

第一章 使用官方提供的源码包进行构建

1.1 搭建开发环境

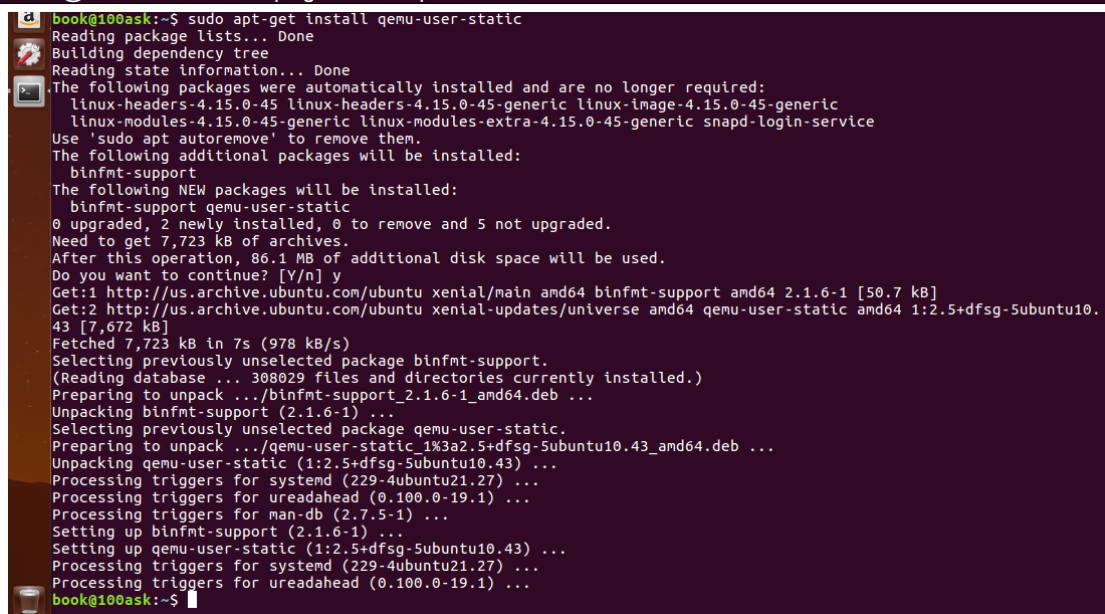
开发涉及的环境/工具:

- Windows 下虚拟化工具 VMware workstation pro。
- vmware 下运行的 ubuntu 16.04 虚拟机系统。
- arm 架构的 ubuntu 根文件系统源码包。
- ch-mount.sh 挂载文件系统脚本。

1.1.1 安装qemu虚拟化工具

Ubuntu 终端下需要安装 qemu 虚拟化工具，在终端下执行如下命令。

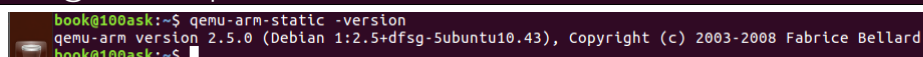
```
book@100ask:~$ sudo apt-get install qemu-user-static
```



```
book@100ask:~$ sudo apt-get install qemu-user-static
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-4.15.0-45 linux-headers-4.15.0-45-generic linux-image-4.15.0-45-generic
  linux-modules-4.15.0-45-generic linux-modules-extra-4.15.0-45-generic snapd-login-service
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  binfmt-support
The following NEW packages will be installed:
  binfmt-support qemu-user-static
0 upgraded, 2 newly installed, 0 to remove and 5 not upgraded.
Need to get 7,723 kB of archives.
After this operation, 86.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 binfmt-support amd64 2.1.6-1 [50.7 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 qemu-user-static amd64 1:2.5+dfsg-5ubuntu10.43 [7,672 kB]
Fetched 7,723 kB in 7s (978 kB/s)
Selecting previously unselected package binfmt-support.
(Reading database ... 308029 files and directories currently installed.)
Preparing to unpack .../binfmt-support_2.1.6-1_amd64.deb ...
Unpacking binfmt-support (2.1.6-1) ...
Selecting previously unselected package qemu-user-static.
Preparing to unpack .../qemu-user-static_1%3a2.5+dfsg-5ubuntu10.43_amd64.deb ...
Unpacking qemu-user-static (1:2.5+dfsg-5ubuntu10.43) ...
Processing triggers for systemd (229-4ubuntu21.27) ...
Processing triggers for ureadahead (0.100.0-19.1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up binfmt-support (2.1.6-1) ...
Setting up qemu-user-static (1:2.5+dfsg-5ubuntu10.43) ...
Processing triggers for systemd (229-4ubuntu21.27) ...
Processing triggers for ureadahead (0.100.0-19.1) ...
book@100ask:~$
```

