毕方 Linux 开发者文档

Bifang Linux developer document

2021 年 **2** 月

毕方 **Linux** 团队

目录

1.毕方系统前世今生

顾名思义 rclinux 毕方 基于 debian ，atzlinux 与 pandaos 是一个普度众生的类 debian 发行版

2019 年 9 月 atzlinux 项目创立 同年 11 月 3 日 rclinux 创立 2 月 26 日 发布 a2 基于 kali linux-light

4 月 6 日发布 future 基于 Debian10

六月，为了尽快兼容 rclinux 配套生态

提出 rclinux for arch

7 月 开发 a3 Sierra 版本

10 月 发布 rclinux arch 末代版本于 Zhousy GitHub

支持图形安装程序 archiso build

同月 13 日 Pokerface 爆出 debianrclinux 项目中的 Guiding Light-Framework 成功制作初代版本

17 日有 憨憨 测试的雏形 rclinux 安装器成功

至此 rclinux 走向完善生态之路

预计 21 年 1 月成功 完善生态

2.毕方 Linux 哲学

我们的 Linux 发行版 是秀气 质朴 与超度的

我们 一向如此 我们不喜欢杠精

但当你步入毕方 Linux 的殿堂 请务必 庄严而神圣

我们会带领你布置自己想要的模样

我们的哲学，保持“简略”。抱负上，也便是“坚持简略”。启示者们但愿由于用庞大的体例做简略的使命铺张少量工夫。

做你自己吧 我们们会永不停息陪伴着你

2.毕方 Linux 测试教程

当你被这个季度的负责人选中参加 rclinux 测试时，很有可能是测试指明灯安装框架。

请听从指挥参与测试 首先找测试部负责人（现为 ixcm）要到 tarball 与引导系统盘与指明灯安装器

然后请您务必将 tarball 放入！！！！！！！镜像 删除删除镜像内原有的“rclinuxinstall.sh” ！！！！！

换上新的脚本！！！！！！！！！！！！

然后保存镜像 按照脚本顺序安装测试

建议使用 Virualbox！！！！！！！

3.毕方 Linux 打包教程

Guiding Light Framework

本为 rclinux 最重要的一环

如果你想使用 rclinuxinstall

请先保证您是 grub 引导的类 unix/unix 系统。（理论上支持全架构，全系统）

所以，rclinux 的安装框架十分的棒

但是 RBBinstaller 框架仍有不足

对分区的极大不支持，强制性 mbr 啥的

但是这都会在下一版中获得原 rclinuxinstall 带来的升级

首先，是备份 tarball 的方法

首先成为 root 用户：

$ sudo su

然后进入文件系统的根目录(当然，如果你不想备份整个文件系统，你也可以进入你想要备份的目录，包括远程目录或者移动硬盘上的目录)：

cd /

你可以进入 live 系统 mount /dev/sda1 /mnt

chroot /mnt /bin/bash 如果打错了软件包 我们应该 mv /target/rc/mnt* //target/rc/

然后再输入命令 （很难卡在/proc/kcore）

下面是我用来备份系统的完整命令：

tar cvpzf backup.tgz –exclude=/proc –exclude=/lost+found –exclude=/backup.tgz –exclude=/mnt –exclude=/sys /

然后您就制作了 tarball

默认拿 rclinux 的 iso 引导，

我先讲讲这个请您把 tarball 放进去，遵循 rclinux 命名方法为 backup.tgz 即可

您可以 mount/使用 ultraiso（windows）

或者使用 Linux 下图形界面进行制作

加备份共约莫 20 分钟

这将极大帮助开发者维护 linux 与用户的备份与对操作系统的自定义

Guiding Light

Guiding Light Framework 本为 rclinux 最重要的一环 如果你想使用 rclinuxinstall 请先保证您是 grub 引导的类 unix/unix 系统。（理论上支持全架构，全系统） 所以，rclinux 的安装框架十分的棒

$ sudo su 然后进入文件系统的根目录(当然，如果你不想备份整个文件系统，你也可以进入你想要备份的目录，包括远程目录或者移动硬盘上的目录)：

在毕方 Linux livecd 下打包

请进入 rclinux 的安装镜像（arch 引导的 然后

mount /dev/sda1 /mnt

/dev/sda1 即安装的系统根目录

cd /mnt

sudo chroot /mnt /bin/bash

完整命令如下：

tar cvpzf backup.tgz —exclude=/proc —exclude=/lost+found —exclude=/backup.tgz —exclude=/mnt —exclude=/sys /

exit

由于本系统打包时要清除用户信息 不要安装虚拟增强工具 虚拟机端口转发 ssh/ftp 连接到根目录获取 backup.tgz 或者 u 盘直接挂载 然后您就制作了您的系统默认拿 rclinux 的 iso 引导， 我先讲讲这个请您把 tarball 放进去，遵循 rclinux 命名方法为 backup.tgz 即可 您可以 mount/使用 ultraiso（windows） 或者使用 Linux 下图形界面进行对毕方 Linux 安装程序挂载 删除内部原有 backup.tgz 替换为 你的 backup.tgz 即可 完成

毕方 Linux 本地系统打包

cd /

tar cvpzf backup.tgz —exclude=/proc —exclude=/lost+found —exclude=/backup.tgz —exclude=/mnt —exclude=/sys —exclude=/media /

—warning=no-file-change

FAQ：

为啥要用指定的镜像引导？

因为我对那些镜像系统进行了修改，方便引导

没了！哈哈哈哈

自行打包要求

到清华安装 rclinux-system.zip

装壁纸

编辑 os-relaase lsb- 申明版权

可以加减软件 配置编译 内核 我方桌面环境

（2.）squashfs 制作

一、安装需要使用的软件

$sudo apt-get install squashfs-tools genisoimage //装入 squashfs 文件系统处理工具以及 ISO 制作工具

$sudo modprobe squashfs //加载 squashfs 模块

二、提取光盘内容

1、创建一个 livecd 文件夹作为工作目录

mkdir ~/livecd

2、挂载 ubuntu 光盘镜像

cd ~/livecd

mkdir mnt

3.毕方 Linux 打包教程

哎 要干些什么

在你使用指明灯打包后具体看 2.

之后如果你安装了普通的 Debian/铜豌豆 Linux

你先复制之前的/home 配置文件来

sudo mount -o loop ubuntu-14.04-desktop-i386.iso mnt

3、展开 ubuntu 镜像文件中的内容到 mycd 目录

mkdir mycd

sudo cp -Ra mnt/* mycd

三、解压 squashfs 系统文件

1、挂载 squashfs 文件系统

mkdir squashfs

sudo mount -t squashfs -o loop mnt/casper/filesystem.squashfs squashfs

2、展开 squashfs 文件系统中的内容到 myedit 目录下

mkdir myedit

sudo cp -Ra squashfs/* myedit/(这一步需要较长的时间)

四、配置设置自己的系统

1、复制 resolv.conf 到系统目录

sudo cp /etc/resolv.conf myedit/etc/

2、hosts 复制一份过去

sudo cp /etc/hosts myedit/etc/

3、替换 apt 源

rm livecd/myedit/etc/apt/sources.list

cp /etc/apt/source.list livecd/myedit/etc/apt

apt-get update

4、在自己系统上挂载一些重要的目录

sudo mount --bind /dev/ myedit/dev

sudo chroot myedit

mount -t proc none /proc

mount -t sysfs none /sys

mount -t devpts none /dev/pts

五、安装卸载系统中的软件，配置以达到自己的要求

1、安装卸载软件

cd ~livecd/

sudo chroot myedit

sudo apt-get install 软件名

sudo apt-get remove 软件名

2、进行清理

sudo apt-get autoremove

sudo apt-get clean

3、记得卸载刚才加入的挂载，退出 chroot 环境

umount /proc

umount /sys

umount /dev/pts

exit

sudo umount myedit/dev

六、生成自己的 ubuntu 系统镜像 playubuntu.iso

1、重新压缩系统文件到 filesystem.squashfs

sudo rm mycd/casper/filesystem.squashfs //删除原有的 filesystem.squashfs

sudo mksquashfs myedit mycd/casper/filesystem.squashfs //生成自己的 filesystem.squashfs

2、创建 playubuntu.iso

cd mycd

sudo mkisofs -D -r -V “$IMAGE_NAME” -cache-inodes -J -l -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../playubuntu.iso ~/livecd/mycd/ (这是一条命令)

七、完成后我们可以在 virtualbox 虚拟机里测试，很有成就感的

1、安装 virtualbox

sudo apt-get install virtualbox

（3）livecd 制作 Debian

Debian 制作 LiveCD

2020-06-22 | Linux

制作前安装软件

1

2

3

4

5

6

7

8

9

```
# 在构建主机上安装如下软件
sudo apt install debootstrap squashfs-tools linux-image-amd64 live-boot grub-efi


# 上述软件提供的功能或作用
debootstrap          # 获取 Debian 的根文件系统
squashfs-tools       # 压缩及解压 squashfs 文件
linux-image-amd64    # 提供所需内核文件
live-boot            # 生成支持 LiveCD 的 initramfs
grub-efi             # 提供 EFI 的启动引导
```

配置启动的环境

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

```
# 获取根文件系统
sudo su
mkdir /opt/debian/
debootstrap --arch=amd64 buster /opt/debian/ http://ftp2.cn.debian.org/debian

# 进入 chroot 环境后配置启动环境
chroot /opt/debian/ /bin/bash -l
```

```
# 安装额外的软件
apt install sudo htop fdisk pciutils usbutils alsa-utils
apt install wireless-tools wpasupplicant bluez bluez-tools

# 安装内核和固件驱动(根据你的硬件选择固件驱动)
apt install linux-image-amd64 firmware-atheros firmware-iwlwifi
apt install firmware-misc-nonfree firmware-linux-nonfree
apt install firmware-realtek firmware-ralink firmware-brcm80211
apt install firmware-intel-sound firmware-amd-graphics nvidia-settings

# 添加一个用户并赋予 sudo 权限
useradd -m -g sudo -s /bin/bash debian
passwd debian
echo "debian    ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
chown -R debian:sudo /home/debian

# 设置 DNS 和软件源
echo "nameserver 127.0.0.1" > /etc/resolv.conf
echo "nameserver 114.114.114.114" >> /etc/resolv.conf
echo "deb http://ftp2.cn.debian.org/debian/ buster main non-free contrib" \
> /etc/apt/sources.list

# 设置主机名和时区
echo Debian > /etc/hostname
sed -i '/localhost/s/$/\t'"Debian"'/g' /etc/hosts
cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
```

清理所有已缓存的包后退出 chroot 环境

apt clean

exit

rm /opt/debian/root/.bash_history

启动时必须文件

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

```
# 挂载 GPT 分区表 U 盘的 Fat32 分区并复制下面的文件到其中
mount /dev/sdb1 /mnt
mkdir -p /mnt/efi/boot/
```

```
# 生成 squashfs 根文件系统和内核到 U 盘
mksquashfs /opt/debian/ /mnt/live/filesystem.squashfs
cp /boot/vmlinuz-4.19.0-9-amd64 /mnt/live/

# 制作 initrd 文件，4.19.0-9-amd64 为/lib/modules 下的内核模块目录名称
mkinitramfs 4.19.0-9-amd64 -o /mnt/live/initrd.img-4.19.0-9-amd64

# 生成 GRUB 的 EFI 单引导文件
echo -e "configfile (\$root)/efi/boot/grub.cfg" > grub.cfg

grub-mkimage -d /usr/lib/grub/x86_64-efi/ -c grub.cfg -p /efi/boot \
-o /mnt/efi/boot/bootx64.efi -O x86_64-efi boot linux linux16 chain \
loopback net search disk part_gpt part_msdos disk blocklist cpio btrfs \
exfat fat ext2 hfs hfsplus iso9660 jfs ntfs procfs squash4 xfs zfs tar \
date echo ls configfile normal file sleep true minicmd play read acpi \
reboot halt efifwsetup efi_gop efi_uga video_bochs video_cirrus \
gfxmenu gfxterm gfxterm_background gfxterm_menu font jpeg png

# 写入 GRUB 的引导配置
cat << EOF > /mnt/efi/boot/grub.cfg
set timeout=3
set menu_color_normal=cyan/blue
set menu_color_highlight=white/blue

menuentry "Debian GNU/Linux Live (kernel 4.19.0-9-amd64)" {
  linux  /live/vmlinuz-4.19.0-9-amd64 boot=live components quiet
  initrd /live/initrd.img-4.19.0-9-amd64
}
EOF
```

优盘中现在的目录及文件

leux@K680:/mnt$ tree

.

├── efi

│ └── boot

│ ├── bootx64.efi

│ └── grub.cfg

└── live

├── filesystem.squashfs

├── initrd.img-4.19.0-9-amd64

└── vmlinuz-4.19.0-9-amd64

3 directories, 5 files

现在就可以直接从 U 盘启动 LiveCD 了

部署到磁盘启动(可选)

由于 squashfs 文件系统无法保存任何更改，也可直接部署到磁盘来启动

如果要安装系统的磁盘还未分区和格式化，先分区并格式化为指定格式

1

2

3

4

5

6

7

leux@K680:/mnt$ sudo fdisk /dev/sdb

Device Start End Sectors Size Type

/dev/sdb1 2048 207000 204953 100.1M Fat32

/dev/sdb2 208896 20971486 20762591 9.9G Ext4

mkfs.vfat /dev/sdb1

mkfs.ext4 /dev/sdb2

挂载要安装系统的磁盘到指定目录并复制文件到其中

1

2

3

4

5

6

mkdir -p /media/boot/ /media/root/

mount /dev/sdb1 /media/boot/

mount /dev/sdb2 /media/root/

unsquashfs -d /media/squashfs/ /mnt/live/filesystem.squashfs

cp -r /media/squashfs/* /media/root/

cp -r /mnt/efi/boot/bootx64.efi /media/boot/bootx64.efi

获取磁盘的 UUID

1

2

3

4

5

leux@K680:/mnt$ lsblk -f

NAME FSTYPE LABEL UUID FSAVAIL FSUSE% MOUNTPOINT

sda

|-sda1 vfat boot 9A46-9516 100.1M 2% /boot/efi

`-sda2 ext4 root 004e6e14-b1bd-384b-84a5-93d03fdcf964 9.9G 7% /

设置开机自动挂载磁盘

1

2

3

```
# /media/root/fstab
UUID=9A46-9516 /boot/efi vfat defaults 0 1
UUID=004e6e14-b1bd-384b-84a5-93d03fdcf964 / ext4 discard,noatime 0 1
```

配置 GRUB 的启动参数

1

2

3

4

5

6

7

8

9

10

```
# /media/boot/efi/boot/grub.cfg
set timeout=1
set menu_color_normal=cyan/blue
set menu_color_highlight=white/blue

menuentry "Debian GNU/Linux Buster (kernel 4.19.0-9-amd64)" {
  search --no-floppy --fs-uuid --set=root 004e6e14-b1bd-384b-84a5-93d03fdcf964
  linux  /vmlinuz root=UUID=004e6e14-b1bd-384b-84a5-93d03fdcf964 quiet
  initrd /initrd.img
}
```

其他说明

本文的构建是在 Debian Buster WSL 下进行的，其他环境请自测

Debian 的 LiveCD 内核是在压缩文件外面。如果你的 LiveCD 已经完美驱动了您的硬件，那么你可以不用安装内核到 rootfs，否则安装内核可以解决多数驱动问题

LiveCD 是不需要配置/etc/fstab 的，设置了可能会启动出错

如果 LiveCD 无法连接网络，驱动等正常的话请执行 dhclient 获取 DHCP 地址试试

4.毕方 Linux 安卓软件配置

Anbox 手动安装 ARM 兼容库

2020-06-10 | Linux

环境配置说明

测试系统：Debian Buster AMD64

参考借鉴：Anbox 上安装 Google Play 商店及启用 ARM 支持 | Anbox 开启 ARM 兼容脚本

检查所需模块

从 Ubuntu 19.04 开始，自带的内核(>= 5.0)已包含上面两个模块，不再需要从 PPA 安装模块。

从 Debian 10 开始，自带的内核(>= 4.19)已包含上面两个模块，不再需要从 DKMS 安装模块。

1

2

3

4

5

6

7

8

测试加载内核模块

sudo modprobe ashmem_linux

sudo modprobe binder_linux

查看加载是否成功

ls -1 /dev/{ashmem,binder}

/dev/ashmem

/dev/binder

安装配置 Anbox

安装 Anbox

1

2

注意：Anbox 在 contrib 源里，找不到包则需要添加 contrib 源

sudo apt install anbox

下载 Android 镜像

1

2

wget https://build.anbox.io/android-images/2018/07/19/android_amd64.img

cp android_amd64.img /var/lib/anbox/android.img

启动 Anbox 容器管理器

1

2

3

4

5

6

7

8

启动 Anbox 容器管理器

sudo systemctl start anbox-container-manager

启动 Session 管理器

/usr/bin/anbox session-manager

启动 Anbox 应用管理界面

/usr/bin/anbox launch --package=org.anbox.appmgr --component=org.anbox.appmgr.AppViewActivity

启用 ARM 兼容

可以使用脚本来自动启用 ARM 兼容，如使用脚本就不需要执行后面步骤了

1

2

3

4

sudo apt install wget lzip unzip squashfs-tools

wget https://raw.githubusercontent.com/geeks-r-us/anbox-playstore-installer/master/install-houdini-only.sh

chmod +x install-playstore.sh

sudo ./install-playstore.sh

下载兼容库和准备镜像

1

2

3

4

5

6

houdini 中 7 代表安卓版本号，x 是 32 位系统用的支持 32 位程序的库

y 是 64 位系统用的支持 32 位程序的库，z 是 64 位系统用的支持 64 位的库

wget http://dl.android-x86.org/houdini/7_y/houdini.sfs -O houdini_y.sfs

wget http://dl.android-x86.org/houdini/7_z/houdini.sfs -O houdini_z.sfs

cp /var/lib/anbox/android.img ./

unsquashfs android.img

将兼容库复制到解包后的 Android 镜像内

1

2

3

4

5

6

7

8

9

10

11

12

13

14

为了保证文件的权限正常，后续操作全部以 ROOT 权限执行

32 位库放到/system/lib/arm/下

unsquashfs -f -d houdini_y houdini_y.sfs

mkdir squashfs-root/system/lib/arm

cp -r ./houdini_y/* squashfs-root/system/lib/arm

chown -R 100000:100000 squashfs-root/system/lib/arm

cp squashfs-root/system/lib/arm/libhoudini.so squashfs-root/system/lib/libhoudini.so

