.9/bin/java
 链接目前指向 /usr/jdk1.8.0_291/bin/java
  链接 java 指向 /usr/bin/java
/usr/jdk-11.0.9/bin/java - 优先级 1109
/usr/jdk1.8.0_291/bin/java - 优先级 180
```

该情形适用于你对路径很熟悉，或者你已经进入了该路径：

```
$ cd /usr/jdk1.8.0_291/bin/
$ sudo update-alternatives --set java $PWD/java
```

```
wyx@wyx-QiTianM435-N000:~$ cd /usr/jdk1.8.0_291/bin/
wyx@wyx-QiTianM435-N000:/usr/jdk1.8.0_291/bin$ sudo update-alternatives --set java $PWD/java
wyx@wyx-QiTianM435-N000:/usr/jdk1.8.0_291/bin$ update-alternatives --display java
java - 手动模式
  最佳链接版本为 /usr/jdk-11.0.9/bin/java
 链接目前指向 /usr/jdk1.8.0_291/bin/java
  链接 java 指向 /usr/bin/java
/usr/jdk-11.0.9/bin/java - 优先级 1109
/usr/jdk1.8.0_291/bin/java - 优先级 180
```

4) 了解一下 update-alternatives

可能你觉得这个命令很偏门，用处不大，但实际上在 linux 中早已被大量的使用。我们来挖掘一下，先看看我们注册的 java 做了什么：

```
wyx@wyx-QiTianM435-N000:~$ ls -l /usr/bin/java
lrwxrwxrwx 1 root root 22 6月   17 11:10 /usr/bin/java -> /etc/alternatives/java
wyx@wyx-QiTianM435-N000:~$ ls -l /etc/alternatives/
总用量 12
lrwxrwxrwx 1 root root  24 6月    9 09:02 aptitude -> /usr/bin/aptitude-curses
lrwxrwxrwx 1 root root  40 6月    9 09:02 aptitude.8.gz -> /usr/share/man/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  43 6月    9 09:02 aptitude.cs.8.gz -> /usr/share/man/cs/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  43 6月    9 09:02 aptitude.de.8.gz -> /usr/share/man/de/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  43 6月    9 09:02 aptitude.es.8.gz -> /usr/share/man/es/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  43 6月    9 09:02 aptitude.fi.8.gz -> /usr/share/man/fi/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  43 6月    9 09:02 aptitude.fr.8.gz -> /usr/share/man/fr/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  43 6月    9 09:02 aptitude.gl.8.gz -> /usr/share/man/gl/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  43 6月    9 09:02 aptitude.it.8.gz -> /usr/share/man/it/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  43 6月    9 09:02 aptitude.ja.8.gz -> /usr/share/man/ja/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  43 6月    9 09:02 aptitude.pl.8.gz -> /usr/share/man/pl/man8/aptitude-curses.8.gz
lrwxrwxrwx 1 root root  23 3月   27 16:47 arptables -> /usr/sbin/arptables-nft
lrwxrwxrwx 1 root root  31 3月   27 16:47 arptables-restore -> /usr/sbin/arptables-nft-restore
lrwxrwxrwx 1 root root  28 3月   27 16:47 arptables-save -> /usr/sbin/arptables-nft-save
lrwxrwxrwx 1 root root  13 3月   27 16:45 awk -> /usr/bin/gawk
lrwxrwxrwx 1 root root  29 3月   27 16:45 awk.1.gz -> /usr/share/man/man1/gawk.1.gz
lrwxrwxrwx 1 root root  29 3月   27 16:51 browser360-cn -> /usr/bin/browser360-cn-stable
```

首先可以看到我们的/usr/bin/java 是一个软链接， 它链接到/etc/alternatives 目录中的另一个软链接。而/etc/alternatives 中有很多软链，这里只列出了一部分。再看看 awk 有哪些版本：