安装完成后，在文件系统下执行如下命令测试是否安装成功。

```
book@100ask:~$ qemu-arm-static -version
```







```
book@100ask:~$ qemu-arm-static -version
qemu-arm version 2.5.0 (Debian 1:2.5+dfsg-5ubuntu10.43), Copyright (c) 2003-2008 Fabrice Bellard
book@100ask:~$
```

1.1.2 获取arm架构ubuntu根文件系统

使用浏览器访问 <http://cdimage.ubuntu.com/ubuntu-base/> 此地址，即可看到 ubuntu 基本系统所有的版本镜像文件，这里我们选择 **releases** 发布版。

Index of /ubuntu-base






























Name	Last modified	Size
 Parent Directory		-
 bionic/	2018-05-03 05:01	-
 daily/	2020-02-27 04:43	-
 releases/	2020-02-12 13:28	-

每日更新版
正式发布版

Apache/2.4.18 (Ubuntu) Server at cdimage.ubuntu.com Port 80

我们点击 **release** 发布版本的连接后进入如下图所示界面，这里列出了 **ubuntu base** 各个版本的下载地址，如下图根据红框所示，可以选择 **16.04** 或者 **18.04** 这里我们选择 **16.04**。

Index of /ubuntu-base/releases

Name	Last modified	Size
 Parent Directory		-
 12.04.4/	2012-04-26 09:21	-
 12.04.5/	2012-04-26 09:21	-
 12.04/	2012-04-26 09:21	-
 14.04.1/	2014-04-17 15:33	-
 14.04.2/	2014-04-17 15:33	-
 14.04.3/	2014-04-17 15:33	-
 14.04.4/	2014-04-17 15:33	-
 14.04.5/	2014-04-17 15:33	-
 14.04.6/	2014-04-17 15:33	-
 14.04/	2014-04-17 15:33	-
 16.04.1/	2016-04-21 11:22	-
 16.04.2/	2016-04-21 11:22	-
 16.04.3/	2016-04-21 11:22	-
 16.04.4/	2016-04-21 11:22	-
 16.04.5/	2016-04-21 11:22	-
 16.04.6/	2016-04-21 11:22	-
 16.04/	2016-04-21 11:22	-
 18.04.1/	2018-04-26 20:56	-
 18.04.2/	2018-04-26 20:56	-
 18.04.3/	2018-04-26 20:56	-
 18.04.4/	2018-04-26 20:56	-
 18.04/	2018-04-26 20:56	-
 19.10/	2019-10-17 14:16	-
 bionic/	2018-04-26 20:56	-
 eoan/	2019-10-17 14:16	-
 precise/	2012-04-26 09:21	-
 trusty/	2014-04-17 15:33	-
 xenial/	2016-04-21 11:22	-

Apache/2.4.18 (Ubuntu) Server at cdimage.ubuntu.com Port 80






















点击 **16.04** 后在弹出新的页面内继续点击 **releases** 。

Index of /ubuntu-base/releases/16.04

Name	Last modified	Size
 Parent Directory	-	-
 release/	2019-02-28 14:37	-

Apache/2.4.18 (Ubuntu) Server at cdimage.ubuntu.com Port 80

之后页面就跳入 ubuntu base 16.04 lts 文件系统的下载页面了，这里列出了各种架构的文件系统源码包，不同的历史版本，我们页面往下滑，找到最新的架构为 **armhf** 的 **ubuntu** 系统源码包，如下图红框所示，下载 **ubuntu-base-16.04.6-base-armhf.tar.gz** 到电脑磁盘上，让后上传此文件到 **VMware Ubuntu-16.04** 系统内。