64 位库放到/system/lib64/arm64/下

unsquashfs -f -d houdini_z houdini_z.sfs

mkdir squashfs-root/system/lib64/arm64

cp -r ./houdini_z/* squashfs-root/system/lib64/arm64/

chown -R 100000:100000 squashfs-root/system/lib64/arm64

cp squashfs-root/system/lib64/arm64/libhoudini.so squashfs-root/system/lib64/libhoudini.so

设置将 ARM 的程序通过 houdini 来运行

1

2

3

4

5

通过文件开始位置的特殊的字节来判断是否是 ARM 程序，是的话将其使用 houdini 来运行

echo ':arm_exe:M::\x7f\x45\x4c\x46\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x28::/system/lib/arm/houdini:P' | tee -a /proc/sys/fs/binfmt_misc/register

echo ':arm_dyn:M::\x7f\x45\x4c\x46\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x03\x00\x28::/system/lib/arm/houdini:P' | tee -a /proc/sys/fs/binfmt_misc/register

echo ':arm64_exe:M::\x7f\x45\x4c\x46\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\xb7::/system/lib64/arm64/houdini64:P' | tee -a /proc/sys/fs/binfmt_misc/register

echo ':arm64_dyn:M::\x7f\x45\x4c\x46\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x03\x00\xb7::/system/lib64/arm64/houdini64:P' | tee -a /proc/sys/fs/binfmt_misc/register

修改 default.prop 和 system/build.prop 文件

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

```
# 在 default.prop 中的 `ro.zygote=zygote64_32` 下面添加一行：
ro.dalvik.vm.native.bridge=libhoudini.so

sed -i '/ro.zygote=zygote64_32/a\ro.dalvik.vm.native.bridge=libhoudini.so'
default.prop


# system/build.prop 的修改

sed -i "/^ro.product.cpu.abilist=x86_64,x86/ s/$/,armeabi-v7a,armeabi,arm64-v8a/"
system/build.prop

sed -i "/^ro.product.cpu.abilist32=x86/ s/$/,armeabi-v7a,armeabi/"
system/build.prop

sed -i "/^ro.product.cpu.abilist64=x86_64/ s/$/,arm64-v8a/" system/build.prop

echo "persist.sys.nativebridge=1" | sudo tee -a system/build.prop

echo "ro.opengles.version=131072" | sudo tee -a system/build.prop
```

system/build.prop 修改前

ro.product.cpu.abilist=x86_64,x86

ro.product.cpu.abilist32=x86

ro.product.cpu.abilist64=x86_64

system/build.prop 修改后

ro.product.cpu.abilist=x86_64,x86,armeabi-v7a,armeabi,arm64-v8a

ro.product.cpu.abilist32=x86,armeabi-v7a,armeabi

ro.product.cpu.abilist64=x86_64,arm64-v8a

在最后面添加下面两行

persist.sys.nativebridge=1

ro.opengles.version=131072

修改 system/etc/permissions/anbox.xml 文件

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

删除下面两句

```
    <unavailable-feature name="android.hardware.wifi" />
    <unavailable-feature name="android.hardware.bluetooth" />
```

添加下面几句

```
        <feature name="android.hardware.touchscreen" />
        <feature name="android.hardware.audio.output" />
        <feature name="android.hardware.camera" />
        <feature name="android.hardware.camera.any" />
        <feature name="android.hardware.location" />
        <feature name="android.hardware.location.gps" />
        <feature name="android.hardware.location.network" />
        <feature name="android.hardware.microphone" />
        <feature name="android.hardware.screen.portrait" />
        <feature name="android.hardware.screen.landscape" />
        <feature name="android.hardware.wifi" />
        <feature name="android.hardware.bluetooth" />"
```

将修改后的镜像打包并替换原版镜像

1

2

3

4

```
mksquashfs squashfs-root android.img
rm /var/lib/anbox/android.img
cp android.img /var/lib/anbox/android.img
systemctl restart anbox-container-manager
```

其他说明

在 Session manager 启动后，可通过 lxc-attach 来进入 Abox 的 Shell 环境

1
2
3
4
5
6
7
8
9
10
11

```
sudo lxc-attach -q \
--clear-env \
-P /var/lib/anbox/containers \
-n default \
-v PATH=/sbin:/system/bin:/system/sbin:/system/xbin \
-v ANDROID_ASSETS=/assets \
-v ANDROID_DATA=/data \
-v ANDROID_ROOT=/system \
-v ANDROID_STORAGE=/storage \
-v ASEC_MOUNTPOINT=/mnt/asec \
-v EXTERNAL_STORAGE=/sdcard -- /system/bin/sh
```

在 Anbox 中安装应用

1
2
3
4

5

6

7

8

sudo apt install android-tools-adb

ABD 基本操作

adb devices # 查看 Android 设备

adb install xxx.apk # 安装 APP 到 Anbox

adb uninstall com.xx.xx # 从设备中卸载 APP

adb shell # 进入 Android 交互

adb shell pm list packages # 列出已安装的包

Anbox 在 ARM64 架构 Linux 上运行 Android 程序

2020-06-09 | Linux

环境配置说明

适用硬件：Raspberry Pi 4B | raspios_arm64-2020-05-28

编译系统：Debian For WSL

参考借鉴：UOS 安装 anbox | Anbox 的安装编译

交叉编译内核

某些发行版内核已自带了模块，但树莓派的内核不包含它们且不支持 DKMS 只能手动编译

从 Ubuntu 19.04 开始，自带的内核(>= 5.0)已包含上面两个模块，不再需要从 PPA 安装模块。

从 Debian 10 开始，自带的内核(>= 4.19)已包含上面两个模块，不再需要从 DKMS 安装模块。

在编译主机上安装交叉编译工具

1

2

sudo apt-get update

sudo apt install git bc bison flex libssl-dev make libncurses-dev gcc-aarch64-linux-gnu

获取内核源码

1

2

3

mkdir /home/leux/rpi4

cd /home/leux/rpi4

git clone -b rpi-5.4.y --depth=1 https://github.com/raspberrypi/linux.git

为内核源码打补丁，使其可生成 ashmem_linux 和 binder_linux 模块

1

2

3

4

5

wget https://salsa.debian.org/kernel-team/linux/-/blob/debian/5.4.19-1/debian/patches/debian/android-enable-building-ashmem-and-binder-as-modules.patch

wget https://salsa.debian.org/kernel-team/linux/-/blob/debian/5.4.19-1/debian/patches/debian/export-symbols-needed-by-android-drivers.patch

cd linux

patch -p1 < ../android-enable-building-ashmem-and-binder-as-modules.patch

patch -p1 < ../export-symbols-needed-by-android-drivers.patch

获取内核编译配置文件

1

2

3

4

5

6

```
# 在 64 位的 Raspbian 系统内执行下面命令来获取官方的内核编译配置文件
sudo modprobe configs
zcat /proc/config.gz > ~/raspi4b.config

# 将上面的 raspi4b.config 拷贝到内核源码下
cp raspi4b.config /home/leux/rpi4/linux/.config
```

配置内核编译参数

1

2

3

4

5

6

7

8

9

10

11

12

```
# 进入内核配置界面
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- menuconfig

# 要运行所必须的内核参数
CONFIG_ANDROID=y
CONFIG_ANDROID_BINDER_IPC=m
CONFIG_ANDROID_BINDER_DEVICES="binder,hwbinder,vndbinder"
```

CONFIG_ASHMEM=m

在内核配置界面里的位置

Device Drivers > Android > Android Drivers > Android Binder IPC Driver

Device Drivers > Staging drivers > Android > Enable the Anonymous Shared Memory Subsystem

编译内核并备份

1

2

3

4

5

6

7

8

9

10

11

12

编译内核

make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- -j12

安装模块到临时文件夹

make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- INSTALL_MOD_PATH=/home/leux/rpi4/kernel/ modules_install

安装内核到临时文件夹

cp arch/arm64/boot/Image ../kernel/kernel-kvm.img

备份生成的内核到 kernel.tgz

cd /home/leux/rpi4

tar -czvf kernel.tgz kernel/

安装内核到系统

1

2

3

4

5

把上步生成的 kernel.tgz 复制到树莓派里

tar -xzvf kernel.tgz && cd kernel

mv kernel-kvm.img /boot/

mv lib/modules/* /lib/modules/

echo "kernel=kernel-kvm.img" >> /boot/config.txt

安装完内核后重启来加载新内核，然后查看模块效果

1

2

3

4

5

6

7

8

测试加载内核模块

sudo modprobe ashmem_linux

sudo modprobe binder_linux

查看模块效果

ls -1 /dev/{ashmem,binder}

/dev/ashmem

/dev/binder

安装配置 Anbox

安装 Anbox

1

2

注意：Anbox 在 contrib 源里，找不到包则需要添加 contrib 源

sudo apt install anbox

下载 Android 镜像

1

2

3

4

5

6

7

8

9

10

11

官方提供的 arm64 镜像不能正常启动，但可以使用别人已经编译好的

链接: https://pan.baidu.com/s/1QDaDtCi4MIMLbQyc-Ov0UA 提取码: w6av

https://build.anbox.io/android-images/2017/06/12/android_1_armhf.img

https://build.anbox.io/android-images/2017/08/04/android_1_arm64.img

https://build.anbox.io/android-images/2018/07/19/android_amd64.img

也可使用 Ubuntu Touch 项目提供的 Anbox-images 镜像

https://cdimage.ubports.com/anbox-images/android-armhf-32binder.img

https://cdimage.ubports.com/anbox-images/android-armhf-64binder.img

cp android_1_arm64.img /var/lib/anbox/android.img

启动 Anbox 容器管理器

1

2

3

4

5

6

7

8

启动 Anbox 容器管理器

sudo systemctl start anbox-container-manager

启动 Session 管理器

/usr/bin/anbox session-manager

启动 Anbox 应用管理界面

/usr/bin/anbox launch --package=org.anbox.appmgr --component=org.anbox.appmgr.AppViewActivity

其他说明

在 Session manager 启动后，可通过 lxc-attach 来进入 Abox 的 Shell 环境

1

2

3

4

5

6

7

8

9

10

11

```
sudo lxc-attach -q \
--clear-env \
-P /var/lib/anbox/containers \
-n default \
-v PATH=/sbin:/system/bin:/system/sbin:/system/xbin \
-v ANDROID_ASSETS=/assets \
-v ANDROID_DATA=/data \
-v ANDROID_ROOT=/system \
-v ANDROID_STORAGE=/storage \
-v ASEC_MOUNTPOINT=/mnt/asec \
-v EXTERNAL_STORAGE=/sdcard -- /system/bin/sh
```

在 Anbox 中安装应用

1

2

3

4

5

6

7

8

9

10

11

```
sudo apt install android-tools-adb
```

ABD 基本操作

adb devices　　# 查看 Android 设备

adb install xxx.apk　　# 安装 APP 到 Anbox

adb uninstall com.xx.xx　　# 从设备中卸载 APP

adb shell　　# 进入 Android 交互

adb shell pm list packages　　# 列出已安装的包

只能安装 arm64-v8a 的应用，不兼容 armeabi-v7a 或 armeabi 的应用

adb: failed to install xx.apk: Failure [INSTALL_FAILED_NO_MATCHING_ABIS: Failed to extract native libraries, res=-113]

最近比较闲，定了一个任务就是学习 linux 的内核原理，网上找了些资料进行了汇总和记录如下：

1、找到 ubuntu 网站（https://launchpad.net/ubuntu/+source/linux/3.8.0-19.29）下载内核源码

linux_3.8.0.orig.tar.gz

解压到本地文件夹；

2、本文采用 make menuconfig 方式进行内核的编译（方法有很多种）；

即 apt-get install libncurses5-dev，安装后进入解压的文件夹，运行命令 make menuconfig 即可出现图形化界面；

得到如下图所示的界面：

以上则表示进入了内核的配置界面。

这过程中也遇到了一个问题，libncurses5-dev 无法安装的问题（无法解析软件包文件 /var/lib/apt/lists/cn.archive.....），主要是源太旧了，需要 apt-get update 一下，网上的方法可以解决问题，即：

正在读取软件包列表... 有错误！

E: Encountered a section with no Package: header

E: Problem with MergeList /var/lib/apt/lists/cn.archive.ubuntu.com_ubuntu_dists_

natty_main_i18n_Translation-en

E: 无法解析或打开软件包的列表或是状态文件。

问题：软件包出错

解决方案：

sudo rm /var/lib/apt/lists/* -vf

sudo apt-get update

下一步翻译和分析下配置选项的含义和作用：

A：最上面说明翻译如下：

箭头表示含有子菜单

enter 选择子菜单

高亮的字母表示热键

<Y>表示包含

<N>表示除外

<M>模块化功能

按 Esc Esc 两次退出

<?>表示帮助

</>表示查询

图例：[*] 内置；[ ] 除外；<M> 模块；< >模块 capable

B：帮助选项中的概览说明如下：

这个接口让你可以选择需要的功能和参数进行编译。功能可以是内置的，模块化的或者直接忽略。参数必须以十进制或十六进制或文本的方式进行输入。

菜单选项中的行首括号代表的含义如下：

[ ] 表示可以内置或者移除

<>表示可以内置，模块化或者移除

{ }表示内置或模块化（根据其他功能选择）

- - 根据其他功能选择

但 括号中为*，M 或者空白时表示内置，以模块编译或者单独的执行该功能

如要改变这些功能选项，则可以将光标移动到对应选项上进行高亮选中，然后按<Y>来内置，<M>模块化功能，或<N>来移除。

也可以通过按空格键来循环可选项（ie. Y->N->M->Y）

下方框中的内容依次是：

x x General setup ---> x x 常规设置

x x [*] Enable loadable module support ---> x x 允许模块加载支持

x x [*] Enable the block layer ---> x x 允许块设备支持

x x Processor type and features ---> x x 处理器类型和功能

x x Power management and ACPI options ---> x x 电源管理和 ACPI 选项

x x Bus options (PCI etc.) ---> x x 总线选项（PCI 等）

x x Executable file formats / Emulations ---> x x 可执行文件格式/仿真

x x -*- Networking support ---> x x 网络支持

x x Device Drivers ---> x x 设备驱动

x x Firmware Drivers ---> x x 固件驱动

x x File systems ---> x x 文件系统

x x Kernel hacking ---> x x 内核监视

x x Security options ---> x x 安全选项

x x -*- Cryptographic API ---> x x 密码 API

x x [*] Virtualization ---> x x 虚拟化

x x Library routines ---> x x 程序库程序

x x --- x x

x x Load an Alternate Configuration File x x 加载备用配置文件

x x Save an Alternate Configuration File x x 保存备用配置文件

（一）General setup：

与 Linux 最相关的程序互动、核心版本说明、是否使用程序代码等信息都在这里设定。这里的项目主要都是针对核心不同程序之间的相关性来设计。

（二）Enable loadable module support：

模块是一小段代码，编译后可在系统内核运行时动态的加入内核，从而为内核增加一些特性或是对某种硬件进行支持。一般一些不常用到的驱动或特性可以编译为模块以减少内核的体积。在运行时可以使用 modprobe 命令来加载它到内核中去(在不需要时还可以移除它)。一些特性是否编译为模块的原则是，不常使用的，特别是在系统启动时不需要的驱动可以将其编译为模块，如果是一些在系统启动时就要用到的驱动比如

说文件系统，系统总线的支持就不要编为模块，否则无法启动系统。在启动时不用到的功能，编成模块是最有效的方式。

（三）Enable the block layer：

使用硬盘/USB/SCSI 设备者必选这选项使得块设备可以从内核移除。如果不选，那么 blockdev 文件将不可用，一些文件系统比如 ext3 将不可用。这个选项会禁止 SCSI 字符设备和 USB 储存设备，如果它们使用不同的块设备。

（四）Processor type and features：

根据自己 CPU 的类型进行适合的设置。

（五）Power management and ACPI options：

对电源进行管理。ACPI 表示高级配置和电源管理接口（Advanced Configuration and Power Management Interface）。

（六）Bus options (PCI etc.)：

这个项目和总线有关。分为最常见的 PCI 和 PCI-express 的支持,还有笔记本电脑常见的 PCMCIA 插卡等等。

（七）Executable file formats / Emulations：

是给 Linux 核心运作执行文件之用的数据。通常与不编译行为有关。

（八）Networking support：

是相当重要的选项，因为他还包括了防火墙相关的项目。由于防火墙是在启动网络之后再设定即可，所以绝大部分的内容都可以被编译成为模块，而且也建议编成模块。有用到再载入到核心即可。

（九）Device Drivers：

设备驱动设定。

（十）Firmware Drivers：

固件驱动设定。

（十一）File systems：

文件系统的支持也是很重要的一项核心功能。因为如果不支持某个文件系统，那我们的 Linux kernel 就无法识别，当然也就无法使用。

（十二）Kernel hacking：

这里和核心开发者比较有关的部分，这部分建议保留默认值即可， 应该不需要去修改，除非你想要进行核心方面的研究。

（十三）Security options：

这里是属于信息安全方面的设定，包括 SELinux 这个细部权限强化模块也是在这里编入核心的，这部分可以作一些额外的设定。

（十四）Cryptographic API：

是微软在 Windows 操作系统中添加的密码编译机能，作为资料加密与解密功能的重要基础，CryptoAPI 支持同步，异步的金钥加密处理，以及操作系统中的数位凭证的管理工作。

（十五）Virtualization：

虚拟化是近年来非常热门的一个议题，因为计算机的能力太强，所以时常闲置在那边，此时，我们可以透过虚拟化技术在一部主机上面同时启劢多个操作系统来运作，这就是所谓的虚拟化。 Linux 核心已经主动的纳入虚拟化功能。而 Linux 认可的虚拟化使用的机制为 KVM(Kernel base Virtual Machine)。