```
wyx@wyx-QiTianM435-N000:~$ update-alternatives --display awk
awk - 自动模式
  最佳链接版本为 /usr/bin/gawk
 链接目前指向 /usr/bin/gawk
  链接 awk 指向 /usr/bin/awk
  从链接 awk.1.gz 指向 /usr/share/man/man1/awk.1.gz
  从链接 nawk 指向 /usr/bin/nawk
  从链接 nawk.1.gz 指向 /usr/share/man/man1/nawk.1.gz
/usr/bin/gawk - 优先级 10
  次要 awk.1.gz: /usr/share/man/man1/gawk.1.gz
  次要 nawk: /usr/bin/gawk
  次要 nawk.1.gz: /usr/share/man/man1/gawk.1.gz
/usr/bin/mawk - 优先级 5
  次要 awk.1.gz: /usr/share/man/man1/mawk.1.gz
  次要 nawk: /usr/bin/mawk
  次要 nawk.1.gz: /usr/share/man/man1/mawk.1.gz
```

5) 管理软件包

开始我们以 java 为例，作为 jre 运行环境可以，但如果你作为开发测试环境，你会发现 javac 找不到。

```
wyx@wyx-QiTianM435-N000:~$ javac
Could not find command-not-found database. Run 'sudo apt update' to populate it.
javac: 未找到命令
```

原因是我们只对 java 命令做了版本管理。

事实上，update-alternatives 的原理是软链管理，可以处理目录。那么

我们就可以把整个软件包目录都纳入管理。

a. 首先清理掉原来配置的 java 命令配置。

```
$ sudo update-alternatives --remove java /opt/jdk1.8.0_291/bin/java
$ sudo update-alternatives --remove java /opt/jdk-11.0.9/bin/java
```

或者可以使用，删除所有 java 配置

```
$ sudo update-alternatives --remove-all java
```

b. 注册 javahome 管理

```
wyx@wyx-QiTianM435-N000:~$ sudo update-alternatives --install /usr/local/jdk jdk /usr/jdk1.8.0_291 180
update-alternatives: 使用 /usr/jdk1.8.0_291 来在自动模式中提供 /usr/local/jdk (jdk)
wyx@wyx-QiTianM435-N000:~$ sudo update-alternatives --install /usr/local/jdk jdk /usr/jdk
jdk-11.0.9/                    jdk1.8.0_291/
jdk-11.0.9_linux-x64_bin.tar.gz  jdk-8u291-linux-x64.tar.gz
wyx@wyx-QiTianM435-N000:~$ sudo update-alternatives --install /usr/local/jdk jdk /usr/jdk
jdk-11.0.9/                    jdk1.8.0_291/
jdk-11.0.9_linux-x64_bin.tar.gz  jdk-8u291-linux-x64.tar.gz
wyx@wyx-QiTianM435-N000:~$ sudo update-alternatives --install /usr/local/jdk jdk /usr/jdk-11.0.9 1109
update-alternatives: 使用 /usr/jdk-11.0.9 来在自动模式中提供 /usr/local/jdk (jdk)
wyx@wyx-QiTianM435-N000:~$ update-alternatives --display jdk
jdk - 自动模式
  最佳链接版本为 /usr/jdk-11.0.9
 链接目前指向 /usr/jdk-11.0.9
  链接 jdk 指向 /usr/local/jdk
/usr/jdk-11.0.9 - 优先级 1109
/usr/jdk1.8.0_291 - 优先级 180
wyx@wyx-QiTianM435-N000:~$
```

配置 jdk 环境变量，指向注册的软链地址。

```
$ echo export JAVA_HOME=/usr/local/jdk >>~/.profile
$ echo export PATH=$PATH:$JAVA_HOME/bin >>~/.profile
$ source ~/.profile
```

c. 管理 JAVA_HOME

```
$ echo $JAVA_HOME
```

```
wyx@wyx-QiTianM435-N000:~$ echo $JAVA_HOME
/usr/local/jdk
wyx@wyx-QiTianM435-N000:~$ java -version
java version "11.0.9" 2020-10-20 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.9+7-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.9+7-LTS, mixed mode)
wyx@wyx-QiTianM435-N000:~$ javac -version
javac 11.0.9
```

修改 jdk

```
wyx@wyx-QiTianM435-N000:~$ sudo update-alternatives --config jdk
有 2 个候选项可用于替换 jdk (提供 /usr/local/jdk)。