	ubuntu-base-16.04.5-base-i386.tar.gz.zsync	2018-08-02 10:08	143K
	ubuntu-base-16.04.5-base-powerpc.tar.gz	2018-07-31 00:30	42M
	ubuntu-base-16.04.5-base-powerpc.tar.gz.zsync	2018-08-02 10:08	146K
	ubuntu-base-16.04.5-base-ppc64el.tar.gz	2018-07-31 00:29	43M
	ubuntu-base-16.04.5-base-ppc64el.tar.gz.zsync	2018-08-02 10:08	152K
	ubuntu-base-16.04.5-base-s390x.tar.gz	2018-07-31 00:27	40M
	ubuntu-base-16.04.5-base-s390x.tar.gz.zsync	2018-08-02 10:08	139K
	ubuntu-base-16.04.6-base-amd64.tar.gz	2019-02-27 16:24	41M
	ubuntu-base-16.04.6-base-amd64.tar.gz.zsync	2019-02-28 14:37	143K
	ubuntu-base-16.04.6-base-arm64.tar.gz	2019-02-27 16:28	37M
	ubuntu-base-16.04.6-base-arm64.tar.gz.zsync	2019-02-28 14:37	130K
	ubuntu-base-16.04.6-base-armhf.tar.gz	2019-02-27 16:28	36M
	ubuntu-base-16.04.6-base-armhf.tar.gz.zsync	2019-02-28 14:37	126K
	ubuntu-base-16.04.6-base-i386.tar.gz	2019-02-27 16:25	41M
	ubuntu-base-16.04.6-base-i386.tar.gz.zsync	2019-02-28 14:37	143K
	ubuntu-base-16.04.6-base-powerpc.tar.gz	2019-02-27 16:28	42M
	ubuntu-base-16.04.6-base-powerpc.tar.gz.zsync	2019-02-28 14:37	146K
	ubuntu-base-16.04.6-base-ppc64el.tar.gz	2019-02-27 16:27	43M
	ubuntu-base-16.04.6-base-ppc64el.tar.gz.zsync	2019-02-28 14:37	152K
	ubuntu-base-16.04.6-base-s390x.tar.gz	2019-02-27 16:25	40M
	ubuntu-base-16.04.6-base-s390x.tar.gz.zsync	2019-02-28 14:37	140K

Apache/2.4.18 (Ubuntu) Server at cdimage.ubuntu.com Port 80

1.1.3 Vmware ubuntu-16.04挂载arm架构ubuntu文件系统

1.1.3.1 解压缩 arm 架构 ubuntu-16.04 文件系统到 ubuntu 下

在 VMware Ubuntu-16.04 虚拟机家目录下创建一个 **ubuntu-rootfs** 目录，用于解压缩保存文件使用。

```
book@100ask:~$ mkdir ubuntu-rootfs
```

```
book@100ask:~$ sudo tar -xvf ubuntu-base-16.04.6-base-armhf.tar.gz -C ubuntu-rootfs/
```

查看当前路径下的文件结构已经 OK

```
book@100ask:~$ ls ubuntu-rootfs/  
bin boot dev etc home lib media mnt opt proc root run sbin srv sys usr var  
book@100ask:~$
```

1.1.3.2 拷贝 qemu 模拟工具到 arm 架构 ubuntu-16.04 文件系统下

```
book@100ask:~$ sudo cp /usr/bin/qemu-arm-static ubuntu-rootfs/usr/bin/
```

注意: `qemu-arm-static` 指的是 `armhf` 架构的虚拟化工具, 果是其他架构的则拷贝其他文件
`ubuntu-rootfs/usr/bin/` 指的是你当前解压 `arm` 架构的 `ubuntu16.04` 文件系统后的所在目录下的 `usr/bin`。

拷贝主机 `DNS` 配置文件到 `arm` 架构 `Ubuntu` 文件系统内(必须拷贝, 否则可能会导致下面操作无法进行)。

```
book@100ask:~$ sudo cp /etc/resolv.conf ubuntu-rootfs/etc/resolv.conf
```

1.1.3.3 chroot 到模拟 arm 文件系统下

我们需要使用 `chroot` 改变根目录来挂载 `arm` 架构的 `ubuntu-16.04` 根文件系统, 并配置或安装一些必要资源, 首先创建 `ch-mount.sh` 脚本。

```
book@100ask:~$ vi ch-mount.sh
```