（十六） Library routines：

常用的核心函式库也可以全部编为模块

（十七）Load an Alternate Configuration File：

（十八）Save an Alternate Configuration File：

3、第一个选项回车后得到如下界面：

突然发现内容挺多的，所以下一篇在写 general setup 的内容，否则看起都累，还容易搞混

所有选项截图如下：

x x [*] Prompt for development and/or incomplete code/drivers x x 即时开发/非完整代码/驱动

x x () Cross-compiler tool prefix x x 交叉编译工具前缀

x x () Local version - append to kernel release x x 本地版本--追加内核版本

x x [ ] Automatically append version information to the version string x x 自动在版本字符串后追加版本信息

x x Kernel compression mode (Gzip) ---> x x 内核压缩模式

x x ((none)) Default hostname x x 默认主机名

x x [*] Support for paging of anonymous memory (swap) x x 支持使用交换分区来作为虚拟内存

x x [*] System V IPC x x system V 进程间通信支持

x x [*] POSIX Message Queues x x POSIX 消息队列

x x [*] open by fhandle syscalls x x 使用文件句柄进行系统调用

x x -*- Auditing support x x 审计支持

x x [*] Enable system-call auditing support x x 支持对系统调用的审计

x x [ ] Make audit loginuid immutable x x 使审计登陆用户 ID 不可变

x x IRQ subsystem ---> x x 中断请求子系统

x x Timers subsystem ---
> x x 定时器子系统

x x CPU/Task time and stats accounting ---
> x x CPU/Task 时间与状态保护

x x RCU Subsystem ---
> x x 高性能的锁机制
RCU 子系统

x x < > Kernel .config
support x x 内核.config 支持

x x (17) Kernel log buffer size (16 => 64KB, 17 => 128KB) x
x 内核日志缓冲大小

x x -*- Control Group support ---
> x x 控制组支持

[*]Prompt for development and/or incomplete code/drivers

显示尚在开发中或尚未完成的代码与驱动，你应该选择它，因为有许多设备可能必需选择这个选项才能进行配置，实际上它是安全的。这个选项同样会让一些老的驱动可用。如果你选了 Y，你将会得到更多的阿尔法版本的驱动和代码的配置菜单。

() Cross-compiler tool prefix

交叉编译工具前缀，如果你要使用交叉编译工具的话输入相关前缀，默认不使用，不需要。

()Local version - append to kernel release

在内核版本后面加上自定义的版本字符串(小于 64 字符)，可以用"uname -a"命令看到

[ ]Automatically append version information to the version string

自动生成版本信息。这个选项会自动探测你的内核并且生成相应的版本，使之不会和原先的重复。这需要 Perl 及 git 仓库支持的支持。

Kernel compression mode (Gzip)

内核压缩模式选 gzip2

gzip 用于 UNIX 系统的文件压缩。 后缀为.gz 的文件。现今已经成为 Internet 上使用非常普遍的一种数据压缩格式，或者说一种文件格式。HTTP 协议上的 GZIP 编码是一种

用来改进 WEB 应用程序性能的技术。大流量的 WEB 站点常常使用 GZIP 压缩技术来让用户感受更快的速度。

bzip2 是一个基于 Burrows- Wheeler 变换的无损压缩软件，压缩效果比传统的 LZ77/LZ78 压缩算法来得好。它是一款免费软件。bzip2 能够进行高质量的数据压缩。它利用先进的压缩技术，能够把普通的数据文件压缩 10%至 15%，压缩的速度和解压的效率都非常高！支持现在大多数压缩格式，包括 tar、gzip 等等。

lzma 是一个 Deflate 和 LZ77 算法改良和优化后的压缩算法，开发者是 Igor Pavlov，2001 年被首次应用于 7-Zip 压缩工具中，是 2001 年以来得到发展的一个数据压缩算法。

[*] Support for paging of anonymous memory (swap)

将使你的内核支持虚拟内存。这个虚拟内存在 LINUX 中就是 SWAP 分区。除非你不想要 SWAP 分区，否则这里必选 Y。

[*] System V IPC

System V 进程间通信(IPC)支持，处理器在程序之间同步和交换信息，如果不选这项，很多程序运行不起来，特别地，你想在 LINUX 下运行 DOS 仿真程序，你必须要选 Y。

[*] POSIX Message Queues

POSIX 消息队列，这是 POSIX IPC 中的一部分。最好将它选上 POSIX 表示可移植操作系统接口

[*] open by fhandle syscalls

If you say Y here, a user level program will be able to map file names to handle and then later use the handle for different file system operations. This is useful in implementing userspace file servers, which now track files using handles instead of names. The handle would remain the same even if file names get renamed.

[*] Auditing support

审计支持，用于和内核的某些子模块同时工作，(例如 SELinux)需要它，只有同时选择其子项才能对系统调用进行审计。

允许审计的下层能够被其他内核子系统使用，比如 SE - Linux，它需要这个来进行登录时的声音和视频输出。没有 CONFIG_AUDITSYSCALL 时（即下一个选项）无法进行系统调用。

[*] Enable system-call auditing support

支持对系统调用的审计。允许系统独立地或者通过其他内核的子系统，调用审计支持，比如 SE - Linux。要使用这种审计的文件系统来查看特性，请确保 INOTIFY 已经被设置。

[ ] Make audit loginuid immutable

使得审计登录用户 ID 不可更改(具体作用不明确，新功能)

IRQ subsystem

中断请求子系统

Timers subsystem

定时器子系统

CPU/Task time and stats accounting

CPU/Task 时间与状态保护

RCU Subsystem

RCU(Read-Copy Update)，顾名思义为读取-复制更新。对于被 RCU 保护的共享数据结构，读者不需要获得任何锁就可以访问它，但在访问它时首先拷贝一个副本，然后对副本进行修改，最后使用一个回调（callback）机制在适当的时机把指向原来数据的指针重新指向新的被修改的数据。这个时机就是所有引用该数据的 CPU 都退出对共享数据的操作。

< > Kernel .config support

这个选项允许.config 文件（即编译 LINUX 时的配置文件）保存在内核当中。它提供正在运行中的或者还在硬盘中的内核的相关配置选项。可以通过内核镜像文件 kernel image file 用命令 script scripts/extract-ikconfig 来提取出来，作为当前内核重编译或者另一个内核编译的参考。如果你的内核在运行中，可以通过/proc/config.gz 文件来读取。下一个选项提供这项支持。看起来好像是一个不错的功能，可以把编译时的 .config 文件保存在内核中，以供今后参考调用。用来重编译和编译其他的内核的时候可以用上。

(17) Kernel log buffer size (16 => 64KB, 17 => 128KB)

内核日志缓存的大小，12 => 4 KB，13 => 8 KB，14 => 16 KB 单处理器，15 => 32 KB 多处理器，16 => 64 KB for x86 NUMAQ or IA-64，17 => 128 KB for S/390

-*- Control Group support

cgroup 支持，如 cpusets 那样来使用 cgroup 子系统进程（不确定可以不选）。添加对进程集合分组的支持，用来处理控制子系统如：CPUsets.CFS.内存控制或设备隔离。(This option adds support for grouping sets of processes together, for use with process control subsystems such as Cpusets, CFS, memory controls or device isolation.)

[ ] Checkpoint/restore support (NEW)

检查点和恢复支持

-*- Namespaces support

命名空间支持，允许服务器为不同的用户信息提供不同的用户名空间服务

[*] Automatic process group scheduling

自动进程分组调度。优化调度器对通常的桌面工作量通过自动创建和填充进任务组。(This option optimizes the scheduler for common desktop workloads by automatically creating and populating task groups. This separation of workloads isolates aggressive CPU burners (like build jobs) from desktop applications. Task group autogeneration is currently based upon task session.)

[ ] Enable deprecated sysfs features to support old userspace tools

允许弃用 sysfs 功能来支持原有用户空间的工具。在某些文件系统上(比如 debugfs)提供从内核空间向用户空间传递大量数据的接口

-*- Kernel->user space relay support (formerly relayfs)

内核-用户空间传递支持。在某些文件系统上(比如 debugfs)提供从内核空间向用户空间传递大量数据的接口

[*] Initial RAM filesystem and RAM disk (initramfs/initrd) suppor

初始 RAM 的文件和 RAM 磁盘（ initramfs /initrd）支持（如果要采用 initrd 启动则要选择，否则可以不选）

() Initramfs source file(s)

initrd 已经被 initramfs 取代，如果不明白这是什么意思，请保持空白

[*] Support initial ramdisks compressed using gzip

[*] Support initial ramdisks compressed using bzip2

[*] Support initial ramdisks compressed using LZMA

[*] Support initial ramdisks compressed using XZ

[*] Support initial ramdisks compressed using LZO

以上五项为支持初始化虚拟内存盘压缩使用 gzip.bzip2.LZMA.XZ.LZO 格式.。

[*] Optimize for size

编译时优化内核尺寸。这个选项将在 GCC 命令后用 “-Os” 代替 “-O2”参数，这样可以得到更小的内核。没必要选。一个编译好的内核才 7 - 10 多 M，大家不会少这么点空间吧。选上了可能会出一些问题。最好不选。有时会产生错误的二进制代码。

[ ] Configure standard kernel features (for small systems)

配置标准的内核特性(为小型系统)。这个选项可以让内核的基本选项和设置无效或者扭曲。这是用于特定环境中的，它允许“非标准”内核。你要是选它，你一定要明白自己在干什么。这是为了编译某些特殊用途的内核使用的，例如引导盘系统。

[*] Enable 16-bit UID system calls

允许 16 位 UID 系统调用

[*] Sysctl syscall support

sysctl 系统调用支持。不需要重启就能修改内核的某些参数和变量，如果你也选择了支持/proc，将能从/proc/sys 存取可以影响内核行为的参数或变量

-*- Load all symbols for debugging/ksymoops

为调试和 ksymoops 加载所有的符号表。装载所有的调试符号表信息，仅供调试时选择

[*] Include all symbols in kallsyms

在 kallsyms 中包含内核知道的所有符号，内核将会增大 300K

[*] Enable support for printk

允许 printk 支持，允许内核向终端打印字符信息，在需要诊断内核为什么不能运行时选择

[*] BUG() support

启用 BUG 事件支持，显示故障和失败条件(BUG 和 WARN)，禁用它将可能导致隐含的错误被忽略

[*] Enable ELF core dumps

内存转储支持，可以帮助调试 ELF 格式的程序

[*] Enable full-sized data structures for core

在内核中使用全尺寸的数据结构。禁用它将使得某些内核的数据结构减小以节约内存，但是将会降低性能

[*] Enable PC-Speaker support

允许扬声器支持

[*] Enable futex support

快速用户空间互斥体。可以使线程串行化，也提高了响应速度。禁用它将导致内核不能正确的运行基于 glibc 的程序

[*] Enable eventpoll support

支持事件轮循的系统调用

[*] Enable signalfd() system call

启用 signalfd()事件的文件描述符系统调用

[*] Enable timerfd() system call

启用 timefd()事件的文件描述符系统调用

-*- Enable eventfd() system call

启用 eventfd()事件的文件描述符系统调用

Use full shmem filesystem

完全使用 shmem 来代替 ramfs。shmem 是基于共享内存的文件系统(可能用到 swap)，在启用 TMPFS 后可以挂载为 tmpfs 供用户空间使用，它比简单的 ramfs 先进许多

[*] Enable AIO support

允许 POSIX 异步，I/O 可能会被一些高性能的线程程序使用(This option enables POSIX asynchronous I/O which may by used by some high performance threaded applications. Disabling this option saves about 7k.)

[ ] Embedded system

嵌入式系统.

Kernel Performance Events And Counters

支持软件和硬件提供的各种性能事件(Enable kernel support for various performance events provided by software and hardware.Software events are supported either built-in or via the use of generic tracepoints.Most modern CPUs support performance events via performance counter registers.).

[*] Enable VM event counters for /proc/vmstat

允许在/proc/vmstat 中包含虚拟内存事件记数器.

[*] Enable PCI quirk workarounds

这能使工作区从各种 PCI 芯片组错误中恢复过来(This enables workarounds for various PCI chipset bugs/quirks. Disable this only if your target machine is unaffected by PCI quirks.).

[*] Enable SLUB debugging support

支持 SLUB 内存分配管理器调试.

[ ] Disable heap randomization

堆不可随机化

Choose SLAB allocator (SLUB (Unqueued Allocator))

使用 SLAB 完全取代 SLOB 进行内存分配，SLAB 是一种优秀的内存分配管理器，推荐使用

[*] Profiling support

支持系统评测(对于大多数用户来说并不是必须的).

<M> OProfile system profiling

评测和性能监控工具.

[ ] OProfile multiplexing support (EXPERIMENTAL)

硬件计数器数字被限制(The number of hardware counters limited. The multiplexing feature enables OProfile to gather more events than counters are provided by the hardware. This is realized by switching between events at an user specified time interval.).

[*] Kprobes (Kernel Dynamic Probes)

内核动态探针.(provides a lightweight interface for kernel modules to implant probes and register corresponding probe handlers.).

[*] Optimize very unlikely/likely branches

相似/不相似分支选择

GCOV-based kernel profiling

这个选项允许 gcov - based 代码剖析(gcov 是 gnu/gcc 工具库中的一个组件,一般来说，都被安装的)(This option enables gcov-based code profiling (e.g. for code coverage measurements).)

[*]Prompt for development and/or incomplete code/drivers

显示尚在开发中或尚未完成的代码与驱动.你应该选择它，因为有许多设备可能必需选择这个选项才能进行配置，实际上它是安全的。这个选项同样会让一些老的驱动的可用。如果你选了 Y，你将会得到更多的阿尔法版本的驱动和代码的配置菜单。

( ) Local version - append to kernel release

在内核版本后面加上自定义的版本字符串(小于 64 字符),可以用"uname -a"命令看到

[ ]Automatically append version information to the version string

自动生成版本信息。这个选项会自动探测你的内核并且生成相应的版本，使之不会和原先的重复。这需要 Perl 的支持。由于在编译的命令 make-kpkg 中我们会加入- –append-to-version 选项来生成自定义版本，所以这里选 N。

Kernel compression mode (Gzip)

内核压缩模式选 baip2

gzip 用于 UNIX 系统的文件压缩。后缀 为.gz 的文件。现今已经成为 Internet 上使用非常普遍的一种数据压缩格式，或者说一种文件格式。HTTP 协议上的 GZIP 编码是一种用来改进 WEB 应用程序性能的技术。大流量的 WEB 站点常常 使用 GZIP 压缩技术来让用户感受更快的速度。

bzip2 是一个基于 Burrows- Wheeler 变换的无损压缩软件，压缩效果比传统的 LZ77/LZ78 压缩算法来得好。它是一款免费软件。bzip2 能够进行高质量的数据压缩。它利用先进的压缩技 术，能够把普通的数据文件压缩 10%至 15%，压缩的速度和解压的效率都非常高！支持现在大多数压缩格式，包括 tar、gzip 等等。

lzma 是一个 Deflate 和 LZ77 算法改良和优化后的压缩算法，开发者是 Igor Pavlov，2001 年被首次应用于 7-Zip 压缩工具中，是 2001 年以来得到发展的一个数据压缩算法。它使用类似于 LZ77 的字典编码机制，在一般的情况

[*] Support for paging of anonymous memory (swap)

将使你的内核支持虚拟内存。这个虚拟内存在 LINUX 中就是 SWAP 分区。除非你不想要 SWAP 分区，否则这里必选 Y。

[*] System V IPC

System V 进程间通信(IPC)支持,于处理器在程序之间同步和交换信息，如果不选这项，很多程序运行不起来,特别地，你想在 LINUX 下运行 DOS 仿真程序，你必须要选 Y。

[*] POSIX Message Queues

POSIX 消息队列,这是 POSIX IPC 中的一部分。建议你最好将它选上 POSIX 表示可移植操作系统接口

[*] BSD Process Accounting

这是允许用户进程访问内核，将账户信息写入文件中。这通常被认为是个好主意，建议你最好将它选上。将进程的统计信息写入文件的用户级系统调用,主要包括进程的创建时间/创建者/内存占用等信息。

[ ]BSD Process Accounting version 3 file format

选 Y，统计信息将会以新的格式（V3）写入，这格式包含进程 ID 和父进程。注意这个格式和以前的 v0/v1/v2 格式不兼容，所以你需要 升级相关工具来使用它。

[*] Export task/process statistics through netlink (EXPERIMENTAL)

处于实验阶段的功能。通过通用的网络输出工作/进程的相应数据，和 BSD 不同的是，这些数据在进程运行的时候就可以通过相关命令访问。和 BSD 类似，数据将在进程结束时送入用户空间。如果不清楚，选 N。

[*] Enable per-task delay accounting (EXPERIMENTAL)

在统计信息中包含进程等候系统资源(cpu,IO 同步,内存交换等)所花费的时间

[*] Enable extended accounting over taskstats (EXPERIMENTAL)

在统计信息中包含扩展进程所花费的时间

[*] Enable per-task storage I/O accounting (EXPERIMENTAL)

在统计信息中包含 I/O 存储进程所花费的时间

[*] Auditing support

审计支持,用于和内核的某些子模块同时工作，(例如 SELinux)需要它,只有同时选择其子项才能对系统调用进行审计。

允许审计的下层能够被其他内核子系统使用，比如 SE - Linux，它需要这个来进行登录时的声音和视频输出。没有 CONFIG_AUDITSYSCALL 时（即下一个选项）无法进行系统调用。

[*] Enable system-call auditing support

支持对系统调用的审计。允许系统独立地或者通过其他内核的子系统，调用审计支持，比如 SE - Linux。要使用这种审计的文件系统来查看特性，请确保 INOTIFY 已经被设置。