  选择       路径              优先级  状态
------------------------------------------------------------
* 0            /usr/jdk-11.0.9     1109      自动模式
  1            /usr/jdk-11.0.9     1109      手动模式
  2            /usr/jdk1.8.0_291   180       手动模式

要维持当前值[*]请按<回车键>，或者键入选择的编号：2
update-alternatives: 使用 /usr/jdk1.8.0_291 来在手动模式中提供 /usr/local/jdk (jdk)
wyx@wyx-QiTianM435-N000:~$ java -version
java version "1.8.0_291"
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)
wyx@wyx-QiTianM435-N000:~$ javac -version
javac 1.8.0_291
```

3.7.9 如何解决应用运行报错 GTK+ version too old

3.7.9.1 系统版本

适用系统：V10

适用架构：X86

其他版本和架构可作参考。

3.7.9.2 问题描述

在 v10 系统使用新版的 eclipse 会报 gtk 版本低的问题，导致 open file 报错。

3.7.9.3 问题分析

通过编译新版的 gtk+来满足运行应用的环境。

3.7.9.4 解决方案

gtk 更新方法

1) gtk+版本需要更新到 3.20.0 及以上。

2) 安装 gtk+-3.20.0，以防其他程序崩溃，仍保留旧版本 gtk。

3) gtk+属于底层库，升级可能会造成系统未知问题。所以，我们将编译安装后的目录打包到软件目录，将环境变量设置在应用启动之前即可，不覆盖系统默认的 gtk。

gtk3.0 与其他相关的库关系如下图所示：

图一 GTL+ 架构

1) 下载 gtk+

下载 gtk+-3.20.0，并解压文件 gtk+-3.20.0.tar

gtk+-3.20.0 下载链接：Index of /sources/gtk+/3.20/ (gnome.org)

```
$ tar -xvf gtk+-3.20.0.tar
$ cd gtk+-3.20.0
```

2) 配置 configure

a.

```
$ ./configure -prefix=/opt/gtk
```

若编译不通过，且表明相关依赖库未找到，执行 sudo apt install libgtk-3-dev 。

b. 配置成功后，修改.bashrc 文件

```
$ vim  ~/.bashrc
```

在文件末尾添加如下内容：

```
CPPFLAGS="-I/opt/gtk/include"
LDFLAGS="-L/opt/gtk/lib"
PKG_CONFIG_PATH="/opt/gtk/lib/pkgconfig"
export CPPFLAGS LDFLAGS PKG_CONFIG_PATH
LD_LIBRARY_PATH="/opt/gtk/lib"
PATH="/opt/gtk/bin:$PATH"
export LD_LIBRARY_PATH PATH
```

c. 保存后，执行

```
$ source ~/.bashrc
```

3) 编译安装 gtk

```
$ make
```

```
$ sudo make install
```

gtk+的依赖库安装，依赖库统一安装到 /opt/gtk 路径下。

eclipse 打包场景

```
#!/bin/bash

CPPFLAGS="-I/opt/apps/eclipse-java/gtk/include"
LDFLAGS="-L/opt/apps/eclipse-java/gtk/lib"
PKG_CONFIG_PATH="/opt/apps/eclipse-java/gtk/lib/pkgconfig"
export CPPFLAGS LDFLAGS PKG_CONFIG_PATH

LD_LIBRARY_PATH="/opt/apps/eclipse-java/gtk/lib"
PATH="/opt/apps/eclipse-java/gtk/bin:$PATH"
export LD_LIBRARY_PATH PATH

cd /opt/apps/eclipse-java
./eclipse#!/bin/bash

CPPFLAGS="-I/opt/apps/eclipse-java/gtk/include"
LDFLAGS="-L/opt/apps/eclipse-java/gtk/lib"
PKG_CONFIG_PATH="/opt/apps/eclipse-java/gtk/lib/pkgconfig"
export CPPFLAGS LDFLAGS PKG_CONFIG_PATH
LD_LIBRARY_PATH="/opt/apps/eclipse-java/gtk/lib"
PATH="/opt/apps/eclipse-java/gtk/bin:$PATH"
export LD_LIBRARY_PATH PATH

