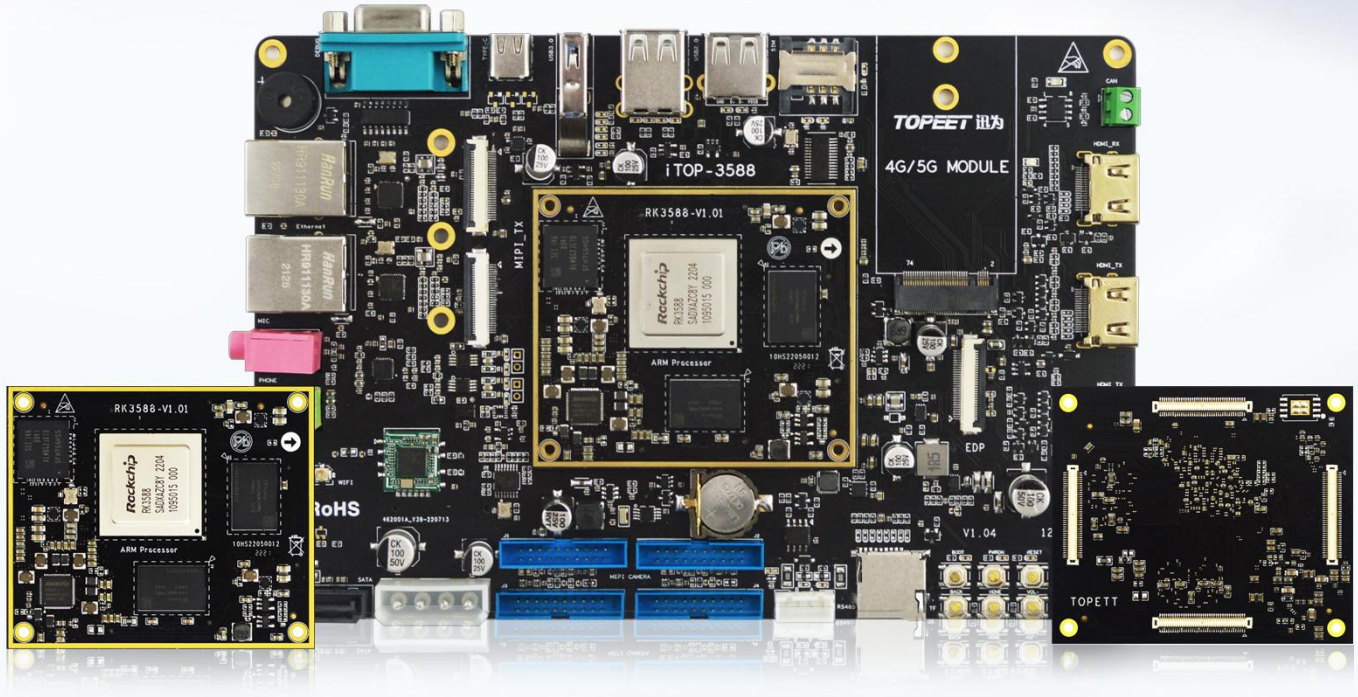
TOPEET北京迅为

强大的 AI 能力 更快更强

超长供货周期 | 7X24 小时稳定运行 | 8K 视频编解码



iTOP-RK3588 开发板使用手册

八核 64 位 CPU | 主频 2.4GHz | NPU 算力 6T | 4800 安防级别 ISP

更新记录

更新版本	修改内容
V1.0	初版
V1.1	buildroot 系统密码更新为 topeet

目录

版权声明

更多帮助

注意事项与维护

- ❖ 请注意和遵循标注在产品上的所有警示和指引信息；
- ❖ 请勿带电插拔核心板及外围模块；
- ❖ 使用产品之前，请仔细阅读本手册，并妥善保管，以备将来参考；
- ❖ 请使用配套电源适配器，以保证电压、电流的稳定；
- ❖ 请勿在冷热交替环境中使用本产品，避免结露损坏元器件；
- ❖ 请保持产品干燥，如果不慎被任何液体泼溅或浸润，请立刻断电并充分晾干；
- ❖ 请勿使用有机溶剂或腐蚀性液体清洗本产品；
- ❖ 请勿在多尘、脏乱的环境中使用本产品，如果长期不使用，请包装好本产品；
- ❖ 如果在震动场景使用，请做好核心板与底板的固定，避免核心板跌落损坏；
- ❖ 请勿在通电情况下，插拔核心板及外围模块(特别是串口模块)；
- ❖ 请勿自行维修、拆解本产品，如产品出现故障应及时联系本公司进行维修；
- ❖ 请勿自行修改或使用未经授权的配件，由此造成的损坏将不予保修；

资料的更新

为了确保您的资料是最新状态，请密切关注我们的动态，我们将会通过微信公众号和 QQ 群推送。

关注“迅为电子”微信公众号，不定期分享教程、资料和行业干货及产品一线资料。

迅为新媒体账号

官网：https://www.topeetboard.com

知乎 https://www.zhihu.com/people/topeetabc123

CSDN：https://blog.csdn.net/BeiJingXunWei

售后服务政策

1. 如产品使用过程中出现硬件故障可根据售后服务政策进行维修
2. 服务政策：参见官方网售后服务说明 https://www.topeetboard.com/sydymfl/Service/bx.html

送修地址：

1. 地址：北京市海淀区永翔北路 9 号中国航发大厦三层
2. 联系人：迅为开发板售后服务部
3. 电话：010-85270716
4. 邮编：100094
5. 邮寄须知：建议使用顺丰、圆通或韵达，且不接受任何到付

技术支持范围

1. 了解产品的软、硬件资源提供情况咨询
2. 产品的软、硬件手册使用过程中遇到的问题
3. 下载和烧写更新系统过程中遇到的问题
4. 产品用户的资料丢失、更新后重新获取
5. 产品的故障判断及售后维修服务。

PS：（由于嵌入式系统知识范围广泛，我们无法保证对各种问题都能一一解答，部分内容无法供技术支持，只能提供建议。 ）

技术支持

1. 周一至周五：（法定节假日除外）
上午 9:00 ~ 11:30 / 下午 13:30 ~ 17:30

2. QQ 技术交流群： 824412014
822183461
95631883
861311530

第 1 章 Buildroot 系统开发

1.1 Buildroot 系统设置屏幕旋转

迅为支持的触摸屏幕有四种：

MIPI 7 寸屏幕（默认物理屏幕为竖屏）

LVDS 7 寸屏幕（默认物理屏幕为竖屏）

LVDS 10.1 寸 1024*600 屏幕（默认物理屏幕为横屏）

HDMI 屏幕（默认物理屏幕为横屏）

本文档将 buildroot 系统不同屏幕如何旋转屏幕。Buildroot 系统启动的过程中，屏幕会依次显示 uboot logo，kernel logo，最后显示桌面。所以如果想要手中的屏幕由竖屏变为横屏或者横屏变为竖屏，首先要旋转 uboot logo，kernel logo，然后旋转文件系统的桌面和触摸。