将以下内容复制到 `ch-mount.sh` 中。

```
#!/bin/bash  
#  
function mnt() {  
    echo "MOUNTING"  
    sudo mount -t proc /proc ${2}proc  
    sudo mount -t sysfs /sys ${2}sys  
    sudo mount -o bind /dev ${2}dev  
    sudo mount -o bind /dev/pts ${2}dev/pts  
    sudo chroot ${2}  
}  
function umnt() {  
    echo "UNMOUNTING"  
    sudo umount ${2}proc  
    sudo umount ${2}sys  
    sudo umount ${2}dev/pts  
    sudo umount ${2}dev  
}  
if [ "$1" == "-m" ] && [ -n "$2" ] ;  
then  
    mnt $1 $2  
elif [ "$1" == "-u" ] && [ -n "$2" ] ;
```

```

then
    umnt $1 $2
else
    echo ""
    echo "Either 1'st, 2'nd or both parameters were missing"
    echo ""
    echo "1'st parameter can be one of these: -m(mount) OR -u(umount)"
    echo "2'nd parameter is the full path of rootfs directory(with trailing '/')"
    echo ""
    echo "For example: ch-mount -m /media/sdcard/"
    echo ""
    echo 1st parameter : ${1}
    echo 2nd parameter : ${2}
fi

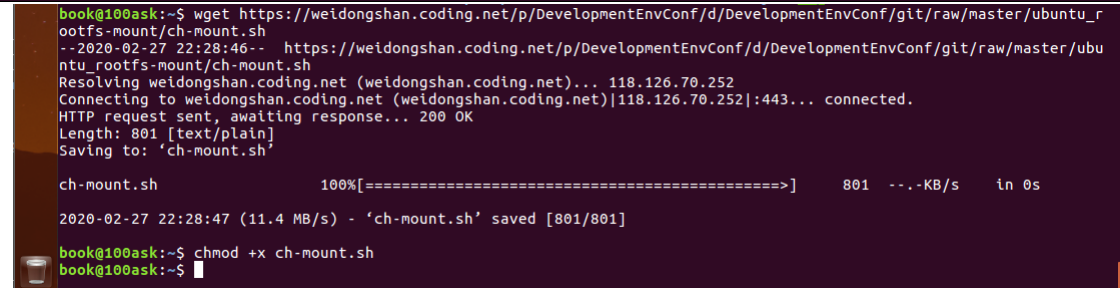
```

考虑到 **shell** 脚本对空格很灵敏，所以将我使用的文件放在如下地址处。

```

book@100ask:~$ wget \
https://weidongshan.coding.net/p/DevelopmentEnvConf/d/DevelopmentEnvConf/git/raw/master/ubuntu_
rootfs-mount/ch-mount.sh
book@100ask:~$ chmod +x ch-mount.sh

```



接下来我们使用 **ch-mount.sh** 脚本挂载 arm 架构 ubuntu-16.04 文件系统，挂载命令如下所示，挂载成功后会提示 **MOUNTING**。

```
book@100ask:~$ sudo ./ch-mount.sh -m ubuntu-rootfs/
```

此时我们可以在此执行 **uname -a** 来查看系统内核的详细信息，你会发现现在是 **armv7** 架构。

```

root@100ask:/# uname -a
Linux 100ask.org 4.15.0-74-generic #83~16.04.1-Ubuntu SMP Wed Dec 18 04:56:23 UTC 2019 armv7l armv7l armv7l GNU/Linux
root@100ask:/#