RCU Subsystem ---> 一个高性能的锁机制 RCU 子系统

RCU(Read-Copy Update)，顾名思义为读取-复制更新。对于被 RCU 保护的共享数据结构，读者不需要获得任何锁就可以访问它，但写者在访问它时首先拷贝一个副本，然后对副本进行修改，最后使用一个回调（callback）机制在适当的时机把指向原来数据的指针重新指向新的被修改的数据。这个时机就是所有引用该数据的 CPU 都退出对共享数据的操作。

RCU Implementation (Tree-based hierarchical RCU) --->

RCU 实现机制

(X) Tree-based hierarchical RCU

基本数按等级划分

[ ] Enable tracing for RCU

激活跟踪

(32) Tree-based hierarchical RCU fanout value

基本数按等级划分分列值

[ ] Disable tree-based hierarchical RCU auto-balancing

< > Kernel .config support

这 个选项允许.config 文件（即编译 LINUX 时的配置文件）保存在内核当中。它提供正在运行中的或者还在硬盘中的内核的相关配置选项。可以通过内核镜 像文件 kernel image file 用命令 script scripts/extract-ikconfig 来提取出来，作为当前内核重编译或者另一个内核编译的参考。如果你的内核在运行中，可以通过/proc/config.gz 文件来读取。下一个选项提供这 项支持。

看起来好像是一个不错的功能，可以把编译时的 .config 文件保存在内核中，以供今后参考调用。用来重编译和编译其他的内核的时候可以用上。你是一个编译内核的狂人的话，这项要选上

[ ] Enable access to .config through /proc/config.gz (NEW)

上一项的子项，可 以通过/proc/config.gz 访问当前内核的.config 。新功能，上一项选的话这个就选上吧。

(18) Kernel log buffer size (16 => 64KB, 17 => 128KB)

内 核日志缓存的大小，12 => 4 KB，13 => 8 KB，14 => 16 KB 单处理器，15 => 32 KB 多处理器，16 => 64 KB for x86 NUMAQ or IA-64，17 => 128 KB for S/390

[*] Control Group support --->

cgroup 支持，如 cpusets 那样来使用 cgroup 子 系统进程（不确定可以不选）

[ ] Example debug cgroup subsystemcgroup

子系统调试例子

[*] Namespace cgroup subsystem cgroup

子系统命名空间

[*] Freezer cgroup subsystem

[*] Cpuset support

只有含有大量 CPU(大于 16 个)的 SMP 系统或 NUMA(非一致内存访问)系统才需要它

[*]   Include legacy /proc//cpuset file

[*] Simple CPU accounting cgroup subsystem

简单 cgroup 子系统 cpu 所花费的时间

[ ]   Memory Resource Controller for Control Groups

cgroup 内存资源控制器

[ ] enable deprecated sysfs features which may confuse old userspce tools

在某些文件系统上(比如 debugfs)提供从内核空间向用户空间传递大量数据的接口

-*- Kernel->user space relay support (formerly relayfs)

内核系统区和用户区进行传递通讯的支持。这个选项在特定的文 件系统中提供数据传递接口支持，它可以提供从内核空间到用户空间的大批量的数据传递工具和设施。如果不清楚，选 N。

-*- Namespaces support

命名空间支持，允许服务器为不同的用户信息提供不 同的用户名空间服务

[*] UTS namespace

通用终端系统的命名空间。它允许容器，比如 Vservers 利用 UTS 命名空间来为不同的服务器提供不同 的 UTS。如果不清楚，选 N。

[*] IPC namespace

IPC 命名空间，不确定可以不选

[*] Initial RAM filesystem and RAM disk (initramfs/initrd) suppor

初始 RAM 的文件和 RAM 磁盘（ initramfs /initrd）支持（如果要采用 initrd 启动则要选择，否则可以不选）

() Initramfs source file(s)

initrd 已经被 initramfs 取代,如果你不明白这是什么 意思,请保持空白

[*] Optimize for size

这个选项将在 GCC 命令后用 “-Os” 代替 “-O2”参数，这样可以得到更小的内核。没必要选。一个编译好的内核才 7 - 10 多 M，大家不会少这么点空间吧。选上了可能会出一些问题。最好不选。有时会产生错误的二进制代码。

[ ] Configure standard kernel features (for small systems) --->

配置标准的内核特性(为小型系统)。这个选项可以让内核的基本选项和设置无效或者扭曲。这是用于特定环境中的，它允许“非标准”内核。你要是选它，你一定要明白自己在干什么。这是为了编译某些特殊用途的内核使用的，例如引导盘系统。

-*- Load all symbols for debugging/ksymoops

装载所有的调试符号表信息,仅供调试时选择

[*] Include all symbols in kallsyms

在 kallsyms 中包含内核知道的所有符号,内核将会 增大 300K

[*] Do an extra kallsyms pass

除非你在 kallsyms 中发现了 bug 并需要 报告这个 bug 才打开该选项

Kernel Performance Events And Counters --->

-*- Kernel performance events and counters

[*] Tracepoint profiling sources

[ ] Kernel performance counters (old config option)

[ ] Disable heap randomization

禁用随机 heap（heap 堆是一个应用层的概念，即堆对 CPU 是不可见的，它的实现方式有多种，可以由 OS 实现，也可以由运行库实现,如果你愿意，你也可以在一个栈中来实现一个堆）

Choose SLAB allocator (SLUB (Unqueued Allocator)) --->

选择内存分配管 理器（强烈推荐使用 SLUB）

( ) SLAB

各种环境通用的内存分配管理器

(X) SLUB (Unqueued Allocator)

更加优秀的内存分配管理器

[*] Profiling support (EXPERIMENTAL)

剖面支持，用一个工具来扫描和提供计算机的剖面图。支持系统评测（对于大多数用户来说并不是必须的）

< > OProfile system profiling (EXPERIMENTAL)

OProfile 评测和性能监控工具

[*] Kprobes

调试内核除非开发人员，否则不选

GCOV-based kernel profiling --->

[ ] Enable gcov-based kernel profiling

[ ] Slow work debugging through debugfs

暂时到此吧，下来写其它的。

Archiso 是用于构建 Arch Linux Live CD ISO 映像的工具。 官方映像 是使用 Archiso 构建的。 Archiso 是可配置的，并且可以用作不同系统的基础，例如救援系统或 Linux 安装程序。 这篇 Wiki 文章介绍了如何安装 Archiso，以及如何配置它以控制生成的 ISO 映像的各个方面，例如随附的软件包和文件。 技术需求和构建步骤可以在 官方项目文档 中找到。 Archiso 通过许多 bash 脚本实现。 Archiso 的核心组件是 mkarchiso 命令。 其选项记录在其用法输出中，此处未涉及。 用户不需要直接与 mkarchiso 交互。

Contents

5

在 QEMU 中运行 ISO

6

参阅

6.1

文档和教程

6.2

示例自定义模板

6.3

创建脱机安装 ISO

6.4

查找以前版本的 ISO 映像

安装和配置

注意： 以下操作请在 ROOT 权限下使用，在错误的权限下操作会造成问题。

安装 archiso 或 archiso-gitAUR。

Archiso 附带 2 个预定义配置（profiles）：releng 和 baseline。

releng 用于创建正式的每月安装 ISO。它可以作为创建自定义 ISO 映像的起点。

baseline 是一种最低限度的配置，它只包括从介质启动实时环境所需的最低限度的软件包。

使用以下命令将您选择的配置文件复制到目录中。 用 releng 或 baseline 替换 profile，并用你选择的目录名称替换 archlive。

cp -r /usr/share/archiso/configs/profile archlive

现在，根据所选的配置文件和您的目标进入不同的章节。

releng：如果您选择了 releng ，并且

如果要自定义映像，请继续进行章节 #配置 Live 介质。

如果要复制官方映像，请继续进行 #构建 ISO。

baseline：如果您选择了 baseline 配置文件，请继续进行 #构建 ISO。

配置 Live 介质

本节详细介绍如何配置映像，以及定义以后将复制到映像的软件包和配置。

在 #安装和配置 中创建的 archlive 目录中有许多文件和目录; 我们只关注以下的：

packages.x86_64 - 此文件列出了 Live 系统映像上安装的软件包。

airootfs - 此目录用作覆盖，并允许您进行自定义。

您可以通过编辑 archlive/airootfs/root/customize_airootfs.sh 来编写遵循安装指南（或安装后）的所有管理任务的脚本，除了软件包安装。该脚本是从正在运行的 Live 系统的角度编写的，即，在脚本中，路径 / 是指正在运行的 Live 系统的根目录。

安装包

编辑 packages.x86_64 中的软件包列表，以配置将哪些软件包安装在 Live 系统映像上，逐行列出软件包。

注意： 如果要在 Live CD 中使用 窗口管理器，则必须添加必要且正确的 video drivers，否则 WM 可能在加载时冻结。

自定义本地库

为了准备自定义软件包或从 AUR/ABS 准备软件包，您还可以 自建本地仓库。

然后，您可以将以下内容放入 ~/archlive/pacman.conf 中，添加到其他仓库条目上方（最高优先级）：

~/archlive/pacman.conf

...

custom repository

[customrepo]

SigLevel = Optional TrustAll

Server = file:///home/user/customrepo/$arch

...

安装 multilib 中的软件包

要从 Multilib 资源库中安装软件包，仅需在~/archlive/pacman.conf 中取消注释以下行以启用 multilib：

pacman.conf

[multilib]

SigLevel = PackageRequired

Include = /etc/pacman.d/mirrorlist

向映像里添加文件

注意： 你必须使用 root 权限来完成这件事，不要改变任何你复制过来的文件的所有权，airootfs 中的所有文件必须为 root 用户所有。适当的所有制将在不久被解决。

airootfs 目录作为要覆盖的文件，把它看作是在当前系统上的根目录'/'，所以你在这个目录中放置的任何文件都将在开机时被复制。

所以，如果你在当前系统上有一组 iptables 脚本且想要在 Live 映像上使用，请这样复制:

cp -r /etc/iptables ~/archlive/airootfs/etc

在用户 home 文件夹里放置文件的方法有些许不同。不要把它们放在 airootfs/home，而是在 airootfs/ 里创建 skel 目录并将其放置在那里。然后我们会将相关命令添加到 customize_airootfs.sh——我们要使用它在引导时复制文件以及梳理权限。

首先，创建 skel 目录

mkdir ~/archlive/airootfs/etc/skel

现在，复制 'home' 的文件到 skel 目录。例如，对于 .bashrc：

cp ~/.bashrc ~/archlive/airootfs/etc/skel/

当 ~/archlive/airootfs/root/customize_airootfs.sh 被执行，并且一个新用户已被创建，skel 目录中的文件将自动被复制到新的 home 文件夹中，并被设置正确的权限。

类似地，需要注意位于层次结构下方的特殊配置文件。作为示例，配置文件 /etc/X11/xinit/xinitrc 位于可能被安装包覆盖的路径上。要将 xinitrc 的配置文件放在 ~/archlive/airootfs/etc/skel/ ，然后修改 customize_airootfs.sh 以适当地移动。

内核

默认 archiso 使用 linux 内核和模块，但创建的 ISO 映像文件可以包含其他或多重内核。首先，编辑 packages.x86_64，将想要的内核包名写入，mkarchiso 运行时会将映像（以及用于 UEFI 引导的 FAT 映像）work_dir/airootfs/boot/vmlinuz-*和 work_dir/boot/initramfs-*.img 中的所有文件导入。 默认的 mkinitcpio 将构建 fallback 镜像。对于映像文件，主 initramfs 镜像不包含 autodetect 钩子拓展，因此没有必要创建一个额外的 fallback 镜像。 自定义替换 archlive/airootfs/etc/mkinitcpio.d/pkgbase.preset，以跳过不必要的创建，如 linux-lts:

archlive/airootfs/etc/mkinitcpio.d/linux-lts.preset

PRESETS=('archiso')

ALL_kver='/boot/vmlinuz-linux-lts'

ALL_config='/etc/mkinitcpio.conf'

archiso_image="/boot/initramfs-linux-lts.img"

最后，创建 boot loader configuration 配置文件来启动内核。

引导器

默认的文件应该可以正常工作，所以你应该不需要去碰它。

由于 isolinux 的模块化性质，你能够使用很多插件，因为所有 *.c32 都被复制且可以可供您使用。看看 syslinux 官方网站 和 Archiso Git Repo。使用所述插件，便有可能做出更具视觉吸引力及复杂的菜单。参见 此处。

UEFI 安全启动

如果要使 Archiso 在启用 UEFI 安全启动的环境下可启动，则必须使用签名过的启动加载程序。您可以按照 Secure Boot#Booting an installation medium 中的说明进行操作。

登录管理器

通过启用您登录管理器的 systemd 服务来做到在引导时启动 X 。如果您知道哪一个 .service 文件需要软链接，那就太好了。如果不知道，你也可以轻松找出——如果你在创建 iso 所用的系统上使用相同的程序的话。只需使用：

$ ls -l /etc/systemd/system/display-manager.service

现在在 ~/archlive/airootfs/etc/systemd/system 中创建相同的软链接。如 LXDM：

ln -s /usr/lib/systemd/system/lxdm.service ~/archlive/airootfs/etc/systemd/system/display-manager.service

这将使您在 live 系统启用在系统上启动 LXDM。(This will enable LXDM at system start on your live system.)

或者，您也可以启用 airootfs/root/customize_airootfs.sh 中的服务以及其中启用的其他服务。

如果您希望图形环境在启动过程中自动启动，请确保编辑 airootfs/root/customize_airootfs.sh 并替换

systemctl set-default multi-user.target

为

systemctl set-default graphical.target

改变自动登录

Getty 的自动登录配置位于 airootfs/etc/systemd/system/getty@tty1.service.d/autologin.conf。

您可以修改这个文件来更改自动登录用户：

[Service]

ExecStart=

ExecStart=-/sbin/agetty --autologin isouser --noclear %I 38400 linux

或者干脆删除它来禁用自动登录。

构建 ISO

现在，你已经准备好把你的文件转换成 .iso，以便可以刻录到 CD 或 USB。

在 ~/archlive 中执行：

mkarchiso -v -w /path/to/work_dir -o /path/to/out_dir /path/to/profile/

-w 指定工作目录。如果未指定该选项，它将默认在当前目录中工作。

-o 指定将放置生成的 ISO 映像的目录。如果未指定该选项，则默认为在当前目录中输出。

应该注意配置文件 profiledef.sh 运行 mkarchiso 时无法指定，只能指定文件的路径。

如果您正在构建未修改的概要文件或自定义概要文件的路径，请将/path/to/profile/替换为/usr/share/archiso/configs/releng/。

该脚本将现在下载并安装你指定的软件包到 work/*/airootfs，创建内核和 init 映像（initramfs），应用您的修改，并最终把 ISO 建立到 out/。

重建 ISO

在修改之后重建 ISO 不被官方支持。但是，通过应用两个步骤很容易。首先，您必须删除工作目录中的锁定文件：

rm -v work/build.make_*

如果您编辑了 airootfs/root/customize_airootfs.sh 以创建非特权用户，则重建将在创建时失败，因为该用户已经存在（FS#41865）。 为避免此问题，您需要在运行 useradd 之前检查用户是否存在，例如 通过运行 id：

```
! id arch && useradd -m -p "" -g users -G "adm,audio,floppy,log,network,rfkill,scanner,storage,optical,power,wheel" -s /usr/bin/zsh arch
```

同时删除创建的用户或符号链接，如 /etc/sudoers 等持久性数据。

This article or section needs expansion.

Reason: 报告需要删除或重置的更多数据。 (Discuss in Talk:Archiso (简体中文)#)

通过编辑 pacstrap 脚本（位于 /bin/pacstrap）并在第 361 行更改以下内容，可以稍微加快重建速度：

修改前:

```
if ! pacman -r "$newroot" -Sy "${pacman_args[@]}"; then
```

修改后:

```
if ! pacman -r "$newroot" -Sy --needed "${pacman_args[@]}"; then
```

这增加了初始引导的速度，因为它不必下载和安装任何已经安装的基础包。

删除工作目录

临时文件将复制到工作目录中。 如果碰巧 build.sh 脚本被中断，请确保在删除之前没有 mount binds - 否则，您可能会丢失数据（例如，安装在 /run/media/$user/$label 的外部设备在构建过程中被绑定在 work/x86_64/airootfs/run/media/$user/$label 中。

使用 ISO

有关各种选项，请参见 Installation guide (简体中文)#启动到 Live 环境。

在 QEMU 中运行 ISO

```
 qemu-system-x86_64 \
   -accel kvm \
   -boot order=d,menu=on,reboot-timeout=5000 \
   -m size=3072,slots=0,maxmem=$((3072*1024*1024)) \
   -k en \
   -name archiso,process=archiso_0 \
   -drive file=/path/to/archlinux-yyyy.mm.dd-x86_64.iso,media=cdrom,readonly=on \
   -display sdl \
   -vga virtio \
   -enable-kvm \
   -no-reboot \
   -no-shutdown
```

自行构建

请确保你在使用 Arch Linux（若使用 Docker 需要 --privileged 选项）且安装了 archiso 和 git 软件包。

在 root 权限下拉取仓库并开始构建：

```
# git clone https://github.com/bobby285271/Archiso.git
# mkarchiso -v -w work -o out Archiso
```

其中 work 是工作目录，out 是制品目录（可按需修改）。在构建完成后可在 out 获取需要的 ISO 文件并可以直接删除 work。

如果需要进一步定制请参考 ArchWiki，需要特别注意的是 customize_airootfs.sh 已经被弃用。

安装 Archiso

打开终端输入 su 回车确定，输入 root 密码,切换到 root 用户。

在终端内输入 cd 回车确定

在终端内输入 pacman -S archiso --noconfirm 回车确定

(如下图所示)在终端内输入 mkdir archlive 回车确定

在终端内输入 cp -r /usr/share/archiso/configs/releng/* archlive 回车确定

在终端内输入 cd archlive && nano packages.x86_64 回车确定

复制下一步（也就是本文第 4 步）的内容到打开的窗口中，按 Ctrl+X ,再按 Y 回车确定,保存文件.