1.1.1 设置屏幕

查看《06【北京迅为】itop-3588 开发板源码编译手册》手册中 Linux 源码编译设置屏幕章节。

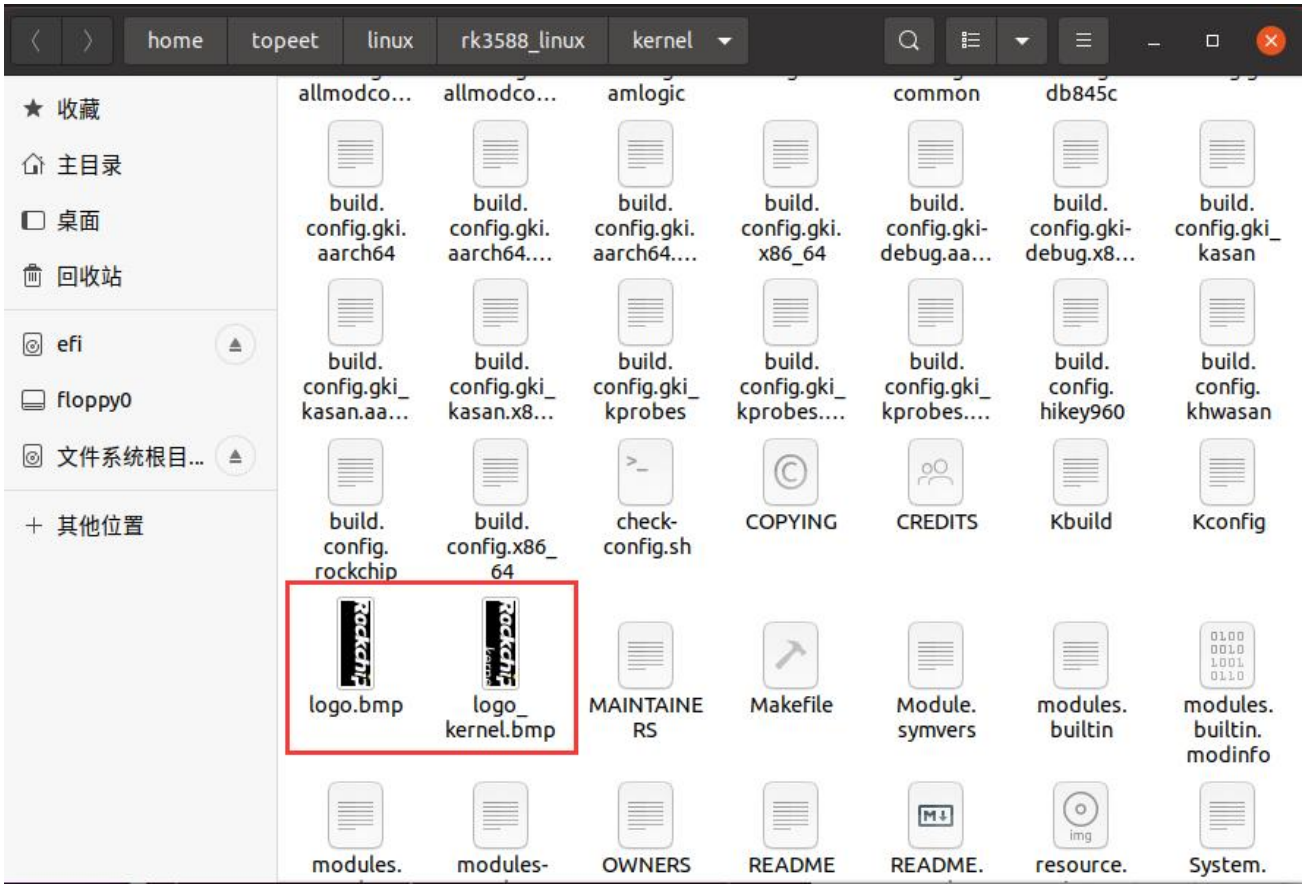
1.1.2 旋转 Uboot logo 和内核 logo

在终端内核目录下输入以下命令，以 root 权限打开文件夹，如图所示：

```
nautilus .
```

```
root@ubuntu:/home/topeet/linux/rk3588_linux/kernel#
root@ubuntu:/home/topeet/linux/rk3588_linux/kernel#
root@ubuntu:/home/topeet/linux/rk3588_linux/kernel# nautilus .
```