```

1.2 配置 arm 架构的 ubuntu 系统

1.2.1 安装基础软件包

Chroot 进入模拟的 arm 架构 Ubuntu 系统后需要先安装如下必须的安装包，安装包安装过程会根据你的网络下载速率可能会需要一段时间。

安装基础软件包之前需要先执行 **apt-get update** 命令来更新软件源，用以获取软件包的地址等。

```
root@100ask:~# apt-get update
Hit:1 http://ports.ubuntu.com/ubuntu-ports xenial InRelease
Get:2 http://ports.ubuntu.com/ubuntu-ports xenial-updates InRelease [109 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports xenial-backports InRelease [107 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports xenial-security InRelease [109 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports xenial/universe armhf Packages [7301 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports xenial/universe armhf Packages [7301 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports xenial/universe Translation-en [4354 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports xenial/multiverse armhf Packages [123 kB]
Get:9 http://ports.ubuntu.com/ubuntu-ports xenial/multiverse Translation-en [106 kB]
Get:10 http://ports.ubuntu.com/ubuntu-ports xenial-updates/main armhf Packages [813 kB]
Get:11 http://ports.ubuntu.com/ubuntu-ports xenial-updates/main Translation-en [424 kB]
Get:12 http://ports.ubuntu.com/ubuntu-ports xenial-updates/restricted armhf Packages [4976 B]
Get:13 http://ports.ubuntu.com/ubuntu-ports xenial-updates/restricted Translation-en [2272 B]
Get:14 http://ports.ubuntu.com/ubuntu-ports xenial-updates/universe armhf Packages [708 kB]
Get:15 http://ports.ubuntu.com/ubuntu-ports xenial-updates/universe Translation-en [330 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports xenial-updates/multiverse armhf Packages [12.2 kB]
Get:17 http://ports.ubuntu.com/ubuntu-ports xenial-updates/multiverse Translation-en [8468 B]
Get:18 http://ports.ubuntu.com/ubuntu-ports xenial-backports/main armhf Packages [7284 B]
Get:19 http://ports.ubuntu.com/ubuntu-ports xenial-backports/main Translation-en [4456 B]
Get:20 http://ports.ubuntu.com/ubuntu-ports xenial-backports/universe armhf Packages [7740 B]
Get:21 http://ports.ubuntu.com/ubuntu-ports xenial-backports/universe Translation-en [4328 B]
Get:22 http://ports.ubuntu.com/ubuntu-ports xenial-security/main armhf Packages [563 kB]
Get:23 http://ports.ubuntu.com/ubuntu-ports xenial-security/main Translation-en [316 kB]
Get:24 http://ports.ubuntu.com/ubuntu-ports xenial-security/universe armhf Packages [427 kB]
Get:25 http://ports.ubuntu.com/ubuntu-ports xenial-security/universe Translation-en [199 kB]
Get:26 http://ports.ubuntu.com/ubuntu-ports xenial-security/multiverse armhf Packages [3012 B]
Get:27 http://ports.ubuntu.com/ubuntu-ports xenial-security/multiverse Translation-en [2708 B]
Fetched 13.4 MB in 18min 56s (11.8 kB/s)
Reading package lists... Done
root@100ask:~#
```