cd /opt/apps/eclipse-java
./eclipse
```

目录树形结构

```
kylin@kylin-GW-001M1A-FTF:~/eclipse-java_4.20_arm64$ tree -L 4
.
├── DEBIAN
│   └── control
├── opt
│   └── apps
│       └── eclipse-java
│           ├── artifacts.xml
│           ├── configuration
│           ├── dropins
│           ├── eclipse
│           ├── eclipse.ini
│           ├── eclipse.sh
│           ├── features
```

```
│           ├── gtk
│           ├── icon.xpm
│           ├── p2
│           ├── plugins
│           └── readme
└── usr
    └── share
        ├── applications
        │   └── eclipse-java.desktop
        └── icons
            └── hicolor

16 directories, 7 files
```

3.7.10 如何使用 nvm 管理系统多个 nodejs 版本

3.7.10.1 系统版本

适用系统：V10、V10（SP1）

适用架构：X86、ARM、Loongarch

其他版本和架构可作参考。

3.7.10.2 问题描述

系统只维护了一个版本的 nodejs，部分应用需使用其它版本的 node 和 npm 来编译程序，那如何才能安装其它版本的 nodejs 和 npm，并随意在多个版本中切换呢。

3.7.10.3 问题分析

需要使用 node 工具来实现，类似工具有很多，例如 n 模块、nvm 等，本次介绍 nvm。

3.7.10.4 解决方案

1）安装 nvm

以 v0.39.3 版本为例

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

或

```
$ wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

脚本会自动把以下执行环境变量写入.bashrc，如果没有自动写入，需要手动执行：

```
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"
```

2）使用 nvm 安装 nodejs

以安装 nodejs 10.8.0 版本为例

```
$ nvm install 10.8.0
```

nvm 默认从 https://nodejs.org/dist/拉取 nodejs 的 tar 包，如果下载过慢，可以使用淘宝 node 镜像，加速 nodejs 安装过程：

```
$ export NVM_NODEJS_ORG_MIRROR=https://npmmirror.com/mirrors/node/
$ nvm install 10.11.0
```

或

```
$ NVM_NODEJS_ORG_MIRROR=https://npmmirror.com/mirrors/node/ nvm install 22.1.0
```

3）切换 nodejs 版本

使用以下命令切换到已安装的版本：

```
$ nvm use 10.8.0
```

验证当前使用的 nodejs 版本：

```
$ node -v
```

3.8 远程连接问题

3.8.1 远程连接 vncserver 连接时异常退出

3.8.1.1 系统版本

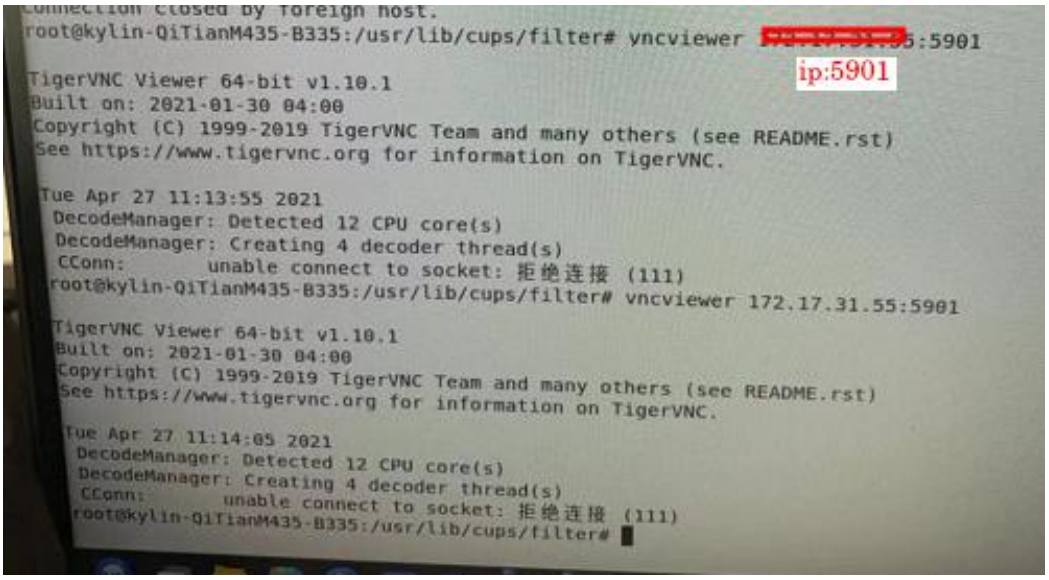
适用系统：V10(SP1)