直接旋转 logo 图片即可，进入源码 kernel 目录下，旋转图片 logo.bmp 和 logo_kernel.bmp 然后保存，如下图所示：



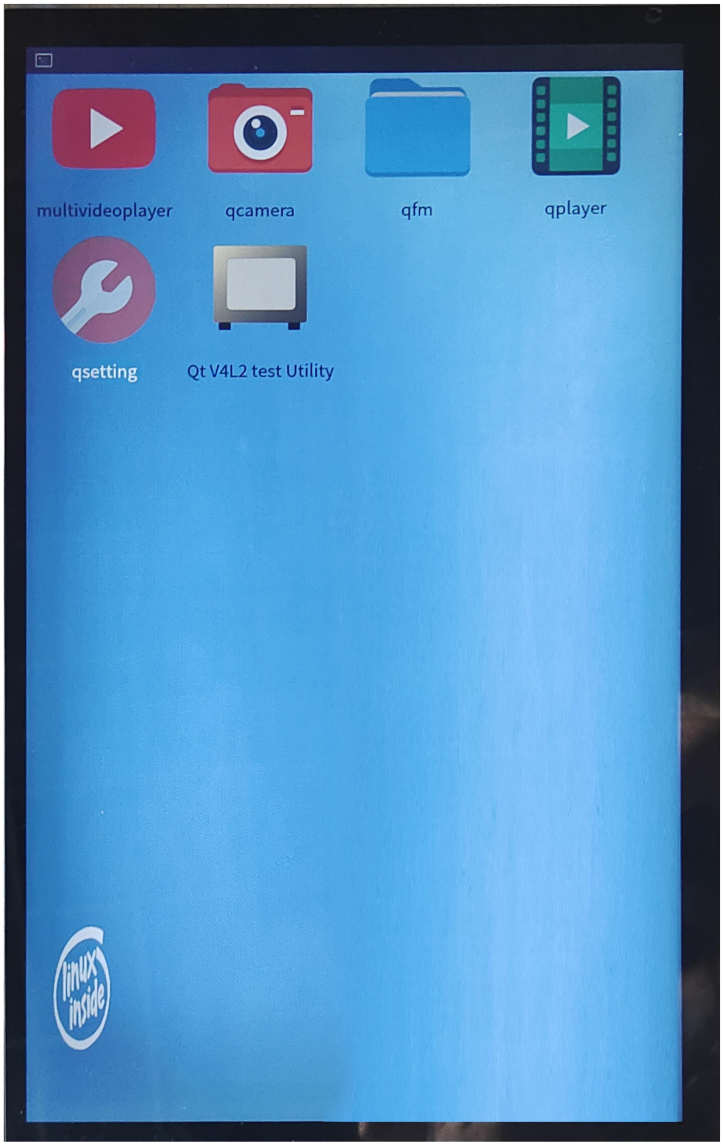
1.1.3 旋转文件系统桌面

Buildroot 系统动态旋转桌面，在串口终端输入以下命令：

```
echo "output:all:rotate270" > /tmp/.weston_drm.conf
```

设置完毕，默认为竖屏显示的屏幕，转为横屏显示，如下图所示：

设置完毕，默认为横屏显示的屏幕，转为竖屏显示，如下图所示：



上述方法输入的命令是“echo "output:all:rotate270" > /tmp/.weston_drm.conf”,其中 all 可替换为屏幕具体的型号比如：DSI-1 等等。旋转的角度可以自定义设置，可设置为：90,180,270,360。

上述命令中，旋转显示的同时，触摸也会跟随旋转。但是重启开发板命令会失效，我们可以将设置旋转桌面的命令写入开机自启动文件中，即可实现开机自动旋转屏幕。

1.2 Buildroot 系统设置待机和锁屏

Weston 的超时待机时长可以在 weston.ini 的 core 段配置。

修改文件系统中/etc/xdg/weston/weston.ini 文件，比如设置 5 秒未操作后进入待机状态，则如下配置:

```
[core]
backend=drm-backend.so

# Allow running without input devices
require-input=false

# Disable screen idle timeout by default
idle-time=5

# The repaint-window is used to calculate repaint delay(ms) after flipped.
#   value <= 0: delay = abs(value)
#   value > 0: delay = vblank_duration - value
repaint-window=-1
```

1.3 Buildroot 系统设置显示颜色格式

Buildroot SDK 内 Weston 目前默认显示格式为 ARGB8888,对于某些低性能平台,可以在 weston.ini 的 core 段配置为 RGB565。

修改文件系统中/etc/xdg/weston/weston.ini 文件，修改如下所示：

```
[core]
gbm-format=rgb565
```

1.4 Buildroot 系统设置分辨率和缩放

Weston 的屏幕分辨率及缩放可以在 weston.ini 的 output 段配置，修改文件系统的/etc/xdg/weston/weston.ini 文件，修改如下：

```
[output]
name=HDMI-A-1
# 需为屏幕支持的有效分辨率
mode=1920x1080
# 需为整数倍数
scale=2
```

如果需要动态配置分辨率及缩放,可以通过动态配置文件,这种方式缩放时需要依赖 RGA 加速。在串口终端输入以下命令配置：

```
# 修改 HDMI-A-1 分辨率为 800x600
echo "output:HDMI-A-1:mode=800x600" > /tmp/.weston_drm.conf
```

1.5 Buildroot 系统设置状态栏

Weston 支持在 weston.ini 配置文件的 shell 段设置状态栏的背景色、位置,以及在 launcher 段设置快捷启动程序，修改文件系统的/etc/xdg/weston/weston.ini 文件，如下所示：

```
[shell]
# 颜色格式为 ARGB8888
panel-color=0xff002244
# top(default)|bottom|left|right|none, none to disable panel
panel-position=bottom
[launcher]
icon=/usr/share/weston/terminal.png
path=/usr/bin/weston-terminal
[launcher]
# 图标路径
icon=/usr/share/weston/icon_flower.png
# 快捷启动命令
path=/usr/bin/qsetting
```

Weston 支持在 weston.ini 配置文件的 shell 段设置背景图案、颜色，修改 /etc/xdg/weston/weston.ini 文件，如下所示：

```
[shell]
# 背景图案(壁纸)绝对路径
background-image=/usr/share/weston/background.png
# scale|scale-crop|tile
background-type=scale
# 颜色格式为 ARGB8888,未设置背景图案时生效
background-color=0xff002244
```

1.6 Buildroot 取消默认 QT 桌面

本小节将讲解如何取消掉默认的 qt 桌面。

首先对开发板进行上电，开发板正常启动后，使用命令“cd /etc/init.d”进入到/etc/init.d 目录下，然后使用以下命令对开机自启动脚本 rcS 进行查看，如下图所示：

```
vi rcS
```

```
#!/bin/sh

# Start all init scripts in /etc/init.d
# executing them in numerical order.
#
for i in /etc/init.d/S??* ;do

     # Ignore dangling symlinks (if any).
     [ ! -f "$i" ] && continue

     case "$i" in
        *.sh)
            # Source shell script for speed.
            (
                trap - INT QUIT TSTP
                set start
                . $i
            )
            ;;
        *)
            # No sh extension, so fork subprocess.
            $i start
            ;;
    esac
done
```

从上图可以看出，开机自启动脚本 rcS 默认会在/etc/init.d 目录下查找所有以“S”开头的脚本并依次执行，而默认 QT 桌面是由 S50launcher 脚本启动的，退出 rcS 文件，回到/etc/init.d 目录下。