软件源更新完成后，可以安装必要软件包，安装速度根据个人网速绝定。

```
apt-get install \
language-pack-en-base sudo ssh net-tools network-manager iputils-ping rsyslog \
bash-completion language-pack-zh-hans vim resolvconf kmod usbutils alsa-base
```

```
root@100ask:~# apt-get install \
> language-pack-en-base sudo ssh net-tools network-manager iputils-ping rsyslog bash-completion \
> language-pack-zh-hans vim resolvconf kmod usbutils alsa-base
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
acl adwaita-icon-theme alsa-utils at-spi2-core binutils ca-certificates colord colord-data crda cron dbus
dbus-x11 dconf-gsettings-backend dconf-service dh-python dns-root-data dnsmasq-base file fontconfig
fontconfig-config fonts-dejavu-core gcr glib-networking glib-networking-common glib-networking-services
gnome-keyring gsettings-desktop-schemas hicolor-icon-theme humanity-icon-theme ifupdown indicator-application
iproute2 iptables iputils-arping isc-dhcp-client isc-dhcp-common iso-codes iw krb5-locales language-pack-en
language-pack-zh-hans-base libappindicator3-1 libasound2 libassuan0 libatk-bridge2.0-0
libatk1.0-0 libatk1.0-data libatm1 libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3
libbluetooth3 libboost-filesystem1.58.0 libboost-system1.58.0 libbsd0 libcairo-gobject2 libcairo2 libcap-ng0
libcapnp-0.5.3 libcolord2 libcolorhug2 libcroco3 libcups2 libdatrie1 libdbus-1-3 libdbus-glib-1-2
libdbusmenu-glib4 libdbusmenu-gtk3-4 libdconf1 libdns-export162 libdrm-amdgpu1 libdrm-common libdrm-etnavi1
libdrm-freedreno1 libdrm-nouveau2 libdrm-radeon1 libdrm2 libedit2 libegl1-mesa libelf1 libepoxy0 libestr0
libexif12 libexpat1 libffi6 libfftw3-double3 libfontconfig1 libfreetype6 libfribidi0 libgbm1 libgck-1-0
libgcr-3-common libgcr-base-3-1 libgcr-ui-3-1 libgd3 libgdk-pixbuf2.0-0 libgdk-pixbuf2.0-common libgl1-mesa-dri
libglapi-mesa libglb2-0-0 libglb2-0-data libgmp10 libgnutls-openssl17 libgnutls30 libgomp1 libgphoto2-6
libgphoto2-l10n libgphoto2-port12 libgpm2 libgraphite2-3 libgssapi-krb5-2 libgtk-3-0 libgtk-3-bin
libgtk-3-common libgudev-1.0-0 libgusb2 libharfbuzz0b libhogweed4 libicu55 libidn11 libieee1284-3
libindicator3-7 libisc-export160 libjbig0 libjpeg-turbo8 libjpeg8 libjson-c2 libjson-glib-1.0-0
libjson-glib-1.0-common libk5crypto3 libkeyutils1 libkrb5-3 libkrb5support0 liblcms2-2 libllvmm6.0 libltdl7
libmagic1 libmbim-glib4 libmbim-proxy libmirclient9 libmircommon7 libmircore1 libmirprotobuf3 libmm-glib0
libnm10 libnbd2 libndp0 libnetfilter-conntrack3 libnettle6 libnewt0.52 libnfnfnetlink0 libnl-3-200
libnl-genl-3-200 libnm0 libnma-common libnma0 libnotify4 libp11-kit-gnome-keyring libp11-kit0
libpam-gnome-keyring libpam-systemd libpango-1.0-0 libpangocairo-1.0-0 libpangotf2-1.0-0 libpcap0.8 libpcsclite1
libpipeline1 libpixman-1-0 libpng12-0 libpolkit-agent-1-0 libpolkit-backend-1-0 libpolkit-gobject-1-0 libpopt0
libprotobuf-lite9v5 libproxy1v5 libpython3-stdlib libpython3.5 libpython3.5-minimal libpython3.5-stdlib
libqmi-glib5 libqmi-proxy librest-0.7-0 librsync2-2 librsync2-common libsamplerate0 libsane libsane-common
libsecret-1-0 libsecret-common libensors4 libslang2 libsoup-gnome2.4-1 libssl1.0.2 libssl1.0.0
libtasn1-6 libthai-data libthai0 libtiff5 libtsc-dxt0 libusb-1.0-0 libvpx3 libwayland-client0
libwayland-cursor0 libwayland-egl1-mesa libwayland-server0 libwrap0 libx11-6 libx11-data libx11-xcb1 libxau6
libxcb-dri2-0 libxcb-dri3-0 libxcb-present0 libxcb-render0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1
libxcomposite1 libxcursor1 libxdamage1 libxdmcp6 libxext6 libxfixes3 libxi6 libxinerama1 libxkbcommon0 libxml2
libxmuu1 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxtables11 libxtst6 linux-sound-base locales logrotate
mime-support mobile-broadband-provider-info modemmanager ncurses-term network-manager-gnome network-manager-pptp
notification-daemon openssh-client openssh-server openssh-sftp-server openssl p11-kit p11-kit-modules
pintentry-gnome3 policykit-1 policykit-1-gnome ppp ptp-linux python3 python3-chardet python3-minimal
python3-pkg-resources python3-requests python3-six python3-urllib3 python3.5 python3.5-minimal sgml-base
shared-mime-info ssh-import-id tcpd ubuntu-mono ucf udev usb-modeswitch usb-modeswitch-data vim-common
vim-runtime wget whiptail wireless-regdb wpasupplicant x11-common xauth xdg-user-dirs xkb-data xml-core
Suggested packages:
```

1.2.2 用户名密码等相关设置

1.2.2.1 添加用户、设定合适的组并设置密码

添加 book 用户并加入 admin sudo 用户组，设置密码为 123456

```
root@100ask:~# useradd -s '/bin/bash' -m -G adm,sudo book
root@100ask:~# echo "Set password for book:"
root@100ask:~# passwd book
```

初始化 root 用户密码，这里设置为 123456


```
root@100ask:/# passwd root
```

1.2.3 其它配置

1.2.3.1 设置主机名称和 hosts

在模拟的 arm 架构 ubuntu 根文件系统中执行如下两条命令即可设置主机名称。

```
root@100ask:/# echo 100ask.org > /etc/hostname
root@100ask:/# echo 100ask.org > /etc/hosts
```