注：下一步的内容是打包到系统映像中的预装软件包，可以添加自己想要预装的软件包到 packages.x86_64 文件中，格式是每行一个。

如果使用的是其它桌面环境，请去除 xfce4、xfce4-goodies 两个软件包，并添加相应的软件包。

(如下图所示)

xf86-video-intel

xf86-video-ati

xf86-video-nouveau

mesa-libgl

xorg-server

xfce4

xfce4-goodies

wqy-zenhei

wqy-microhei

firefox

flashplugin

fcitx

fcitx-im

fcitx-configtool

fcitx-googlepinyin

libmtp

mtpfs

gvfs-mtp

pulseaudio

gnome-alsamixer

在终端内输入 echo 'startxfce4' > airootfs/s && chmod +x airootfs/s 回车确定

注：airootfs 目录是将来系统的根目录，可以把想要打包到系统映像中的文件，添加到此目录中。

在终端内输入 cp -r /home/用户名/{.config,.mozilla} airootfs/root/ 回车确定

注：把命令中的用户名改成自己的用户名，后面的 .config 文件是当前系统的桌面环境、输入法、一些应用程序等的配置文件。

如果没有安装火狐浏览器，请去除命令中的 .mozilla 注意前面有一个点。

(如下图所示)

在终端内输入 mkdir out && ./build.sh -v 回车确定

就会自动开始构建系统映像(ISO 文件),等构建完成会在 out 目录下出现一个类似

archlinux-20xx.xx.xx-x86_64.iso 的系统映像文件，其中的 xx 代表具体的构建时间。

(如下图所示)

ADPCR-0.2

Atzlinux DEB 软件包测试规范 (版本 0.2)

Atzlinux DEB Package Check Rules (version 0.2)

版本: 0.2（草案）

拟稿:

更新日期: 2020-12-13

一、适用目标和范围

DEB 格式的 Linux 软件包。通常适用于以下 Linux 版本，以及相应的衍生版本。如:

Debian

Ubuntu, UbuntuKylin

LinuxMint, LMDE

Deepin, UOS

Spark-store (星火应用商店。即 Deepin 和 UOS 的社区 DEB 包应用商店)

Atzlinux 项目组收录的第三方 DEB 包（包括软件厂商官方发布的软件包），以及 Atzlinux 项目组制作/维护/修改/优化/打补丁的 DEB 软件包。

二、测试项目:

1. 对 DEB 包进行 lintian 检查。

2. 进行安装测试、启动测试、卸载测试。

2.1 “安装测试”包含两组:

apt-get install XXX.deb

dpkg -i XXX.deb

2.2 “启动测试”包括:

在命令行（或终端），直接运行主程序命令，以及其它主要的附加程序命令。

2.3 “卸载测试”

3. 包名变更后的“升级安装测试”（仅适用于软件包存在多个名称的情况）。

示例 1: 如 zwcad （中望 CAD）软件，Kylin 适配版的 DEB 包名为 zwcad-linuxpreinst。而 Deepin/UOS 适配版的包名为 com.zwsoft.zwcad

三、测试后的 DEB 软件包质量等级评定标准（草案）

质量等级: 由高到低，依次分为 A/B/C/D/E 共五大类。

A/B/C 类: 能正常启动程序的（如可执行程序），或者实现功能调用的（如: 字体包、图标资源包、中国象棋电子棋谱文件）。

D 类（无法实现基本功能）。

E 类: 暂未进行测试，或暂未提交测试结果。

详细分类定义:

A 类（高质量）

101: 打包质量基本达到可以直接提交到 Debian/Ubuntu 的档次；

102: lintian 检测，异常信息输出（主要是 X/E/W 这三类），已控制到最少；

103: 必要时，可以包含 lintian overrides 文件。

B 类（质量中等）

相关缺陷代码定义:

201: md5sums 文件清单校验文件异常。如:

201A: 未包含 DEBIAN/md5sums 文件清单校验文件；

201B: 或者文件与校验码不匹配；

201C: 有校验码无文件；

201D: 有文件无校验码。

202: changelog 文件不规范。如:

202A: 无 doc/changelog.Debian 文件，也无 doc/changelog 文件；

202B: 文件存在，但实际内容空缺；

202C: 文件格式严重不规范。

203: copyright 文件不规范。如:

203A: 无 doc/copyright 文件；

203B: 文件存在，但实际内容空缺；

203C: 文件格式严重不规范（如存在空白模板字段未清理）。

204: 安装后，无 /usr/share/applications/XXX.desktop 桌面菜单项文件 （包括通过安装脚本如 postinst 创建的符号链接）， 但是在其它目录下存在 XXX.desktop 文件（常见于 Deepin/UOS 适配的某些第三方软件包）；

205: desktop 桌面菜单项文件异常。如:

有 XXX.desktop 桌面菜单项文件，但是直接运行命令时，对应的图标显示异常（不显示图标，或者显示为其它图标）；

内部标签值定义语法不符合标准分类或规范（如 Mimetype 或分类 字段后面分号缺失）。

206: DEBIAN/control 文件中，多处标签值填写不符合标准定义，或者标签值填写不完整（如维护人信息中，缺失 e-mail 信息）；

207: DEBIAN/control 文件中，未对同一包名的多个版本名称，设定 冲突/替换/已提供等标签进行处理，导致升级安装时，因文件冲突，导致安装失败。

异常示例 1: 某软件包，针对 Kylin Linux 的适配包，包名为 XXX；针对 Deepin/UOS Linux 的适配包，包名为 com.AAA.XXX。这两个同一来源，不同编译版本的包，安装时通常不能共存。

异常示例 2: 某输入法软件包，旧版本的包名和版本号为 fcitx-xxx 0.1, 新版本名调整为 xxx 0.2。在已安装 fcitx-xxx 0.1 的系统上，直接升级到 xxx 0.2 时，报错“文件冲突”导致升级安装失败。

299: 其它缺陷。

C 类（质量一般。属于“能用”的档次阶段）

301: 大量文件/目录权限异常。包括:

301A: 普通目录，权限非 755。异常示例: 某目录，权限误为 777 或 775。

301B: 普通文件，权限非 644。异常示例: 某图片文件，权限误为 755 （可执行）或 664。

301C: 主要的可执行文件，或者 deamon 守护进程，权限非 755。异常示例: 某主程序相关的 bash 脚本程序，权限误为 644 （不可执行权限）。

302: 大量文件/目录属主异常；

303: 存在无意义的安装脚本未删除(包括 preinst, postinst, prerm, postrm)；

304: 安装脚本中，存在错误的，或者不必要的高风险指令;

305: DEBIAN/control 中架构定义不规范。示例:

某字体包、输入法皮肤包、文档包、图标资源包，架构应为 all, 误设定为 amd64 或者 i386；

实为 i386 的软件包，误标为 amd64，且经 amd64 平台测试验证，确认为 amd64 无法使用。 但本包中的某些文件或资源，能被其它程序和软件包借用和调取，仅提供参考借鉴功能和价值。

390: 中文处理存在功能缺陷

如: 某文档处理软件/行业软件，英文显示、处理、保存、重新读取正常，但中文输入、显示、处理、保存、重新读取时，乱码或文件损坏。

399: 其它严重缺陷。

D 类（无法实现基本功能）

异常码定义:

401: 可执行程序，无法启动运行；

402: 某文档处理软件，保存文件时，异常退出；

499: 其它异常。

E 类: 暂未进行测试，或未对测试结果进行汇总成文

铜豌豆 Linux 软件源 同步

该目录下存放的文件用于 《铜豌豆 Linux》软件源 同步

mirror.list 在服务器上用 apt-mirror 命令直接同步的软件包

用于有 apt 源的软件包

同步第三方 apt 源

2020-11-30 同步: https://github.com/coslyk/debianopt-repo/wiki/Package-list

如果 软件包有 apt 软件源，但是其 apt 软件源仓库太大，则不使用该方式同步

pkg-download-url.list

使用 http 下载的软件包

北方联通下载

https://lx.atzlinux.com:24359/atzlinux-cd/

搬瓦工 CN2-GIA 美国洛杉矶机房

https://www.177vv.com/atzlinux-cd/

https://osdn.net/projects/atzlinux/

清华大学开源软件镜像站下载

https://mirrors.tuna.tsinghua.edu.cn/osdn/storage/g/a/at/atzlinux/atzlinux-cd/

北京外国语大学开源软件镜像站

https://mirrors.bfsu.edu.cn/osdn/storage/g/a/at/atzlinux/atzlinux-cd/

xTom 开源镜像

https://mirror.xtom.com.hk/osdn/storage/g/a/at/atzlinux/atzlinux-cd/

```
############ config ##################
#
# set base_path    /var/spool/apt-mirror
#
# set mirror_path  $base_path/mirror
# set skel_path    $base_path/skel
# set var_path     $base_path/var
# set cleanscript $var_path/clean.sh
# set defaultarch  <running host architecture>
# set postmirror_script $var_path/postmirror.sh
# set run_postmirror 0
set nthreads     20
set _tilde 0
#
############ end config ##############

#deb http://ftp.us.debian.org/debian unstable main contrib non-free
```

```
#deb-src http://ftp.us.debian.org/debian unstable main contrib non-free

# google-chrome-stable
deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main
# google-earth-pro-stable
deb [arch=amd64] http://dl.google.com/linux/earth/deb/ stable main
# opera-stable
deb https://deb.opera.com/opera-stable/ stable non-free
# code
#deb [arch=amd64] http://packages.microsoft.com/repos/vscode stable main
# typora
deb https://typora.io/ linux
# steam-launcher
deb https://repo.steampowered.com/steam stable steam
# skypeforlinux
deb [arch=amd64] https://repo.skype.com/deb stable main
# https://github.com/coslyk/debianopt-repo
# https://github.com/coslyk/debianopt-repo/wiki/Package-list
deb-amd64 https://dl.bintray.com/debianopt/debianopt buster main
# teamviewer
#deb https://linux.teamviewer.com/deb stable main
# freedownloadmanager
deb [arch=amd64] https://debrepo.freedownloadmanager.org/ bionic main
# fsearch-trunk
deb http://ppa.launchpad.net/christian-boxdoerfer/fsearch-daily/ubuntu focal main
# anydesk
deb http://deb.anydesk.com/ all main
# http://archive.kylinos.cn/kylin/partner/pool/
#deb-amd64 http://archive.kylinos.cn/kylin/partner 10.1 main
deb-amd64 http://archive.kylinos.cn/kylin/partner juniper main
```

mirror additional architectures

#deb-alpha http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-amd64 http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-armel http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-hppa http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-i386 http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-ia64 http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-m68k http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-mips http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-mipsel http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-powerpc http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-s390 http://ftp.us.debian.org/debian unstable main contrib non-free

#deb-sparc http://ftp.us.debian.org/debian unstable main contrib non-free

#clean http://ftp.us.debian.org/debian

pkgname section url

section: main contrib non-free

alacritty main https://github.com/alacritty/alacritty/releases

com.thunisoft.input non-free https://pinyin.thunisoft.com/index.html#/sy

bitwarden main
https://vault.bitwarden.com/download/?app=desktop&platform=linux&variant=deb

weibird contrib

llk-linux main http://llk-linux.sourceforge.net/

freedownloadmanager main
https://dn3.freedownloadmanager.org/6/latest/freedownloadmanager.deb

yozo-office non-free http://www.yozosoft.com/

zwcad-linuxpreinst non-free https://www.zwcad.com/product/cad_linux.html

pencil main http://pencil.evolus.vn/Downloads.html

master-pdf-editor non-free https://code-industry.net/get-master-pdf-editor-for-ubuntu/?download

xmind-vana non-free https://www.xmind.net/download/

synergy main https://members.symless.com/download-authentication

openboard main https://github.com/OpenBoard-org/OpenBoard/releases

slack-desktop non-free https://slack.com/intl/en-cn/downloads/instructions/ubuntu

electron-ssr main https://github.com/shadowsocksrr/electron-ssr/tags

teamviewer non-free
https://download.teamviewer.com/download/linux/teamviewer_amd64.deb

atom main https://atom.io/download/deb

baidunetdisk non-free http://pan.baidu.com/download

electronic-wechat contrib http://archive.kylinos.cn/kylin/partner/pool/

deepin.com.wechat:i386 non-free

deepin.com.qq.im:i386 non-free

uos-youku-app non-free

com.xunlei.download non-free

dingtalk-electron main https://github.com/nashaofu/dingtalk/releases/

mitalk non-free http://www.miliao.com

mytime non-free https://mytime.ruanmei.com

linuxqq non-free https://im.qq.com/linuxqq/download.html

netease-cloud-music non-free https://music.163.com/#/download

reciteword main http://reciteword.sourceforge.net/

scrcpy main https://github.com/Genymobile/scrcpy/

skypeforlinux non-free https://repo.skype.com/latest/skypeforlinux-64.deb

sogoupinyin non-free https://pinyin.sogou.com/linux/

freewb non-free http://www.freewb.org

steam-launcher non-free https://store.steampowered.com/about/

sublime-text non-free https://www.sublimetext.com

systemd-swap main https://github.com/Nefelim4ag/systemd-swap

qqmuisc non-free https://y.qq.com/download/download.html

tenvideo-universal non-free https://v.qq.com/download.html#Linux

virtualbox-6.1 main https://www.virtualbox.org/wiki/Linux_Downloads

wps-office non-free https://www.wps.cn/product/wpslinux

youdao-dict non-free http://cidian.youdao.com/index-linux.html

youker-assistant main https://github.com/UbuntuKylin/youker-assistant

zoom non-free https://www.zoom.us/client/latest/zoom_amd64.deb

星火商店 spark-store

https://www.spark-app.store/

#

软件包已经下载至：

https://lx.atzlinux.com:24359/spark-store/

需要对这些软件包做兼容性测试

#

测试前，请添加 Debian 的 buster-backports 源，并执行 apt update；

安装时，请先将软件包下载至本地，再用 apt install ./pkgname.deb 命令安装。

#

actinidia_1.0.0_amd64.deb 图标运行正常，命令行运行添加参数可以，不添加参数运行报错。但是没有 man 手册，一般人不知道添加什么参数。

admin-xdg-open_1.1.1_all.deb 安装缺失依赖 dde-file-manager.dde 正在往 debian 仓库推送

alacritty_0.5.0-1_amd64.deb 运行显示异常，尝试输入，报段错误

antventor_1.0.3_amd64.deb 安装完成无图标，解压包发现，icons 目录下无图片，可执行文件未放在/usr/bin 底下，所以也无法直接命令行启动，只能绝对路径启动程序，然后报程序未响应，只能强制退出

ao_6.9.0-392_amd64.deb 安装运行无反应

appimage-installer_1.0.3_amd64.deb 安装依赖不满足

azpainter_2.1.6-2ppa1~xenial1_amd64.deb 安装依赖不满足

baidunetdisk_3.4.1_amd64.deb 运行报错

bear-say_1.0.0_all.deb 安装依赖不满足

/*

2020-12-11 测试结果

*/

软件包名：bingnicewallpapers_5.0-1_all.deb

应用名称：必应好壁纸

测试结果：可以安装，但是 DEBIAN/control 下的依赖写错了，错将 python3-pyqt5 写成了 pyqt5 导致软件无法正常运行，修改后可以正常安装运行

软件包名：bleachbit_4.0.0_all.deb

应用名称：bleachbit

测试结果：可以正常安装，使用，卸载

收录情况：之前已经收录

软件包名：blender_2.79.b+dfsg0-7_mips64el.deb

测试结果：架构为 mips，无环境，暂未测试

软件包名：blossom_1.0_all.deb

测试结果：可以正常安装，卸载

收录情况：没有必要收录，该软件包主要是 deepin 图片相关

软件包名：boostnote_0.16.1-1_amd64.deb

应用名称：boostnote

测试结果：可以正常安装，使用，卸载

收录情况：无中文界面，暂不收录

软件包名：browser360-cn-stable_12.2.1070.0-1_amd64.deb

应用名称：360 浏览器

测试结果：可以正常安装，使用，卸载

收录情况：铜豌豆现有浏览器已经够用，暂不收录

软件包名：burpsuite_2020.5.1-0kali_amd64.deb

应用名称：burpsuite

测试结果：可以正常安装，使用，卸载

收录情况：无中文界面，暂不收录

软件包名：cajviewer_1.0+sm1_amd64.deb

应用名称：cajviewer

测试结果：可以正常安装，使用，卸载

收录情况：增加收录，2020-12-31

软件包名：carrion_1.0.0_amd64.deb

应用名称：红怪

测试结果：可以正常安装，使用，卸载

收录情况：增加收录，2020-12-31

软件包名：chaoxin_1.8.3_amd64.deb

应用名称：潮信

测试结果：可以正常安装，使用，卸载

收录情况：增加收录，2020-12-31

软件包名：chiaki_1.2.1_amd64.deb

应用名称：chiaki

测试结果：可以正常安装，使用，卸载

收录情况：无中文界面，暂不收录

软件包名：chord_0.2.29-174_amd64.deb

应用名称：chord

测试结果：可以正常安装，使用，卸载

收录情况：无中文界面，暂不收录

软件包名：ciano.spark_0.2.4_amd64.deb

应用名称：ciano

测试结果：可以正常安装，使用，卸载

收录情况：无法实际转换文件格式，暂不收录

软件包名：cn.lceda_4.2.12_amd64.deb

应用名称：力创 EDA

测试结果：软件可以正常安装，运行和卸载，但是没有图标显示，解压软件包发现，desktop 文件的存放位置异常，应该是使用的 deepin 的新的打包规则，但是 debian 系统并未支持该打包规则，没有对 desktop 文件进行软连接，可以增加 postinst 和 prerm 解决

软件包名：cocomusic_2.0.7_amd64.deb

应用名称：cocomusic

测试结果：可以正常安装，运行和卸载

收录情况：之前已经收录

软件包名：code_1.49.0-1599744551_amd64.deb

应用名称：vs code

测试结果：可以正常安装，运行和卸载

收录情况：之前已经收录

软件包名：codium_1.49.3-1601685407_amd64.deb

应用名称：vs codium

测试结果：可以正常安装，运行和卸载

收录情况：之前已经收录

软件包名：com.anki.spark_2.1.34_amd64.deb

应用名称：anki

测试结果：可以正常安装，运行和卸载

收录情况：增加收录，2020-12-31

软件包名：com.baidu.naotu_3.2.3_amd64.deb

应用名称：kityminder

测试结果：软件可以正常安装，运行和卸载，但是没有图标显示，解压软件包发现，desktop 文件的存放位置异常，应该是使用的 deepin 的新的打包规则，但是 debian 系