使用以下命令对 S50launcher 文件进行查看修改，进入文件之后如下图所示：

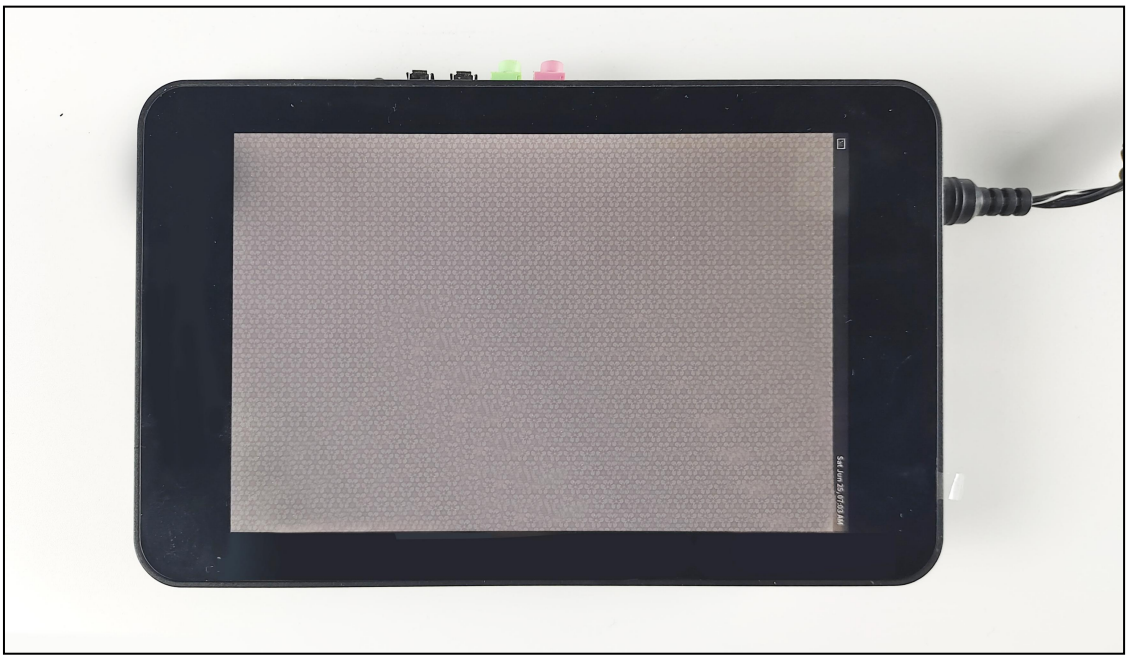
```
vi S50launcher
```

```
# Load default env variables from profiles(e.g. /etc/profile.d/qlauncher.sh)
. /etc/profile

start_qlauncher()
{
        # Wait for weston ready
        [ "${QT_QPA_PLATFORM}" == wayland ] && \
                while [ ! -e ${XDG_RUNTIME_DIR}/wayland-0 ]; do
                        sleep .1
                done

        /usr/bin/QLauncher&
}

stop_qlauncher()
{
        killall QLauncher
}

case "$1" in
        start)
                echo -n "starting qlauncher... "
                start_qlauncher
                echo "done."
                ;;
```

我们将运行 Qlauncher 桌面代码进行注释，就可以取消默认的 Qlauncher 桌面了，注释完成如下图所示：

```
start_qlauncher()
{
        # Wait for weston ready
        [ "${QT_QPA_PLATFORM}" == wayland ] && \
                while [ ! -e ${XDG_RUNTIME_DIR}/wayland-0 ]; do
                        sleep .1
                done

        #/usr/bin/QLauncher&
}
```

保存退出之后，重新启动开发板，就会发现默认的 QT 桌面已经被取消了。如下图所示：



1.7 Buildroot 系统自启动 QT 程序

本小节将讲解如何开机自启动 QT 程序。

在网盘“iTOP-3588 开发板\02_【iTOP-RK3588 开发板】开发资料 \09_Linux 系统开发配套资料\01_Buildroot 系统开发配套\01_QT 可执行程序及图标”目录下，迅为提供了 QTdemo 程序，我们以此为例测试开机自启动 QT 程序。

首先使用命令“cd /etc/init.d/”进入到/etc/init.d 目录下，然后使用命令“vi S50launcher”对 S50launcher 文件进行修改，进入文件之后如下图所示：

```
#!/bin/sh -e
### BEGIN INIT INFO
# Provides:          qlauncher
# Required-Start:    mountvirtfs
# Required-Stop:
# Should-Start:
# Should-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Qt launcher
### END INIT INFO

PATH="/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin"

# Load default env variables from profiles(e.g. /etc/profile.d/qlauncher.sh)
. /etc/profile

start_qlauncher()
{
        # Wait for weston ready
        [ "${QT_QPA_PLATFORM}" == wayland ] && \
                while [ ! -e ${XDG_RUNTIME_DIR}/wayland-0 ]; do
                        sleep .1
                done
```

然后注释掉之前的桌面程序，并在下方添加自己的 qt 程序，在这里以 QCalendar 例程为例进行示范，修改完成如下图所示：

```
/usr/bin/QCalendar &
```

```
start_qlauncher()
{
        # Wait for weston ready
        [ "${QT_QPA_PLATFORM}" == wayland ] && \
                while [ ! -e ${XDG_RUNTIME_DIR}/wayland-0 ]; do
                        sleep .1
                done
        #/usr/bin/QLauncher&
        /usr/bin/QCalendar &
}
```

注释之前的桌面程序

添加自己的qt程序

重启开发板之后，就会看到 QCalendar 日历 QT 程序已经默认启动了，如下图所示：

1.8 Buildroot 系统创建 QT 程序桌面快捷方式

本小节将讲解如何创建 QT 程序桌面快捷方式。

首先对开发板进行上电，开发板启动后，使用以下命令对 S50launcher 文件进行查看修改

```
vi etc/init.d/S50launcher
```

由于本小节需要用到默认桌面，如果根据前两小节对 S50launcher 文件添加或者注释过，需要删除添加内容和取消掉对应的注释，修改完成如下图所示：

```
start_qlauncher()
{
        # Wait for weston ready
        [ "${QT_QPA_PLATFORM}" == wayland ] && \
                while [ ! -e ${XDG_RUNTIME_DIR}/wayland-0 ]; do
                        sleep .1
                done
        /usr/bin/QLauncher&
        #/usr/bin/QCalendar &
}

stop_qlauncher()
{
        killall QLauncher
}

case "$1" in
        start)
                echo -n "starting qlauncher... "
                start_qlauncher
                echo "done."
                ;;
        stop)
```

保存退出之后，使用命令“cd /usr/share/applications”进入到 applications 目录下如下图所示：

```
root@RK3588:/mnt# cd /usr/share/applications/
root@RK3588:/usr/share/applications#
root@RK3588:/usr/share/applications#
root@RK3588:/usr/share/applications# ls
chromium.desktop  multivideoplayer.desktop  qfm.desktop      qsetting.desktop
mimeapps.list     qcamera.desktop           qplayer.desktop  qv4l2.desktop
root@RK3588:/usr/share/applications#
root@RK3588:/usr/share/applications#
```