1.2.3.2 配置登陆的启动串口脚本

因为暂时未安装桌面，所以这里的配置要具体和内核中登录的串口的设备对应起来，不然会导致无法通过串口登录的问题。

在 /etc/init/ 下添加或修改 ttymxc0.conf

```
root@100ask:/# cat > /etc/init/ttymxc0.conf << EOT
start on stopped rc RUNLEVEL=[2345]
stop on runlevel [!2345]
respawn
exec /sbin/getty -L 115200 ttymxc0
EOT
```

```
root@100ask:/# cat > /etc/init/ttymxc0.conf << EOT
> start on stopped rc RUNLEVEL=[2345]
> stop on runlevel [!2345]
> respawn
> exec /sbin/getty -L 115200 ttymxc0
> EOT
root@100ask:/#
```

1.2.3.3 配置网卡接口

```
root@100ask:/# cat >> /etc/network/interfaces << EOT
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
EOT
```

```
root@100ask:/# cat >> /etc/network/interfaces << EOT
> auto lo
> iface lo inet loopback
> auto eth0
> iface eth0 inet dhcp
> EOT
root@100ask:/# cat /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
root@100ask:/#
```

1.2.3.4 退出 arm 模拟文件系统

配置或安装完基本的设置后，就可以退出模拟的 arm 架构文件系统了，操作步骤如下，先在模拟的 arm 架构文件系统中执行 `exit` 退出到 VMwareubuntu-16.04 虚拟机终端界面，让后卸载 `chroot` 挂载。

```
root@100ask:/# exit
book@100ask:~$ sudo ./ch-mount -u ubuntu-rootfs
```

完成这些后，我们需要把内核镜像设备树，以及模块驱动等文件拷贝到 arm 架构的

ubuntu 文件系统相应目录内。

1.2.4 配置系统内核模块以及固件

拷贝编译好的内核镜像和设备树文件到 arm 架构 ubuntu 根文件系统的 boot 目录下。

```
book@100ask:~$ sudo cp zImage 100ask_imx6ull-14x14.dtb ubuntu-rootfs/boot/
book@100ask:~$ ls ubuntu-rootfs/boot/
```

```
book@100ask:~$ ls ubuntu-rootfs/boot/
100ask_imx6ull-14x14.dtb  zImage
book@100ask:~$
```

安装内核模块到 arm 架构的 ubuntu 文件系统内,如下命令所示,INSTALL_MOD_PATH 后面的目录地址为 arm 架构 ubuntu 文件系统所在绝对路径(模块安装前需要先编译模块)。

```
book@100ask:~/100ask_imx6ull-sdk/Linux-4.9.88$ sudo make ARCH=arm
INSTALL_MOD_PATH=/home/book/ubuntu-rootfs modules_install
```

安装完成后查看 arm 架构 ubuntu 文件系统的 lib/modules/4.9.88 目录下是否有如下相应文件生成

```
book@100ask:~$ ls ubuntu-rootfs/lib/modules/4.9.88/
build      modules.alias.bin  modules.dep        modules.order      modules.symbols.bin
kernel     modules.builtin    modules.dep.bin    modules.softdep    source
modules.alias  modules.builtin.bin  modules.devname    modules.symbols
```

自此, arm 架构的 ubuntu 文件系统已经基本制作完成,接下来我们需要制作为可烧录的镜像文件。

1.2.5 使用nfs方式启动系统系统

1.2.6 配置mate桌面环境

1.3 制作可烧录的固件

1.3.1 制作ext4文件系统镜像

如下命令所示,需要先生成一个大小为 2GB 的 ubuntu-rootfs.ext4 镜像文件,让后格式化镜像为 ext4 格式,之后通过挂载镜像方式把制作好的镜像文件拷贝到文件系统内。

```
book@100ask:~$ dd if=/dev/zero of=ubuntu-rootfs.ext4 bs=1M count=2048
book@100ask:~$ sudo mkfs.ext4 -F ubuntu-rootfs.ext4
```

```
book@100ask:~$ dd if=/dev/zero of=ubuntu-rootfs.ext4 bs=1M count=2048
2048+0 records in
2048+0 records out
2147483648 bytes (2.1 GB, 2.0 GiB) copied, 2.6847 s, 800 MB/s
book@100ask:~$ sudo mkfs.ext4 -F ubuntu-rootfs.ext4
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 524288 4k blocks and 131072 inodes
Filesystem UUID: 50c492e3-6a5a-4a52-bife-1e6f134a068f
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

book@100ask:~$
```

如下命令所示,创建一个 ubuntu-mount 目录,并挂载 ubuntu-rootfs.ext4 镜像到该目录下,之后拷贝制作好的文件系统内所有文件到此挂载目录,紧接着使用 sync 命令同步数据