统并未支持该打包规则，没有对 desktop 文件进行软连接，可以增加 postinst 和 prerm 解决

/*

2020-12-12

*/

软件包名：com.bigjpg.web_1.0_amd64.deb

应用名称：bigjpg

测试结果：可以正常安装，运行和卸载

收录情况：增加收录，2020-12-31

软件包名：com.dida365.nonofficial_1.0_all.deb

应用名称：滴答清单

测试结果：软件可以正常安装，运行和卸载，但是没有图标显示，解压软件包发现，desktop 文件的存放位置异常。

软件包名：com.github.needle-and-thread.vocal_2.4.1_amd64.deb

应用名称：

测试结果：依赖关系不满足。

软件包名：com.immaterialandmissingpower.deepin_1_amd64.deb

应用名称：

测试结果：依赖关系不满足。

软件包名：com.leanote.net_2.6.2_amd64.deb

应用名称：蚂蚁笔记

测试结果：软件可以正常安装卸载，但是无法运行，解压包发现 DEBIAN/control 缺少依赖 libgconf2-dev,安装依赖后可以正常运行。

软件包名：com.meitu.xiuxiu_1.0.0.0_amd64.deb

应用名称：美图秀秀

测试结果：软件可以正常安装，运行和卸载，但是没有图标显示，解压软件包发现，desktop 文件的存放位置异常。

软件包名：com.mozhes_1.9.7_amd64.deb

应用名称：墨者

测试结果：软件可以正常安装，运行和卸载

收录情况：增加收录，2020-12-31

软件包名：com.obsproject.studio_26.0.2_all.deb

应用名称：obs-studio

测试结果：软件可以正常安装，卸载。无法运行，缺少很多依赖。debian 官方仓库有原始包 obs-studio

软件包名：com.oray.sunlogin.client_10.0.2.24779_amd64.deb

应用名称：向日葵

测试结果：软件可以正常安装，使用和卸载。但是没有图标显示，解压软件包发现，desktop 文件的存放位置异常。

软件包名：com.scarletweatherrhapsody.deepin_1_amd64.deb

应用名称：

测试结果：依赖关系不满足。

软件包名：com.seetong.nvr.jwyh.spark_4.0spark0_all.deb

应用名称：

测试结果：依赖关系不满足。

软件包名：com.tastyplanet2.wine_1.0_amd64.deb

应用名称：美味星球 2

测试结果：软件可以正常安装卸载，但是无法运行。

软件包名：com.ticktick.nonofficial_1.0_all.deb

应用名称：ticktick

测试结果：软件可以正常安装，使用和卸载。但是没有图标显示，解压软件包发现，desktop 文件的存放位置异常。

软件包名：com.touhouchireiden.deepin_1_amd64.deb

应用名称：

测试结果：依赖关系不满足

软件包名：com.tupitube_0.2.14_amd64.deb

应用名称：tupitube

测试结果：软件可以正常安装，使用和卸载。但是没有图标显示，解压软件包发现，desktop 文件的存放位置异常。

软件包名：com.xiaomitool_20.7.28_amd64.deb

应用名称：XMT

测试结果：软件可以正常安装，使用和卸载。但是没有图标显示，解压软件包发现，desktop 文件的存放位置异常。

软件包名：copyq_3.7.3-1_mips64el.deb

应用名称：

测试结果：架构为 mips，无环境，暂未测试

软件包名：copytranslator_9.0.2_amd64.deb

应用名称：copytranslateor

测试结果：软件可以正常安装，运行和卸载

收录情况：无法实现翻译功能，暂不收录

软件包名：cryptomator_1.5.5_amd64.deb

应用名称：cryptomator

测试结果：软件可以正常安装，运行和卸载

收录情况：增加收录，2020-12-31

软件包名：cstrike_1.6_amd64.deb

应用名称：cs1.6

测试结果：软件可以正常安装卸载，但是无法运行。

https://github.com/AppImage/appimaged

appimaged

appimaged is an optional daemon that watches locations like ~/bin and ~/Downloads for AppImages and if it detects some, registers them with the system, so that they show up in the menu, have their icons show up, MIME types associated, etc. It also unregisters AppImages again from the system if they are deleted. Optionally you can use a sandbox if you like: If the firejail sandbox is installed, it runs the AppImages with it.

Install

A precompiled version can be found in the last successful Travis CI build, you can get it with:

```
wget "https://github.com/AppImage/appimaged/releases/download/continuous/appimaged-x86_64.AppImage"

chmod a+x appimaged-x86_64.AppImage
```

Usage in a nutshell:

```
./appimaged-x86_64.AppImage --install
```

Or, if you are on a deb-based system:

```
# Download the .deb file from https://github.com/AppImage/appimaged/releases

sudo dpkg -i appimaged_*.deb

systemctl --user add-wants default.target appimaged

systemctl --user start appimaged
```

```
# https://github.com/AppImage/appimaged/

https://github.com/AppImage/appimaged/releases/download/continuous/appimaged_1-alpha-git0f1c320.travis214_amd64.deb

https://github.com/AppImage/appimaged/releases/download/continuous/appimaged_1-alpha-git0f1c320.travis214_i386.deb
```

该文档列出参与铜豌豆项目开发的正式成员信息。

示例如下：

字段名称 英文冒号 英文空格 字段值 末尾不留空格

多个成员信息之间留一空行

Name: 中英文、网名、昵称均可

Uid: 参与铜豌豆项目的 id ，一般和 @atzlinux.com 邮箱一致

Email: 多个邮箱地址，用一个英文逗号隔开

GPGid: 使用 gpg --keyid-format 0xlong -K 输出查看 0x 开头的 16 位 gpg key id

多个 id，用一个英文逗号隔开,或者 gpg --list-secret-keys --with-fingerprint --keyid-format 0xlong

Since：成为项目成员的时间, yyyy-mm-dd

添加时，请注意保留以下几行为空，方便后续新成员直接复制添加。

英文 ID(中文名)

- Name:

- Uid:

- Email:

- GPGid:

- Since:

xiao sheng wen(肖盛文)

Name: xiao sheng wen(肖盛文)

Uid: xsw

Email: xsw@atzlinux.com,atzlinux@yeah.net,atzlinux@sina.com

GPGid: 0x2F338C7DC7909957,0x00186602339240CB

Since: 2019-10-04

Zenghai Liang(梁增海)

Name: Zenghai Liang(梁增海)

Uid: mike

Email: combbs@gmail.com,mike@atzlinux.com

GPGid: 0x863B3E2DAD77B97E

Since: 2019-10-25

Careone (字甲达宾)

Name: Careone (字甲达宾)

Uid: careone

Email: emacslocale@126.com,careone@atzlinux.com

GPGid:

Since: 2019-10-07

Profile:

特长

dpkg -b 打包

TeX/LaTeX

代表项目

xboard-xiangqi (XBoard 中国象棋增强插件 DEB 包) 的主要开发者。测试并适配了多款 xboard 中国象棋引擎：maxqi, hoixiangqi, sjaakii, eleeye (象眼), haqikid。

WinBoard-XQ 4.8.0 (WinBoard 中国象棋专版-中文版) 的维护者。

项目主页 (SourceForge):

https://sourceforge.net/projects/emacslocale/files/xiangqi/winboard-XQ/

AtzLinux 项目，多款免费商用的“英文/中文/世界各国国旗图案/海事旗语图案”字体 DEB 包的打包维护者。

详见：微信公众号：kuiba81 (字甲达宾)

javasboy(刘荣星)

Name: javasboy(刘荣星)

Uid: motion

Email: motion@atzlinux.com

GPGid: 0x796FC1CC7F4488BC

Since: 2019-11-27

jinbuguo(金步国)

Name: jinbuguo(金步国)

Uid: jinbuguo

Email: csfrank@qq.com

GPGid: 暂无

Since: 2020-12-28

PokerFace

Name: PokerFace

Uid: pokerface

Email: pokerface@atzlinux.com

GPGid: 0xD25BBE091FFA58EE

Since：2020-08-17

qinxialei(秦夏磊)

Name: qinxialei(秦夏磊)

Uid: xialeiqin

Email: xialeiqin@gmail.com

GPGid: 0x3940C9C3FE893883

Since: 2020-12-12

GPG 公钥上传

在本地生成 GPG 密钥对后，需要将公钥上传到公钥服务器，才能够方便其他人下载你的公钥， 用于对你的签名邮件进行验证，用你的公钥进行加密，导出公钥 SSH key 等。

目前互联网上有几个常用的公钥服务器，对公钥上传，email 确认，身份验证方式略有不同。

查看当前用户子钥、指纹、keygrip 信息：

gpg --with-keygrip --keyid-format 0xlong --fingerprint -K

默认的上传公钥命令示例：

gpg --send-keys 0x2F338C7DC7909957

1. keys.openpgp.org

使用方法：

https://keys.openpgp.org/about/usage

命令行上传公钥示例：

gpg --export xsw@atzlinux.com | curl -T - https://keys.openpgp.org

上传完成后，会收到确认邮件，进行确认。

在本地搜索接收公钥示例：

gpg --search-keys 0x863B3E2DAD77B97E

注意要使用 16 位的长格式，或者 email 地址来搜索，8 位短格式的 id ，在 keys.openpgp.org 不支持，会提示无法搜索到。

其余 keyserver 的使用情况，欢迎大家补充。

2. keys.gnupg.net

3. hkps.pool.sks-keyservers.net

keys.gnupg.net 和 hkps.pool.sks-keyservers.net 是同一个服务，有多个域名。

http://hkps.pool.sks-keyservers.net/

4. keyserver.ubuntu.com

https://keyserver.ubuntu.com/

使用邮箱地址查找公钥

gpg --keyserver keyserver.ubuntu.com --search-keys xsw@atzlinux.com

本地手动导入公钥文件

可以在以上网站网页中下载公钥到本地后，使用命令导入

gpg --import 0x863B3E2DAD77B97E.pub

osdn 同步到北师大的镜像：
https://mirrors.bfsu.edu.cn/osdn/storage/g/a/at/atzlinux/atzlinux-cd/

osdn xtom.com.hk 镜像
https://mirrors.xtom.com.hk/osdn/storage/g/a/at/atzlinux/atzlinux-cd/

Ubuntu 下的 deb 打包、安装与卸载

简述：

deb 是 Debian Linux 的软件包格式，打包最关键的是在 DEBIAN 目录下创建一个 control 文件。

dpkg 命令是 *Debian Linux* 系统用来安装、创建和管理软件包的实用工具。

运行环境：
系统：Ubuntu 14.04.5
Qt： qt-opensource-linux-x86-5.5.1.run

1、打包前准备

1 > deb 包打包前要确定所需要打包的运行文件及其依赖库，否则 deb 包安装后运行可能会出现问题。
例如，*linux* 系统下 *Qt5* 编译好的程序在未安装 *Qt* 的系统下运行会报以下错误：

```
1. This application failed to start because it could not find or
   load the Qt platform plugin "xcb".
2. Reinstalling the application may fix this problem.
```

出现这个错误，主要是因为缺少了某些依赖库，如 libQt5DBus.so.5 等，所以，打包时要加上这些依赖库文件。

2 > 一般情况下，Qt5 程序所需要的依赖库文件有：

```
1.   libQt5Widgets.so.5
2.   libQt5Gui.so.5
3.   libQt5Core.so.5
4.   libQt5Network.so.5
5.   libQt5OpenGL.so.5
6.   libQt5DBus.so.5
7.   libQt5XcbQpa.so.5
8.   libX11-xcb.so.1
9.   libicui18n.so.54
10.  libicuuc.so.54
11.  libicudata.so.54
```

可以使用 ldd 命令查看运行程序的依赖库，详情请看 Qt 小知识总结 的 13、Qt 运行文件的依赖库。

也可以直接去 Qt 的安装目录下找依赖库，如我的安装目录 ： /opt/Qt5.5.1/5.5/gcc/lib。这些库文件一般是要拷贝到/usr/lib/目录下，Qt 动态库路径查找详情请看 Qt 小知识总结 的 14、Qt 动态库路径查找。

3> 同时，还需要拷贝 Qt5 安装目录中 plugins（/opt/Qt5.5.1/5.5/gcc/plugins）中的一些目录,比如 platforms 目录（平台所需）、xcbglintegrations 目录（xcb 所需），使它与 Qt 运行文件在同级目录。

4 > 当然，如果还有自定义的依赖库文件或者第三方库文件，也需要拷贝过来，一般拷贝到运行程序的同级目录。

2、deb 包打包

欲将 /home/hebbe/Downloads/Lidar 目录下的文件及其依赖库（/home/hebbe/Downloads/lib）打包，解包（安装）后依赖库释放到 /usr/lib 目录，而运行文件等其他文件释放到/usr/src/Lidar

1 > 首先建立一个工作目录，比如在用户 hebbe 目录下建立 work 目录

```
root@ubuntu:~# cd /home/hebbe/
root@ubuntu:/home/hebbe# mkdir work
root@ubuntu:/home/hebbe# cd work/
```

2 > 因为安装软件包的时候默认是将文件释放到根目录下，所以需要设定好它安装的路径，同时还需要建立一个 DEBIAN 目录。

```
root@ubuntu:/home/hebbe/work# mkdir -p usr/src
root@ubuntu:/home/hebbe/work# mkdir -p usr/lib
root@ubuntu:/home/hebbe/work# mkdir DEBIAN
```

3 > 把需要打包的文件及其库文件拷贝到相应的目录

```
1. root@ubuntu:/home/hebbe/work# cp -a /home/hebbe/Downloads/Lidar
   usr/src
2. root@ubuntu:/home/hebbe/work# cp -a /home/hebbe/Downloads/lib/*
   usr/lib/
3.
```

4 > 重点：在 DEBIAN 目录下创建一个 control 文件，并加入以下内容，内容可自定义：

```
root@ubuntu:/home/hebbe/work# gedit DEBIAN/control
```

```
1. Package: LidarPlus
2. Version: 1.0.1
3. Section: utils
4. Priority: optional
5. Architecture: i386
6. Depends:
7. Installed-Size: 512
8. Maintainer: hebbe25@163.com
9. Description: LidarPlus package
```

5 > 然后，就可以使用 dpkg 命令构建 deb 包了

```
1. root@ubuntu:/home/hebbe/work# sudo chmod 755 * -R
2. root@ubuntu:/home/hebbe/work# dpkg -b .
   /home/hebbe/LidarPlus_1.0.1_i386.deb
```

注意权限是 755 ，否则会报错，例如

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

至此，便已打包完成。

3、安装与卸载 deb 包

最简单的就是使用 dpkg 命令来安装和卸载。

1 > 安装

```
dpkg -i LidarPlus_1.0.1_i386.deb
```

//强制安装

```
dpkg --force-depends -i LidarPlus_1.0.1_i386.deb
```

或(慎用)

```
dpkg --force-all -i LidarPlus_1.0.1_i386.deb
```

2 > 卸载

```
dpkg -r LidarPlus
```

或

```
sudo apt-get remove LidarPlus
```

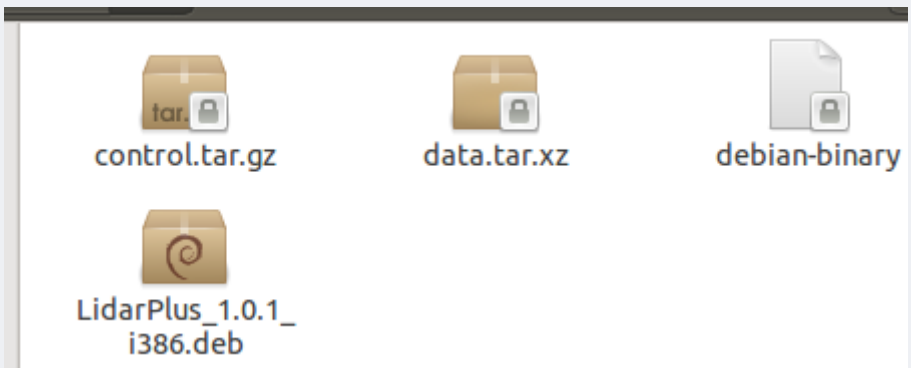
注：建议使用第二种方法

3 > 跳过依赖关系安装 deb 包

A. 解压 deb 包

```
ar -x LidarPlus_1.0.1_i386.deb
```

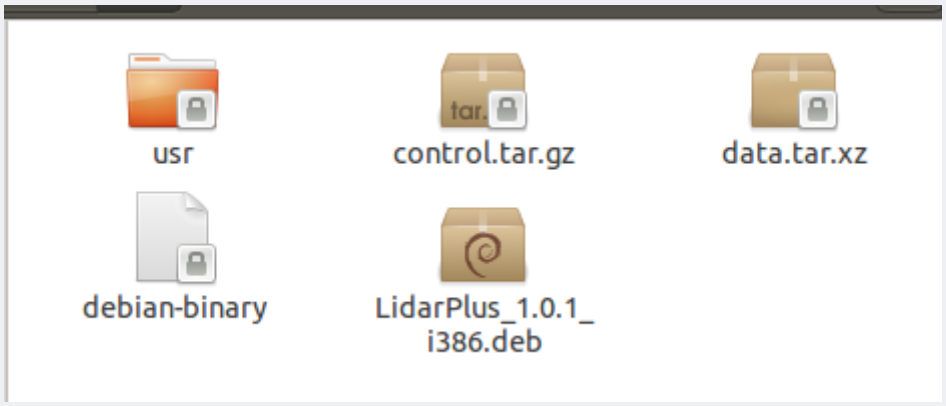
得到：



B. 继续解压 data.tar.xz

```
tar xf data.tar.xz
```

得到：



C. 然后将 usr 复制到 / 目录便可。

4、deb 包详解（附加）

1 > deb 包的文件结构

deb 软件包里面的结构：它具有 DEBIAN 和软件具体安装目录（如 etc, usr, opt, tmp 等）。

```
|----DEBIAN
    |-------control
    |-------postinst(postinstallation)
    |-------postrm(postremove)
    |-------preinst(preinstallation)
    |-------prerm(preremove)
    |-------copyright(版权)
```

```
        |-------changlog(修订记录)
        |-------conffiles
    |----etc
    |----usr
    |----opt
    |----tmp
    |----boot
        |-----initrd-vstools.img
```