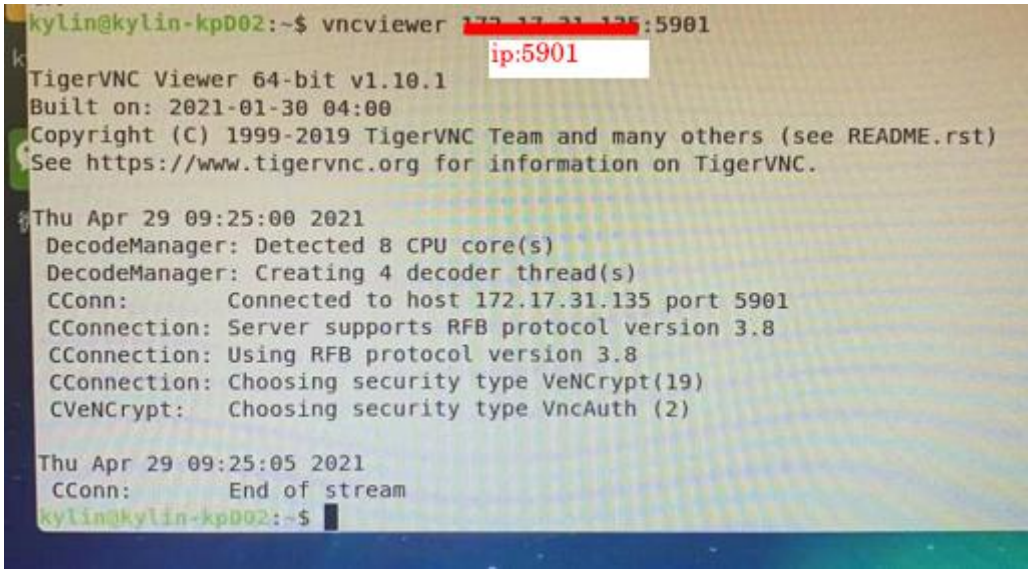
适用架构：ARM、MIPS

其他版本和架构可作参考。

3.8.1.2 问题描述

厂商反馈使用 vnc 时，一连接就报错退出。启动 vncserver 后，在客户端 vncviewer 连接，输完密码后报错无法连接，确定后退出。





3.8.1.3 问题分析

以以下具体版本为例：

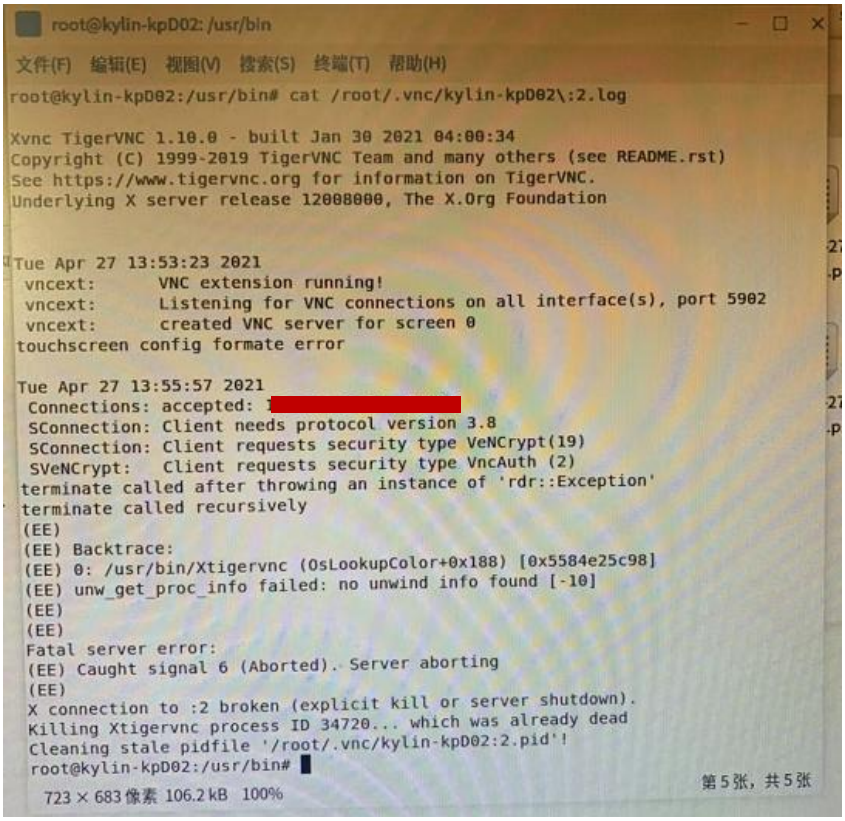
V10SP1-arm 版 ：Kylin-Desktop-V10-SP1-RC1-Build01-210326-arm64.iso

V10SP1-MIPS ：Kylin-Desktop-V10-SP1-RC1-Build01-210326-MIPS64el.iso

1） 查看 server 端，vncserver 已异常退出；

2） 查看 log：

```
$ cat /root/.vnc/kylin-kpD02\:2.log （其中 kylin-kpD02 为计算机名）
```



log 中可看到有异常抛出：rdr::Exception，应该是链接库类的问题

3.8.1.4 解决方案

启动 vncserver 时指定运行库：

1） arm64 版本系统：

LD_PRELOAD=/lib/aarch64-linux-gnu/libgcc_s.so.1 vncserver -localhost no