Qlauncher 桌面程序会读取以.desktop 为结尾的文件内容，并根据相应的内容进行 QT 桌面应用的添加。使用以下命令对 qplayer.desktop 文件内容进行查看，具体内容如下所示：

```
vi qplayer.desktop
```

```
1 [Desktop Entry]
2 Name=qplayer
3 Exec=/usr/bin/qplayer
4 Icon=/usr/share/icon/icon_player.png
5 Type=Application
```

2 到 5 行参数解释如下：

Name：在默认桌面显示的名称

Exec：运行 QT 程序所要用的命令（使用的是绝对路径）

Icon：应用的图标存放路径

Type：QT 应用程序类型，一般都为 Application 类型

下面我们为 QCalendar 程序创建桌面快捷方式。

首先准备好 QT 可执行程序和图标文件，这里 QT 可执行文件名称为 QCalendar ，对应的

图标文件为 icon_QCalendar.png，测试资料在网盘“iTOP-3588 开发板\02_【iTOP-RK3588 开发板】开发资料 \09_Linux 系统开发配套资料\01_Buildroot 系统开发配套\01_QT 可执行程序及图标”目录下。如下图所示：

然后使用以下命令分别将可执行文件和对应的图标文件拷贝到开发板的/usr/bin/和/usr/share/icon/目录下，如下图所示：

```
cp QCalendar /usr/bin/
cp icon_QCalendar.png /usr/share/icon/
```

```
root@RK3588:/mnt#
root@RK3588:/mnt# cp QCalendar /usr/bin/
root@RK3588:/mnt# cp icon_QCalendar.png /usr/share/icon/
root@RK3588:/mnt#
```

然后使用以下命令进入“/usr/share/applications”文件如下图所示：

```
cd /usr/share/applications/
```

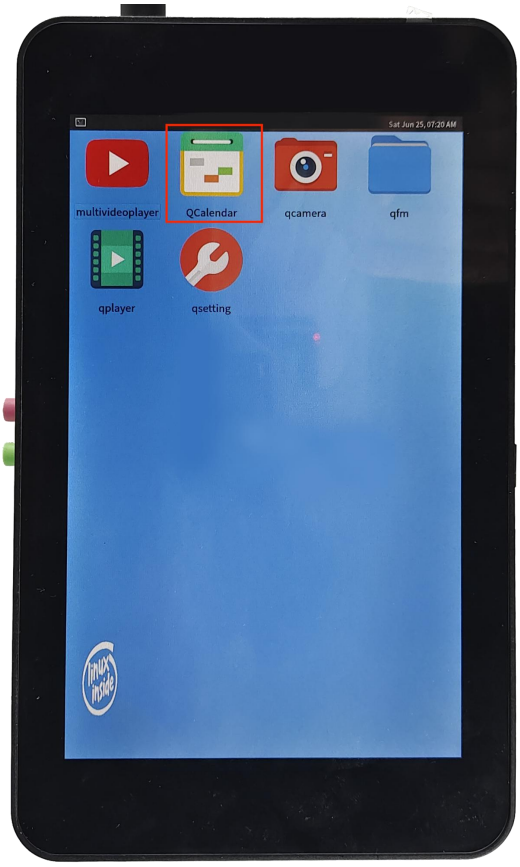
```
root@RK3588:/mnt# cd /usr/share/applications/
root@RK3588:/usr/share/applications#
root@RK3588:/usr/share/applications#
root@RK3588:/usr/share/applications# ls
chromium.desktop  multivideoplayer.desktop  qfm.desktop      qsetting.desktop
mimeapps.list     qcamera.desktop           qplayer.desktop  qv4l2.desktop
root@RK3588:/usr/share/applications#
root@RK3588:/usr/share/applications#
```

随后使用命令“vi QCalendar.desktop”创建 QCalendar.desktop 文件并向其中添加以下内容，添加完成如下图所示：

```
[Desktop Entry]
Name=QCalendar
Exec=/usr/bin/QCalendar
Icon=/usr/share/icon/icon_QCalendar.png
Type=Application
```

```
1 [Desktop Entry]
2 Name=QCalendar
3 Exec=/usr/bin/QCalendar
4 Icon=/usr/share/icon/icon_QCalendar.png
5 Type=Application
```

保存退出，重启开发板，会发现我们新添加的 QT 程序已经出现在了桌面上，如下图所示：



然后点击该图标，就会运行我们的 QT 程序，QT 程序运行成功如下图所示：

至此，添加桌面快捷方式完成！

1.9 Buildroot 系统设置开机密码登录

首先对开发板进行上电，开发板正常启动后，使用命令“vi /etc/inittab”对文件进行修改，如下图所示：

```
::sysinit:/bin/mount -t proc proc /proc
::sysinit:/bin/mount -o remount,rw /
::sysinit:/bin/mkdir -p /dev/pts /dev/shm
::sysinit:/bin/mount -a
::sysinit:/sbin/swapon -a
null::sysinit:/bin/ln -sf /proc/self/fd /dev/fd
null::sysinit:/bin/ln -sf /proc/self/fd/0 /dev/stdin
null::sysinit:/bin/ln -sf /proc/self/fd/1 /dev/stdout
null::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr
::sysinit:/bin/hostname -F /etc/hostname
# now run any rc scripts
::sysinit:/etc/init.d/rcS

# Put a getty on the serial port
::respawn:-/bin/sh # ttyS0::respawn:/sbin/getty -L ttyS0 115200 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
::shutdown:/etc/init.d/rcK
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r
~
~
```

无需密码登录

设置为密码登陆时配置如下图（注意将 ttyS0 修改为 ttyFIQ0）：

```
null::sysinit:/bin/ln -sf /proc/self/fd /dev/fd
null::sysinit:/bin/ln -sf /proc/self/fd/0 /dev/stdin
null::sysinit:/bin/ln -sf /proc/self/fd/1 /dev/stdout
null::sysinit:/bin/ln -sf /proc/self/fd/2 /dev/stderr
::sysinit:/bin/hostname -F /etc/hostname
# now run any rc scripts
::sysinit:/etc/init.d/rcS

# Put a getty on the serial port
#::respawn:-/bin/sh
ttyFIQ0::respawn:/sbin/getty -L ttyFIQ0 115200 vt100 # GENERIC_SERIA

# Stuff to do for the 3-finger salute
#::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
::shutdown:/etc/init.d/rcK
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r
~
```

需要密码登录

修改完，保存退出，重启开发板，输入用户名：root，密码：topeet，登入到系统，如下图所示：

```
[   34.647493] vcc3v3_pcie30: disabling
[   34.647532] vcc_3v3_sd_s0: disabling
[   34.647544] vbus5v0_typec: disabling

Welcome to RK3588 Buildroot
RK3588 login: root
Password:
root@RK3588:~#
root@RK3588:~#
```