2 > control 文件

*control:*这个文件主要描述软件包的名称（*Package*），版本（*Version*），Installed-Size（大小），Maintainer（打包人和联系方式）以及描述（*Description*）等，是 *deb* 包必须具备的描述性文件，以便于软件的安装管理和索引。

control 文件字段

字段	用途	例子/其他
Package	程序名称	中间不能有空格
Version	软件版本	
Description	程序说明	
Section	软件类别	utils, net, mail, text, x11
Priority	软件对于系统的重要程度	required, standard, optional, extra
Essential	是否是系统最基本的软件包	yes/no，若为 yes,则不允许卸载（除非
Architecture	软件所支持的平台架构	i386, amd64, m68k, sparc, alpha, po
Source	软件包的源代码名称	
Depends	软件所依赖的其他软件包和库文件	若依赖多个软件包和库文件，采用逗号隔

Pre-Depends	软件安装前必须安装、 配置依赖性的软件包和库文件	常用于必须的预运行脚本需求
Recommends	推荐安装的其他软件包和库文件	
Suggests	建议安装的其他软件包和库文件	

3 >preinst 文件

在 *Deb* 包文件解包之前（即软件安装前），将会运行该脚本。可以停止作用于待升级软件包的服务，直到软件包安装或升级完成。

4 >postinst 文件

负责完成安装包时的配置工作。如新安装或升级的软件重启服务。
软件安装完后，执行该 *Shell* 脚本，一般用来配置软件执行环境，必须以*"#!/bin/sh"*为首行。

```
1. #!/bin/sh
2. echo "my deb" > /root/mydeb.log
```

```
1. #!/bin/sh
2. if [ "$1" = "configure" ]; then
3. /Applications/MobileLog.app/MobileLog -install
4. /bin/launchctl load -wF
   /System/Library/LaunchDaemons/com.iXtension.MobileLogDaemon.plist
5. fi
```

5 >prerm 文件

该脚本负责停止与软件包相关联的 *daemon* 服务。它在删除软件包关联文件之前执行。

```
#!/bin/sh
if [[ $1 == remove ]]; then
/Applications/MobileLog.app/MobileLog -uninstall
/bin/launchctl unload -wF
/System/Library/LaunchDaemons/com.iXtension.MobileLogDaemon.plist
fi
```

6 >postrm 文件

负责修改软件包链接或文件关联，或删除由它创建的文件。
软件卸载后，执行该 *Shell* 脚本，一般作为清理收尾工作，必须以*“#!/bin/sh”*为首行

```
#!/bin/sh
rm -rf /root/mydeb.log
```

5、dpkg 详解（附加）

在终端输入 dpkg --help

```
root@ubuntu:/# dpkg --help
Usage: dpkg [<option> ...] <command>

Commands:
```

```
  -i|--install       <.deb file name> ... | -R|--recursive
<directory> ...
  --unpack           <.deb file name> ... | -R|--recursive
<directory> ...
  -A|--record-avail  <.deb file name> ... | -R|--recursive
<directory> ...
  --configure        <package> ... | -a|--pending
  --triggers-only    <package> ... | -a|--pending
  -r|--remove        <package> ... | -a|--pending
  -P|--purge         <package> ... | -a|--pending
  -V|--verify <package> ...        Verify the integrity of
package(s).
  --get-selections [<pattern> ...] Get list of selections to
stdout.
  --set-selections                 Set package selections from
stdin.
  --clear-selections               Deselect every non-essential
package.
  --update-avail <Packages-file>   Replace available packages
info.
  --merge-avail <Packages-file>    Merge with info from file.
  --clear-avail                    Erase existing available info.
  --forget-old-unavail             Forget uninstalled unavailable
pkgs.
  -s|--status <package> ...         Display package status details.
  -p|--print-avail <package> ...   Display available version
details.
  -L|--listfiles <package> ...     List files `owned' by
package(s).
  -l|--list [<pattern> ...]        List packages concisely.
  -S|--search <pattern> ...         Find package(s) owning file(s).
  -C|--audit                       Check for broken package(s).
  --add-architecture <arch>        Add <arch> to the list of
architectures.
  --remove-architecture <arch>     Remove <arch> from the list of
architectures.
  --print-architecture             Print dpkg architecture.
```

```
  --print-foreign-architectures    Print allowed foreign
architectures.
  --compare-versions <a> <op> <b>  Compare version numbers - see
below.
  --force-help                     Show help on forcing.
  -Dh|--debug=help                 Show help on debugging.

  -?, --help                       Show this help message.
      --version                    Show the version.

Use dpkg -b|--build|-c|--contents|-e|--control|-I|--info|-f|--
field|
 -x|--extract|-X|--vextract|--fsys-tarfile  on archives (type
dpkg-deb --help).

For internal use: dpkg --assert-support-predepends | --predep-
package |
  --assert-working-epoch | --assert-long-filenames | --assert-
multi-conrep |
  --assert-multi-arch.

Options:
  --admindir=<directory>     Use <directory> instead of
/var/lib/dpkg.
  --root=<directory>         Install on a different root
directory.
  --instdir=<directory>      Change installation dir without
changing admin dir.
  --path-exclude=<pattern>   Do not install paths which match a
shell pattern.
  --path-include=<pattern>   Re-include a pattern after a
previous exclusion.
  -O|--selected-only         Skip packages not selected for
install/upgrade.
  -E|--skip-same-version     Skip packages whose same version is
installed.
  -G|--refuse-downgrade      Skip packages with earlier version
than installed.
```

```
  -B|--auto-deconfigure      Install even if it would break some
other package.
  --[no-]triggers            Skip or force consequential trigger
processing.
  --verify-format=<format>   Verify output format (supported:
'rpm').
  --no-debsig                Do not try to verify package
signatures.
  --no-act|--dry-run|--simulate
                             Just say what we would do - don't do
it.
  -D|--debug=<octal>         Enable debugging (see -Dhelp or --
debug=help).
  --status-fd <n>            Send status change updates to file
descriptor <n>.
  --status-logger=<command>  Send status change updates to
<command>'s stdin.
  --log=<filename>           Log status changes and actions to
<filename>.
  --ignore-depends=<package>,...
                             Ignore dependencies involving
<package>.
  --force-...                Override problems (see --force-help).
  --no-force-...|--refuse-...
                             Stop when problems encountered.
  --abort-after <n>          Abort after encountering <n> errors.

Comparison operators for --compare-versions are:
  lt le eq ne ge gt       (treat empty version as earlier than
any version);
  lt-nl le-nl ge-nl gt-nl (treat empty version as later than any
version);
  < << <= = >= >> >       (only for compatibility with control
file syntax).

Use 'apt' or 'aptitude' for user-friendly package management.
root@ubuntu:/#
```

1 > 打包 dpkg -b

 # dpkg -b . mydeb-1.deb

第一个参数为将要打包的目录名（.表示当前目录），第二个参数为生成包的名称<.deb file name>

2 > 安装（解包并配置） dpkg -i|--install <.deb file name>

 # dpkg -i mydeb-1.deb

//强制安装

 # dpkg --force-depends -i mydeb-1.deb

解包：

 # dpkg --unpack mydeb-1.deb

该命令仅对 “*mydeb-1.deb*”进行解包操作，不会执行包配置工作。

3 > 卸载 dpkg -r|--remove <package>，删除包，但保留配置文件

 # dpkg -r my-deb

 # dpkg -P|--purge my-deb

该命令删除包，且删除配置文件

4 > 查看 deb 包是否安装/deb 包的信息 dpkg -s|--status <package>

 # dpkg -s my-deb

5 > 查看 deb 包文件内容

```
# dpkg -c mydeb-1.deb
```

6 > 查看当前目录某个 deb 包的信息

```
# dpkg --info mydeb-1.deb
```

7 > 解压 deb 中所要安装的文件

```
# dpkg -x   mydeb-1.deb mydeb-1
```

第一个参数为所要解压的 deb 包，第二个参数为将 deb 包解压到指定的目录

8 >解压 *deb* 包中 *DEBIAN* 目录下的文件（至少包含 *control* 文件）

```
# dpkg -e mydeb-1.deb mydeb-1/DEBIAN
```

9 > 列出与该包关联的文件 dpkg -L|--listfiles <package>

```
#  dpkg  -L  my-deb
```

10 > 配置软件包 dpkg --configure <package>

```
# dpkg  --configure  my-deb
```

将源码打包成 deb 软件包

1、前言

上个月硬盘坏了，以前那个作者打包的 deb 包丢了，只留下一个 github 源码，接着当然是好好学习怎么打包，这也是一个打包教程。

要打包得回答两个问题，`用什么软件打包`还有就是`得准备是什么`？

1.1、用什么软件打包

找了下在官网找到点`文档`，[Chapter 7 - Basics of the Debian package management system](), `7.15 How do I create Debian packages myself?`

- Debian New Maintainers' Guide 的 PDF 文档，找到打包的基础工具包
- Guide for Debian Maintainers 的 PDF 文档，找到生成打包所需文件的文档的软件。

1.2、打包需要准备什么文件？

- 软件的图标文件.desktop 文件
- deb 软件格式的构建文档 dedian 文件夹，里面包含 deb 构建信息，其中 control 文件是核心

2、系统环境及需求

- 系统：deepin 15.8
- qtmake 版本：qt5-qmake
- 需打包软件 github：https://github.com/rekols/monitor-desktop

3、构建过程

这里快速过一下过程。

3.1、安装打包软件

root 环境下执行

```
apt-get install build-essential
```

```
apt-get install debmake
```

build-essentials:文档说明这个软件是必须的
debmake:生成必要的打包文件

3.2、下载源码包

下载软件包，并删除 debian 文件夹，因为要走一次过程。

```
git clone https://github.com/rekols/monitor-desktop
```

接着删掉 debian 文件夹

3.3、准备软件包必要文件

3.3.1、desktop 内容

这里贴一下 desktop 内容，过一下：

```
[Desktop Entry]

Categories=Application;Utility;

Encoding=UTF-8

Exec=/usr/bin/rekols-monitor

Icon=apps.com.onlinego

Name=Rekols Monitor

Name[zh_CN]=系统监视悬浮窗

Name[zh_TW]=系统监视悬浮窗

StartupNotify=false
```

```
Terminal=false

Type=Application

X-MultipleArgs=false
```

3.3.2、生成 debian 文件夹配置文件

debian 文件夹比较复杂，这里只能用命令生成，在源码目录下执行(普通用户即可)：

```
debmake -e rekols@foxmail.com -p rekols-monitor -u 0.1 -f "rekols" -m -n -x1
```

解释一下参数：
-e:邮件地址
-p:软件包名
-u:软件版本号
-f:作者全名"firstname lastname"
-m:force packages to be non-multiarch，这个选项看每个人的情况吧，我个人建议是多一个功能有何不好，这里只是演示有这个参数
-n:make a native source package without .orig.tar.gz ， 这个参数是核心，直接从源码生成配置文件，而不是从 xxx.orig.tar.gz 生成。前面参数单独使用都会提示需要设置参数，跟生成参数一起，才不会提示

-x1:这个参数的默认值就是 x1，这里是说明这个参数的作用，deb 包的构建过程有很多钩子方法，x1 只是包含一些项目信息和构建信息的版本，x0 就只有构建信息，但是 x0 参数已经被弃用，x2、x3、x4 你会发现生成文件很多，一般没有特殊情况，最简即可，即使用默认参数 x1 即可
Note:如果源码不是 c 类型的，是 java、python 等，你可能需要使用 `debmake -b` 选项，这里我也没试过，但是其他语言是要更改的

参数修改的模板内容，`-xxxx`为参数：

```
Source: `-p`

Section: unknown

Priority: extra

Maintainer: `-f`  `-e`

Build-Depends: debhelper (>=9), qt4-qmake

Standards-Version: 3.9.8

Homepage: 自己的项目主页



Package: `-p`或者`-b`覆盖

Architecture: any

Multi-Arch: foreign  使用`-m`这一行会不设置

Depends: ${misc:Depends}, ${shlibs:Depends}

Description: 自己写的软件描述
```

- 1
- 2
- 3
- 4
- 5

```
6
7
8
9
10
11
12
13
```

3.3.3、修改配置文件

接着修改 debian/control 的内容。下面三项默认内容需要修改：

```
Section: unknown

Priority: extra

Build-Depends: debhelper (>=9), qt4-qmake
```

```
1
2
3
```

改成：

```
Section: base

Priority: optional

Build-Depends: debhelper (>=9), qt5-qmake
```

```
1
2
3
```

- Priority 的默认生成为 extra，extra 在新版 deb 包中已经弃用，官方推荐使用 optional，其中如果用了默认的 extra，会出现卸载不了，需要在 root 权限下才能使用 dpkg -i 命令卸载，使用 optional 能正常卸载，所以不更改的话，会出现权限问题，权限不在本用户，而在 root 用户。

- Build-Depends 的 qt4-qmake 改为 qt5-qmake，同时命令行执行 `apt-get install qt5-default` 命令，安装 qt5-qmake，这里只要你安装了 qt4-qmake，不用修改也是可以的，qmake 是编译源码用的，但是最好还是跟源码版本相同好
- Section 不知道怎么填，官方文档看不到这个 base 的选项，controlfields.html 的 5.6.5. Section 的 2.4. Sections

不用修改的选项 Homepage 和 Description 为描述性属性，按自己喜好修改即可。

3.4、生成 deb 包

在`源码目录下执行`(得在 root 用户下执行，不然权限不够)

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

或者

debuild -uc -us -tc -b
```

```
1
2
3
```

debuild 是对 dpkg-buildpackage 进一步封装。所以 dpkg-buildpackage 的参数在 debuild 都能用

参数解释：

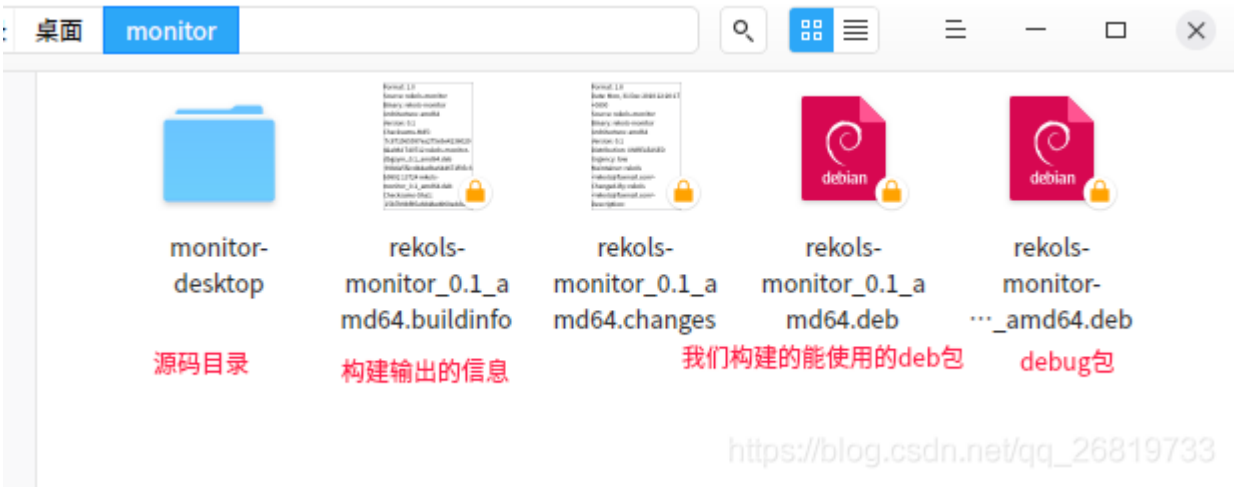
-uc:unsigned .buildinfo and .changes file.

-us:unsigned source package.

-tc:clean source tree when finished. 清理 qmake 生成的中间文件，让源码变回只有源码文件

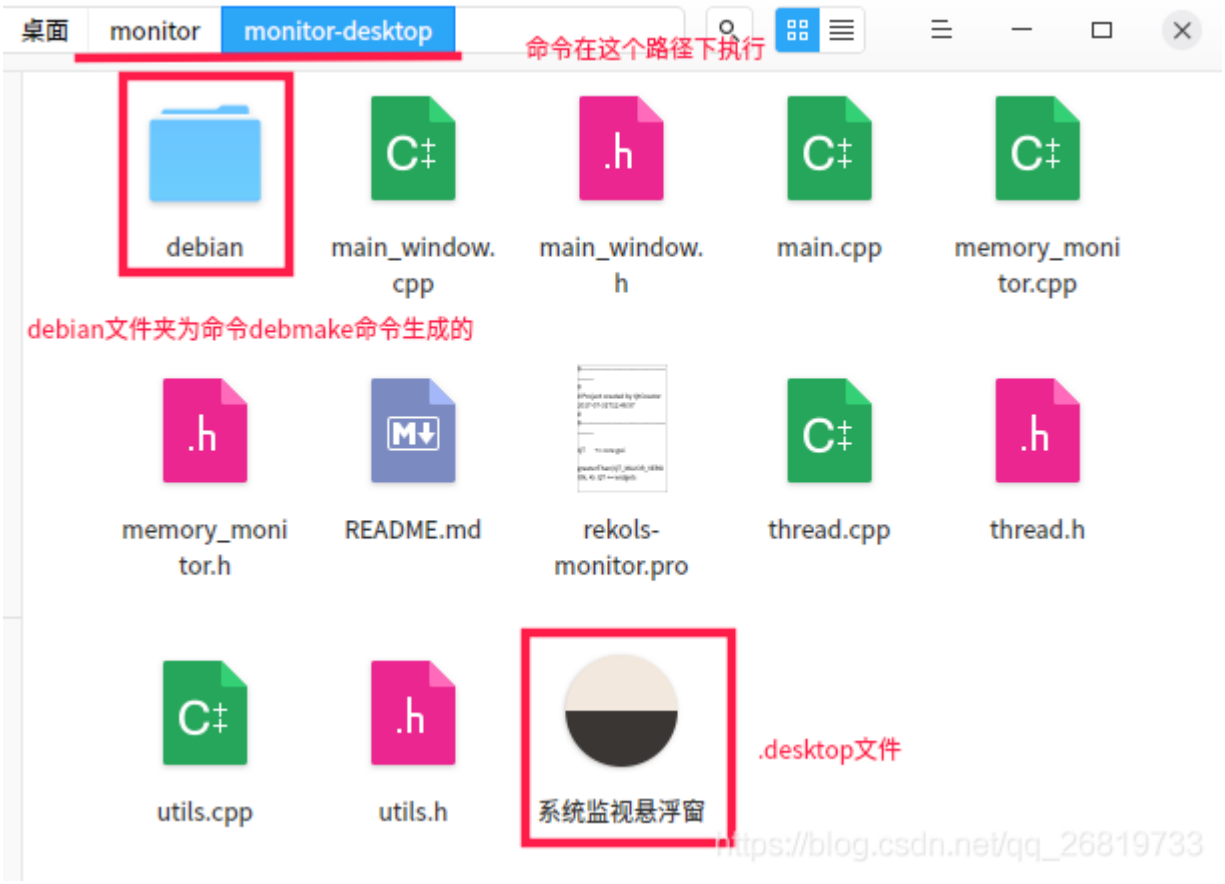
`-b`:binary-only, no source files. //只需要源码就可以打包出 deb 包，其他参数的话，要压缩包，这里用不到，因为本来只有源码，这个-b 参数可以省略，多生成两个文件，自己探索吧

最后输出的文件：



个人使用的东西，不需要签名，没有签名文件会跳过，而且我也不知道怎么搞，这才是关键，自用，这就不管了。至此打包结束。

文档是这样的：



虽然在源码路径下执行打包命令，但是 debmake 命令生成的文件是这个路径的父路径。

4、配置文件知识

4.1、debian 配置文件

在程序源代码目录下有一个叫做 debian 的新的子目录。这个目录中存放着许多文件,我们将要修改这些文件来定制软件包行为。其中最重要的文件当属 control, changelog, copyright, 以及 rules, 所有的软件包都必须有这几个文件。

其实 copyright 文件不是必须的，因为少了还能构建，必须的文件： `control、changelog、rules、compat`

debian 文件夹目录结构：

自动生成的文件中，必须保证 control、changelog、rules、compat 文件存在，不然报错。其余文件斟酌要不要删了

Note:其中生成命令中如果不使用-p 指定软件包名和-u 指定软件版本，将自动用文件夹格式生成软件名和版本，格式:`软件名-版本`，这里不设置，默认为软件名:monitor、版本:desktop，版本号不是数字；打 deb 包时，changelog 的讲报错，因为抬头格式不正确，需要手动修改为正确的版本号，或者你的文件夹名起名规范。

这里不出情况，只需要关注 control 下即可。

4.2、debian/control 配置文件

`debmake -n`命令生成的 control：