缓存保证拷贝完成，完成后可以使用 `sudo umount ubuntu-mount/` 卸载当前挂载的镜像。

```
book@100ask:~$ mkdir ubuntu-mount
book@100ask:~$ sudo mount ubuntu-rootfs.ext4 ubuntu-mount/
book@100ask:~$ sudo cp -rvf ubuntu-rootfs/* ubuntu-mount/
book@100ask:~$ sync
book@100ask:~$ sudo umount ubuntu-mount/
```

1.3.2 使用genimage制作可烧录镜像

我们只制作好 `ext4` 格式的文件系统并不能直接烧录到开发板启动，此时我们需要使用一个 `buildroot` 下的一个镜像生成工具 `genimage` 来制作。

首先把编译好生成的 `genimage` 可执行程序拷贝到家目录下，同时把编译好的 `uboot` 镜像文件 `u-boot-dtb.imx` 文件也拷贝到 `~` 目录下。

```
book@book-virtual-machine:~/100ask_imx6ull-sdk/Buildroot_2019.02/output/host/bin$ cp genimage ~
```

此时需要在家目录下新建一个名为 `ubuntu-genimage.cfg` 的分区配置文件，里面写入如下信息。

```
image ubuntu-16.04-armhf_100ask_imx6ull.img {
    hdimage {
    }

    partition u-boot {
        in-partition-table = "no"
        image = "u-boot-dtb.imx"
        offset = 1024
    }
    partition arduino {
        partition-type = 0xC
        size = 50M
        offset = 10M
    }
    partition rootfs {
        partition-type = 0x83
        image = "ubuntu-rootfs.ext4"
        size = 2050M
    }
}
```

此时目录下有如下 4 个文件，分别是 `ubuntu-genimage.cfg` `ubuntu-rootfs.ext4` `u-boot-dtb.imx` `genimage`

```
book@book-virtual-machine:~$ ls ubuntu-genimage.cfg ubuntu-rootfs.ext4 u-boot-dtb.imx genimage -la
-rwxr-xr-x 1 book book 72368 2月 28 16:28 genimage
-rw-r--r-- 1 book book 523264 2月 28 16:29 u-boot-dtb.imx
-rw-rw-r-- 1 book book 350 2月 28 16:31 ubuntu-genimage.cfg
-rwxr--r-- 1 book book 2147483648 2月 28 16:30 ubuntu-rootfs.ext4
book@book-virtual-machine:~$
```

此时我们可以执行如下命令来生成 `ubuntu-16.04-armhf_100ask_imx6ull.img` 系统镜像文件。

```
book@book-virtual-machine:~$ mkdir root
book@book-virtual-machine:~$ sudo ./genimage --inputpath ./ --outputpath ./ --config ./ubuntu-genimage.cfg
```

执行步骤如下图所示，执行完成后会生成一个 `ubuntu-16.04-armhf_100ask_imx6ull.img` 文件可以直接用来烧写 `sd` 卡或者 `emmc` 启动。

```
book@book-virtual-machine:~$ mkdir root
book@book-virtual-machine:~$ sudo ./genimage --inputpath ./ --outputpath ./ --config ./ubuntu-genimage.cfg
INFO: cmd: "rm -rf "/home/book/tmp/*" (stderr):
INFO: cmd: "mkdir -p "/home/book/*" (stderr):
INFO: cmd: "mkdir -p "/home/book/tmp" (stderr):
INFO: cmd: "cp -a "/home/book/root" "/home/book/tmp/root" (stderr):
INFO: hdimage(ubuntu-16.04-armhf_100ask_imx6ull.img): adding partition 'u-boot' from 'u-boot-dtb.imx' ...
INFO: hdimage(ubuntu-16.04-armhf_100ask_imx6ull.img): adding partition 'arduino' (in MBR) ...
INFO: hdimage(ubuntu-16.04-armhf_100ask_imx6ull.img): adding partition 'rootfs' (in MBR) from 'ubuntu-rootfs.ext4' ...
INFO: cmd: "rm -rf "/home/book/tmp/*" (stderr): writing MBR
book@book-virtual-machine:~$
```

1.4 启动后常见问题

1.4.1 网络相关问题

1.4.1.1 不能上网

配置 **dns** 服务器文件

```
vim /etc/resolv.conf
```

添加你的 **dns** 地址

至于怎么获取就不过多阐述

```
route add default gw 192.168.1.1
```