用户名root
密码 topeet

1.10 设置静态 IP

在一些情况下，需要设置静态 IP，本小节将描述 buildroot 系统下设置静态 IP 的方法。烧写完 buildroot 系统之后如下图所示：

```
root@RK3588:/#
root@RK3588:/# ls
8723du.ko          etc      lost+found  opt            run     tmp       vendor
bin                info     media       proc           sbin    udisk
busybox.fragment   lib      misc        rockchip-test  sdcard  userdata
data               lib64    mnt         root           sys     usr
dev                linuxrc  oem         rtk_btusb.ko   system  var
root@RK3588:/#
```

由于默认情况下，buildroot 系统是通过 dhcp 自动获取 ip 的，所以需要先使用以下命令取消开机启动项中的 dhcp 相关内容，如下图所示：

```
rm -rf /etc/init.d/S41dhcpcd
```

```
root@RK3588:/#
root@RK3588:/# rm -rf /etc/init.d/S41dhcpcd
root@RK3588:/#
```

而由于 dns 文件/etc/resolv.conf 是一个指向/tmp/resolv.conf 的软链接，使用静态获取 IP 的方式不能正常解析 DNS 文件，所以需要使用以下命令删掉软链接文件并重新创建，如下图所示：

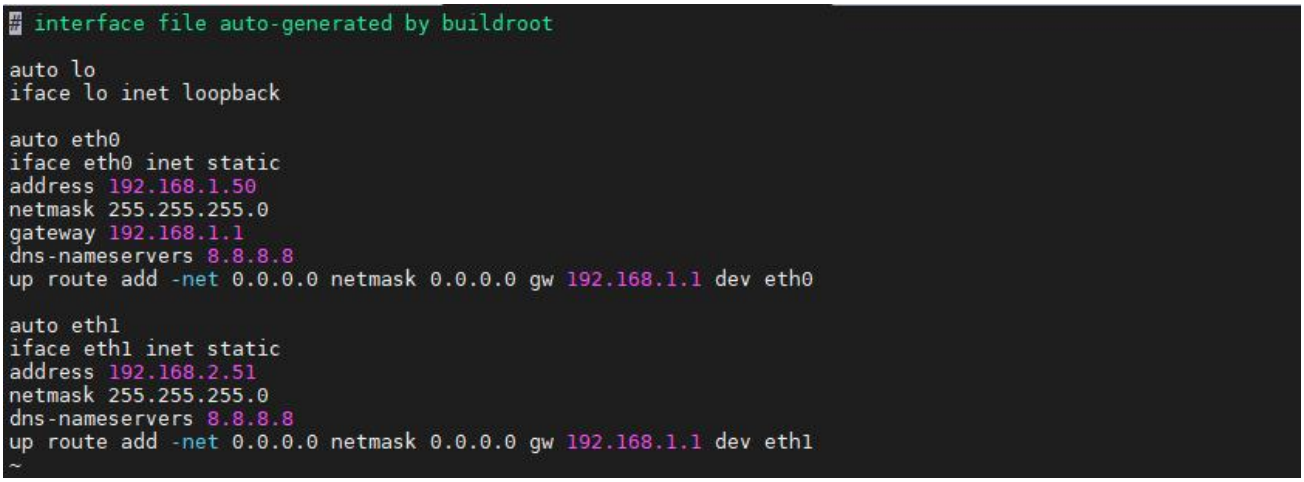
```
rm -rf /etc/resolv.conf
echo nameserver 8.8.8.8 >>/etc/resolv.conf
```

```
root@RK3588:/#
root@RK3588:/# ls -al /etc/resolv.conf
lrwxrwxrwx 1 root root 18 Jun  7  2022 /etc/resolv.conf -> ../tmp/resolv.conf
root@RK3588:/#
root@RK3588:/# rm -rf /etc/resolv.conf
root@RK3588:/#
root@RK3588:/#
root@RK3588:/# echo nameserver 8.8.8.8 >>/etc/resolv.conf
sh: $'echo\302\240nameserver': command not found
root@RK3588:/# echo nameserver 8.8.8.8 >>/etc/resolv.conf
root@RK3588:/# cat /etc/resolv.conf
nameserver 8.8.8.8
root@RK3588:/#
```

最后使用“vi /etc/network/interfaces”命令修改网络配置文件，添加以下内容

```
auto eth0
iface eth0 inet static
address 192.168.1.50
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 8.8.8.8
up route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.1.1 dev eth0

auto eth1
iface eth1 inet static
address 192.168.2.51
netmask 255.255.255.0
dns-nameservers 8.8.8.8
up route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.1.1 dev eth1
```

添加完成如下图所示:



至此，关于静态 ip 的设置就完成，接下来重启开发板，重启完成之后分别使用“ifconfig”命令查看 ip 地址，如下图所示：

```
root@RK3588:/#
root@RK3588:/# ifconfig
eth0      Link encap:Ethernet  HWaddr B2:A2:DA:AC:65:D3
          inet addr:192.168.1.50  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::b0a2:daff:feac:65d3/64 Scope:Link
          inet6 addr: 240e:341:d09d:7100:b0a2:daff:feac:65d3/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31 errors:0 dropped:2 overruns:0 frame:0
          TX packets:25 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2714 (2.6 KiB)  TX bytes:1998 (1.9 KiB)
          Interrupt:80

eth1      Link encap:Ethernet  HWaddr AE:A2:DA:AC:65:D3
          inet addr:192.168.2.51  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::aca2:daff:feac:65d3/64 Scope:Link
          inet6 addr: 240e:341:d09d:7100:aca2:daff:feac:65d3/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27 errors:0 dropped:2 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2084 (2.0 KiB)  TX bytes:602 (602.0 B)
          Interrupt:120

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:316 (316.0 B)  TX bytes:316 (316.0 B)

root@RK3588:/#
```