2） MIPS64el 版本系统:

LD_PRELOAD=/lib/MIPS64el-linux-gnuabi64/libgcc_s.so.1 vncserver -localhost no

客户端重新连接即可（vncviewer server 端 ip:5901）

3.8.2 向日葵远程无法连接图形界面

3.8.2.1 系统版本

适用系统：V10(SP1)

适用架构：X86

其他版本和架构可作参考。

3.8.2.2 问题描述

远程连接时无法启图形化界面。

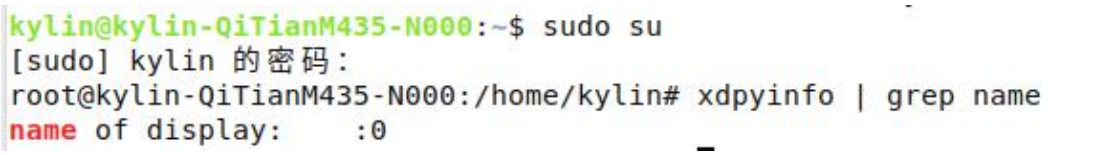
3.8.2.3 问题分析

可以使用 xhost+解决。

3.8.2.4 解决方案

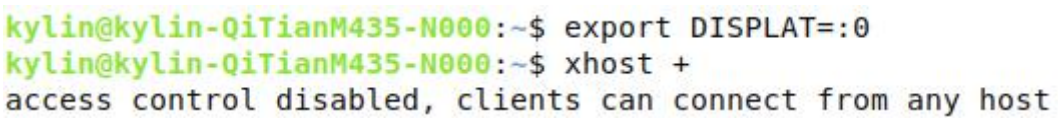
1) 先切换到 root 用户查看 DISPLAY 设置

```
$ sudo su
$ xdpyinfo | grep name
```



2) 在切换到 kylin 用户

```
$ su kylin
$ export DISPLAY=:0 （此处为 root 看到的 display 信息）
$ xhost +
```



有时使用向日葵时一连接就断开，也可以用 xhost + 解决。

4 联系我们

如有需要帮助，可联系我们：

咨询热线	400-089-1870