```
Source: monitor

Section: unknown

Priority: extra

Maintainer: <hui@localhost>

Build-Depends: debhelper (>=9), qt4-qmake

Standards-Version: 3.9.8

Homepage: <insert the upstream URL, if relevant>



Package: monitor

Architecture: any

Multi-Arch: foreign

Depends: ${misc:Depends}, ${shlibs:Depends}

Description: auto-generated package by debmake

 This Debian binary package was auto-generated by the

 debmake(1) command provided by the debmake package.
```

1
2
3
4
5
6

```
7
8
9
10
11
12
13
14
15
```

参数解释：

参数	解释
Source	This field identifies the source package name，不知道用来干嘛的，但是改了这需要同时修改
Section	This field specifies an application area into which the package has been classified，暂时不知可能不是–session-sid，暂时只知道一个选项是 base
Priority	This field represents how important it is that the user have the package installed，这个属性standard、optional、extra(已弃用)五个选项，其中官方推荐是 optional，但是根据实际情况
Maintainer	维护者信息或者说是作者信息
Build-Depends	构建的依赖，列出来的依赖，需要系统安装了才能制作 deb 包成功
Standards-Version	The most recent version of the standards (the policy manual and associated texts) with whi complies，不清楚怎么给出来的。
Homepage	主页地址，描述性信息
Package	The name of the binary package，最后生成的软件名，安装后 `dpkg -l` 显示的名字，卸载也
Architecture	软件的架构，可以用 dpkg-architecture -L 中列出的字符串，最多人用的是 `all` 或者 `any`
Multi-Arch	具体参看 `man debmake` 的-b 参数，自动设置。或者-b 设置特定类型的应用自动设置，如果不

参数	解释
	c++的，可能需要设置一下 `debmake -b` 设置具体的程序类型，令 Architecture 和 Multi-Arch
Depends	默认即可，要知道详细可到 4.1 章节处了解
Description	软件描述，dpkg 安装时显示。

权威解释请看：https://www.debian.org/doc/manuals/maint-guide/dreq.zh-cn.html#control 的 4.1 章节。
和 https://www.debian.org/doc/debian-policy/ch-controlfields.html#source-package-control-files-debian-control

4.3、desktop 配置文件

.desktop 文件是桌面版本的软件的快捷方式描述文件。
只是普通文本文件，找个模板修改即可，之后看看有没空，找找生成工具什么的。

```
[Desktop Entry]

Categories=Application;Utility;

Encoding=UTF-8

Exec=/usr/bin/rekols-monitor

Icon=apps.com.onlinego

Name=Rekols Monitor

Name[zh_CN]=系统监视悬浮窗

Name[zh_TW]=系统监视悬浮窗

StartupNotify=false

Terminal=false
```

```
Type=Application

X-MultipleArgs=false
```

关键字	意义
[Desktop Entry]	文件声明行
Categories	应用的类型（内容相关）
Encoding	编码
Exec	执行的命令
Icon	图标路径
Name	应用名称
Name[xx]	不同语言的应用名称，这里配置了两个语言
StartupNotify	是否能使用提示反馈，默认值为 false
Terminal	是否使用终端

关键字	意义
Type	启动器类型
X-MultipleArgs	是否有多个参数

参数参考：

- Linux 创建启动器（.Desktop 文件）
- Desktop Entry 文件
- https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html

Ubuntu 中 deb 包详解及打包教程

一、deb 包详解

1->deb 包的文件结构

deb 软件包里面的结构：它具有 DEBIAN 和软件具体安装目录（如 etc, usr, opt, tmp 等）。

deb 包本身有三部分组成	
组成	**详细**
数据包	包含实际安装的程序数据，文件名为“data.tar.XXX”
安装信息及控制包	包含 deb 的安装说明，标识，脚本等，文件名为“control.tar.gz”
二进制数据	包含文件头等信息，需要特殊软件才能查看

|----DEBIAN
|-------control
|-------postinst(postinstallation)
|-------postrm(postremove)

|-------preinst(preinstallation)

|-------*prerm(preremove)*

|-------*copyright(版权)*

|-------*changlog(修订记录)*

|-------*conffiles*

|----*etc*
|----*usr*
|----*opt*
|----*tmp*

|----*boot*

|-----*initrd-vstools.img*

2->control 文件

control:这个文件主要描述软件包的名称（Package），版本（Version），Installed-Size（大小），Maintainer（打包人和联系方式）以及描述（Description）等，是 deb 包必须具备的描述性文件，以便于软件的安装管理和索引。

control 文件

段	用途	例子/其他
kage	程序名称	中间不能有空格
sion	软件版本	
cription	程序说明	
tion	软件类别	utils, net, mail, text, x11
ority	软件对于系统的重要程度	required, standard, optional, extra 等;
ential	是否是系统最基本的软件包	yes/no，若为 yes,则不允许卸载（除非强制性卸载）
hitecture	软件所支持的平台架构	i386, amd64, m68k, sparc, alpha, powerpc 等
rce	软件包的源代码名称	

ends	软件所依赖的其他软件包和库文件	若依赖多个软件包和库文件，采用逗号隔开
-Depends	软件安装前必须安装、 配置依赖性的软件包和库文件	常用于必须的预运行脚本需求
ommends	推荐安装的其他软件包和库文件	
gests	建议安装的其他软件包和库文件	

备注：
- inst 是 install（安装）的缩写
- pre 是表示 XX 之前的前缀
- post 是表示 XX 之后的前缀
- rm 是 remove（移除）的缩写

3 >preinst 文件

在 Deb 包文件解包之前（即软件安装前），将会运行该脚本。可以停止作用于待升级软件包的服务，直到软件包安装或升级完成。

4 >postinst 文件

负责完成安装包时的配置工作。如新安装或升级的软件重启服务。软件安装完后，执行该 Shell 脚本，一般用来配置软件执行环境，必须以“#!/bin/sh”为首行。

```
1. #!/bin/sh
2. echo "my deb" > /root/mydeb.log
```

```
1. #!/bin/sh
2. if [ "$1" = "configure" ]; then
3. /Applications/MobileLog.app/MobileLog -install
4. /bin/launchctl load -wF
   /System/Library/LaunchDaemons/com.iXtension.MobileLogDaemon.plist
5. fi
```

5 >prerm 文件

该脚本负责停止与软件包相关联的 daemon 服务。它在删除软件包关联文件之前执行。

```
1. #!/bin/sh
2. if [[ $1 == remove ]]; then
3. /Applications/MobileLog.app/MobileLog -uninstall
4. /bin/launchctl unload -wF
   /System/Library/LaunchDaemons/com.iXtension.MobileLogDaemon.plist
5. fi
```

6 >postrm 文件

负责修改软件包链接或文件关联，或删除由它创建的文件。软件卸载后，执行该 Shell 脚本，一般作为清理收尾工作，必须以“#!/bin/sh”为首行

```
1. #!/bin/sh
2. rm -rf /root/mydeb.log
```

二、dpkg 详解

1 > 打包 dpkg -b

 # dpkg -b . mydeb-1.deb
第一个参数为将要打包的目录名（.表示当前目录），第二个参数为生成包的名称<.deb file name>

2 > 安装（解包并配置） dpkg -i|--install <.deb file name>

 # dpkg -i mydeb-1.deb
//强制安装

 # dpkg --force-depends -i mydeb-1.deb
解包：
 # dpkg --unpack mydeb-1.deb
该命令仅对 “mydeb-1.deb”进行解包操作，不会执行包配置工作。

3 > 卸载 dpkg -r|--remove <package> ，删除包，但保留配置文件

 # dpkg -r my-deb
 # dpkg -P|--purge my-deb
该命令删除包，且删除配置文件。

4 > 查看 deb 包是否安装/deb 包的信息 dpkg -s|--status <package>

 # dpkg -s my-deb
5 > 查看 deb 包文件内容

 # dpkg -c mydeb-1.deb
6 > 查看当前目录某个 deb 包的信息

 # dpkg --info mydeb-1.deb
7 > 解压 deb 中所要安装的文件

 # dpkg -x mydeb-1.deb mydeb-1
第一个参数为所要解压的 deb 包，第二个参数为将 deb 包解压到指定的目录

8 >解压 deb 包中 DEBIAN 目录下的文件（至少包含 control 文件）

 # dpkg -e mydeb-1.deb mydeb-1/DEBIAN
9 > 列出与该包关联的文件 dpkg -L|--listfiles <package>

 # dpkg -L my-deb

10 > 配置软件包 dpkg --configure <package>

 # dpkg --configure my-deb

三、制作 deb 流程

准备好可执行的二进制文件，这个二进制文件要可执行，提前要考虑兼容性，如果程序有目录要完整的一个程序目录。

1->新建软件文件夹
我们测试名称为 JFeng-deb

新建一个名为 DEBIAN 文件夹
此文件夹内存放控制信息

在 DEBIAN 里新建一个文本文档, 名为 control, 编码为 utf-8, 内容如下所示：

```
Package: JFeng
Version: 1.1.0
Architecture: amd64
Section: utils
Priority: optional
Maintainer: MC
Homepage: http://montecarlo.org.cn
Description: Gale debug
```

然后我们创建对应的二进制包安装完成后的路径信息放置在 DEBIAN 的同级目录下，也就是把当前的目录当成根(“/”)目录,制作完成后安装时，当前目录下除了 DEBIAN 目录的其他目录都会被默认安装到系统的“/”目录下。

下面是一个程序目录的例子。

```
|—JFeng-deb
    |———usr
        |———bin
            |—可执行文件（安装后，就在你的/usr/bin 生成相应的可执行文件）
        |———share

            |———icons

                |—deb.png(图标文件生成到/usr/share/icons/)

            |———applications
```

```
                    |——deb.desktop（桌面文件生成到/usr/share/applications/）
13.
14.     |———DEBIAN(大写、用来制作打包文件)
15.             |———control(描述 deb 包的信息必须的文件)
```

完整实验例子目录结构：

```
JFeng-deb
├── DEBIAN
│   └── control
├── opt
│   └── JFeng
│       ├── heart
│       └── heart.desktop
└── usr
    ├── bin
    │   └── heart -> /home/wxyz/桌面/JFeng-deb/opt/MyDeb/heart
    └── share
        ├── applications
        │   └── heart.desktop
        └── icons
            └── heart_98.png

8 directories, 6 files
```

3->打包

sudo dpkg -b JFeng-deb/ JFeng-linux-amd64.deb

一、打包标准

1、官方包：

请从软件官网下载，经自行测试可用后投递。

2、自行打包：

请将程序打包在/opt/durapps/目录，如无特殊情况，本站所有 deb 包都应当使用 debreate 应用打包来防止出现不标准包。

打包教程请参考：

https://bbs.deepin.org/forum.php?mod=viewthread&tid=195472

deb 包经检验位置合格后投递。

3、Appimage 套娃包：

建议使用星火商店专用转换器，下载地址：spk://store/tools/a2d（该下载地址需要先安装星火商店）

请确认程序在/opt/durapps/目录，经检验位置合格后投递。

4、可执行文件打包：
经检验位置合格，应用依赖完整，测试可用后投递

Debreate 获取方法：
deepin20 可以在商店获取 debreate；
deepin15.11 或其他发行版请在 https://repositorio.deepines.com/pub/deepines/pool/main/d/debreate/debreate_0.7.13_all.deb 获取。

二、投递流程
1、准备好软件图标、软件截图（不超过 5 张）、软件描述和软件包本体。

2、打开网址：https://upload.spark-app.store/index

3、点击左上角上传，并按要求输入软件包信息。

4、等待审核，审核通过后，您提交的软件会在对应分类下出现，如果审核未通过，我们会通知您未通过的原因。

Pycharm 打包教程（也可用于其他软件的打包）

本帖最后由 anysets 于 2020-6-3 22:27 编辑

今天想安装 pycharm，想到 https://bbs.deepin.org/user/223313 大神的网站里好像有，于是就兴冲冲地跑过去看
结果……不维护了

没事，既然没有现成的，那咱不如自己动手来打包一个 pycharm 吧

首先感谢@shenmo 在 B 站上传的教程，这里做一个文字版的讲解（b 站：https://www.bilibili.com/video/BV1xf4y1U7ZU?t=53）
（教程不仅仅适用于 pycharm）

一、准备
去 pycharm 官网下载官方的包（我下的是 community 版的，按照自己需求下载）

得到了 文件后，解压

然后去应用商店搜索 debreate 打包工具，安装

好了，准备工作就完成了

二、开始制作

我们打开 debreate，开始制作

1、information

这是个欢迎界面，我们直接下一步（control 旁边蓝色的小箭头）

2、control

在这里，我们需要填写 deb 包的一些信息
我只介绍我们需要填的：

Package（包名）

这里填写这个 deb 包的包名（软件名），这里我就填的是 Pycharm

Version（版本）

这里填写软件的版本，我下的是 2020.1.1 版，所以我填的是 2020.1.1

Maintainer（维护者）

填写自己名字就行了

Email（邮箱）

可以填自己的邮箱，但是不能不填

Architecture（包架构）

选择 amd64 即可（当然也可以打包成别的架构）

Short Description（短描述）

随便填，安装时可以看到你填的内容。

其他的部分可以自己 DIY

示例：

3、Dependencies and Conflicts

这里可以添加依赖，这里 pycharm 官方包的依赖没有问题，我就不再添加了

4、Files

我们需要在这里把官方包里的文件弄进来

先填写安装目录（一般在 opt 目录下），这里我就填/opt/pycharm-community

然后我们把官方包里的内容拖进来

5、Scripts
我们可以写一个卸载的脚本，选择 Post-Remove,把 Make this script 钩上
然后写入：sudo rm -rf +安装目录
比如这里：sudo rm -rf /opt/pycharm-community

6、Changelog 和 Copyright
不需要改动，直接下一步

7、Menu Launcher
这里可以修改快捷方式的配置

Name：快捷方式名
比如我的快捷方式就叫 Pycharm

Executable：可执行文件位置
可以在官方包里找到，pycharm 的是 bin 文件夹下的 pycharm.sh(双击可直接运行 pycharm)
注意要填用户安装后的路径，比如我第 5 部填的安装目录是/opt/pycharm-community，那么我这里就应该填写/opt/pycharm-community/bin/pycharm.sh

icon：图标位置
也可以在官方包里找到，pycharm 的是 bin 文件夹下的 pycharm.png
和上面一样，需要填安装后的路径，我这里就填/opt/pycharm-community/bin/pycharm.png

Category：软件类型
这个可以随便选，安装完程序会出现在启动器的对应分类下

8、最后，点击绿色的按钮，开始制作 deb 吧！

三、安装 deb & 享受自己打包的成果
制作完成后，你就可以双击 deb 文件开始安装使用啦！

---

rclinux 资讯

1.redclinux 安装脚本终于完美的做出了下周发布

2.rclinuxbeta3 内部测试即将出现

3.待验证后公布 rclinux 中文名进展将采用《山海经》命名