可以看到 eth0 和 eth1 的地址已经成功设置为了 192.168.1.50 和 192.168.2.51，然后分别受用以下命令测试两个网口的网络是否正常，如下图所示：

```
ping www.baidu.com -I eth0
ping www.baidu.com -I eth1
```

```
root@RK3588:/#
root@RK3588:/# ping www.baidu.com -I eth0
PING www.baidu.com(240e:83:205:5a:0:ff:b05f:346b) from 240e:341:d09d:7100:b0a2:daff:feac:65d3 eth0: 56 data bytes
64 bytes from 240e:83:205:5a:0:ff:b05f:346b: icmp_seq=1 ttl=54 time=9.86 ms
64 bytes from 240e:83:205:5a:0:ff:b05f:346b: icmp_seq=2 ttl=54 time=10.0 ms
64 bytes from 240e:83:205:5a:0:ff:b05f:346b: icmp_seq=3 ttl=54 time=10.1 ms
64 bytes from 240e:83:205:5a:0:ff:b05f:346b: icmp_seq=4 ttl=54 time=9.94 ms
64 bytes from 240e:83:205:5a:0:ff:b05f:346b: icmp_seq=5 ttl=54 time=9.86 ms
64 bytes from 240e:83:205:5a:0:ff:b05f:346b: icmp_seq=6 ttl=54 time=9.89 ms
64 bytes from 240e:83:205:5a:0:ff:b05f:346b: icmp_seq=7 ttl=54 time=10.0 ms
64 bytes from 240e:83:205:5a:0:ff:b05f:346b: icmp_seq=8 ttl=54 time=9.95 ms
^C
--- www.baidu.com ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7012ms
rtt min/avg/max/mdev = 9.858/9.945/10.065/0.070 ms
root@RK3588:/#
root@RK3588:/#
root@RK3588:/#
root@RK3588:/# ping www.baidu.com -I eth1
PING www.baidu.com(240e:83:205:58:0:ff:b09f:36bf) from 240e:341:d09d:7100:aca2:daff:feac:65d3 eth1: 56 data bytes
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=1 ttl=54 time=12.2 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=2 ttl=54 time=12.1 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=3 ttl=54 time=12.1 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=4 ttl=54 time=12.1 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=5 ttl=54 time=12.2 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=6 ttl=54 time=12.1 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=7 ttl=54 time=12.0 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=8 ttl=54 time=12.0 ms
^C
--- www.baidu.com ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7011ms
rtt min/avg/max/mdev = 11.952/12.107/12.201/0.081 ms
root@RK3588:/#
```

1.11 创建自启动程序

首先对开发板进行上电，开发板正常启动后，使用命令“cd /etc/init.d”进入到/etc/init.d 目录

下，然后使用以下命令对开机自启动脚本 rcS 进行查看，如下图所示：

```
vi rcS
```

```
#!/bin/sh

# Start all init scripts in /etc/init.d
# executing them in numerical order.
#
for i in /etc/init.d/S??* ;do

     # Ignore dangling symlinks (if any).
     [ ! -f "$i" ] && continue

     case "$i" in
        *.sh)
            # Source shell script for speed.
            (
                trap - INT QUIT TSTP
                set start
                . $i
            )
            ;;
        *)
            # No sh extension, so fork subprocess.
            $i start
            ;;
    esac
done
```

从上图可以看出，开机自启动脚本 rcS 默认会在/etc/init.d 目录下查找所有以“S”开头的脚本并依次执行，因此我们在/etc/init.d 目录下新建以“S”开头的脚本，在脚本中添加想要开机自启动的程序即可。

我们以添加开机自动加载 WiFi 模块驱动为例，首先在/etc/init.d 目录下新建以“S”开头的脚本 S99，如下图所示：

```
root@RK3588:/# ls /etc/init.d/S99
/etc/init.d/S99
root@RK3588:/#
root@RK3588:/#
```

然后在脚本中添加加载驱动命令，如下图所示：

```
root@RK3588:/# cat /etc/init.d/S99
#! /bin/sh
mkdir -p /data/cfg
cp /etc/wpa_supplicant.conf /data/cfg
cp /usr/lib/firmware/rtl8723du_* /lib/firmware/
insmod /8723du.ko
insmod /rtk_btusb.ko
root@RK3588:/#
```

按自己驱动路径修改

添加完成后，使用以下，命令给予脚本权限，如下图所示：

```
root@RK3588:/#
root@RK3588:/# chmod 777 /etc/init.d/S99
root@RK3588:/#
```

修改完脚本权限重启开发板即可。

1.12 瑞芯微 Buildroot 编译

Rockchip Buildroot Linux SDK 是基于 Buildroot-2021.11 的版本的软件开发包,其包含了基于 Linux 系统开发用到的各种系统源码,驱动,工具,应用软件包。Buildroot 是 Linux 平台上一个开源的嵌入式 Linux 系统自动构建框架。整个 Buildroot 是由 Makefile 脚本和 Kconfig 配置文件构成的。你可以通过 Buildroot 配置,编译出一个完整的可以直接烧写到机器上运行的 Linux 系统软件。

1.12.1 编译前配置

本小节介绍 buildroot 固件的编译流程，推荐在 Ubuntu20.04 系统环境下进行开发。进入 /home/topeet/Linux/3588_linux 根目录下选择开发板默认的配置文件 configs/rockchip_rk3588_defconfig。输入以下命令：

```
cd /home/topeet/Linux/3588_linux/buildroot
make rockchip_rk3588_defconfig
```

```
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$ make rockchip_rk3588_defconfig
/home/topeet/Linux/3588-linux-full/buildroot/build/parse_defconfig.sh /home/topeet/Linux/3588-linux-full/buildroot/configs/rockchip_rk3588_defconfig /home/topeet/Linux/3588-linux-fu
ll/buildroot/output/.config.in
Parsing defconfig: /home/topeet/Linux/3588-linux-full/buildroot/configs/rockchip_rk3588_defconfig
Using /home/topeet/Linux/3588-linux-full/buildroot/configs/rockchip_rk3588_defconfig as base
#
# merged configuration written to /home/topeet/Linux/3588-linux-full/buildroot/output/.config.in (needs make)
#
BR2_DEFCONFIG='' KCONFIG_AUTOCONFIG=/home/topeet/Linux/3588-linux-full/buildroot/output/build/buildroot-config/auto.conf KCONFIG_AUTOHEADER=/home/topeet/Linux/3588-linux-full/buildr
oot/output/build/buildroot-config/autoconf.h KCONFIG_TRISTATE=/home/topeet/Linux/3588-linux-full/buildroot/output/build/buildroot-config/tristate.config BR2_CONFIG=/home/topeet/Linu
x/3588-linux-full/buildroot/.config HOST_GCC_VERSION="9" BASE_DIR=/home/topeet/Linux/3588-linux-full/buildroot/output SKIP_LEGACY= CUSTOM_KERNEL_VERSION="5.10" BR2_DEFCONFIG=/home/t
opeet/Linux/3588-linux-full/buildroot/configs/rockchip_rk3588_defconfig /home/topeet/Linux/3588-linux-full/buildroot/output/build/buildroot-config/conf --defconfig=/home/topeet/Linu
x/3588-linux-full/buildroot/output/.config.in Config.in
#
# configuration written to /home/topeet/Linux/3588-linux-full/buildroot/.config
#
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$
```

执行之后会生成编译输出“.config”。

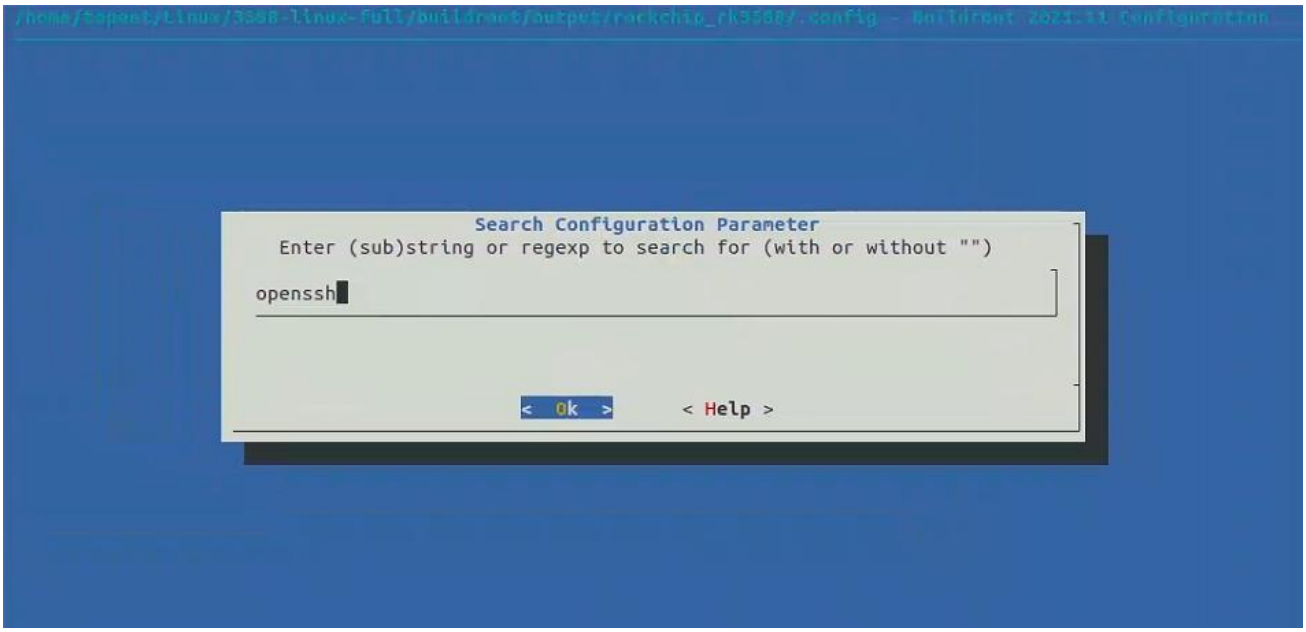
1.12.2 软件包配置

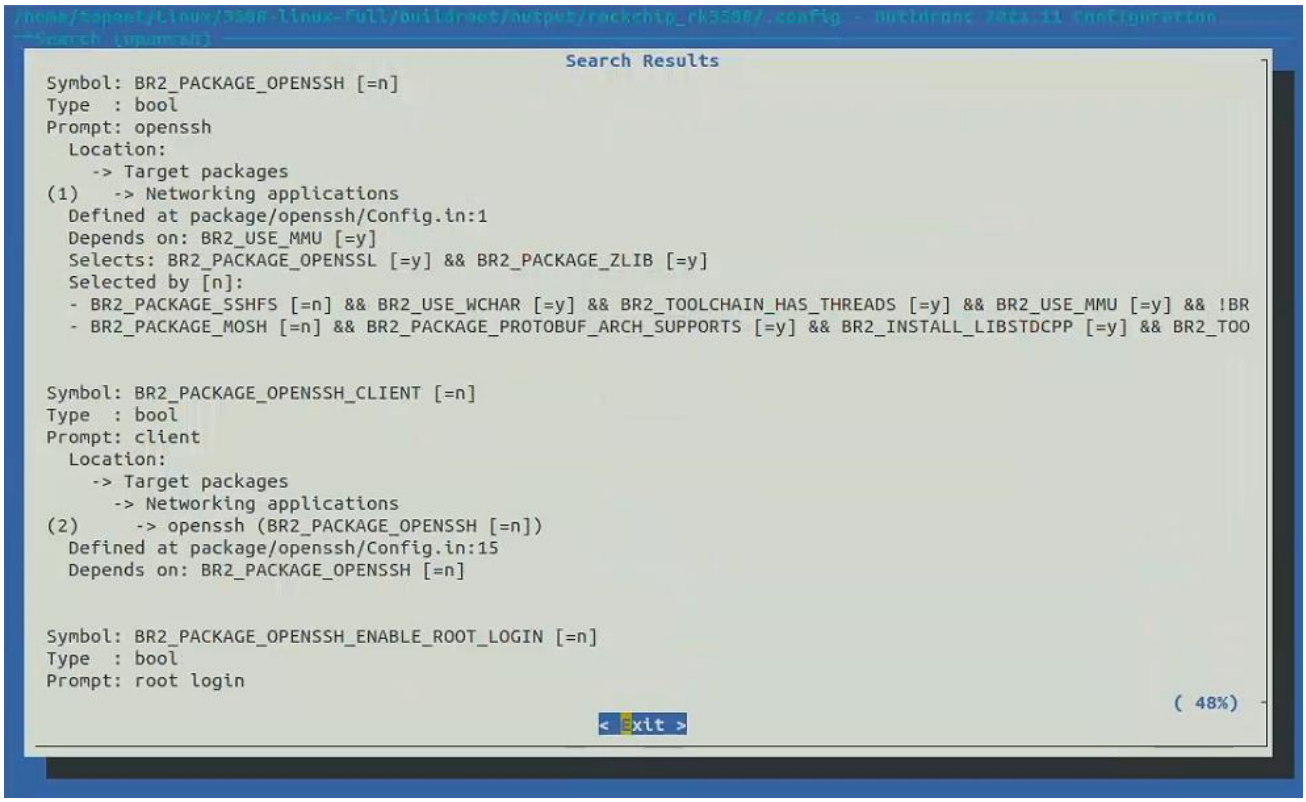
打开配置界面，输入以下命令：

```
make menuconfig
```

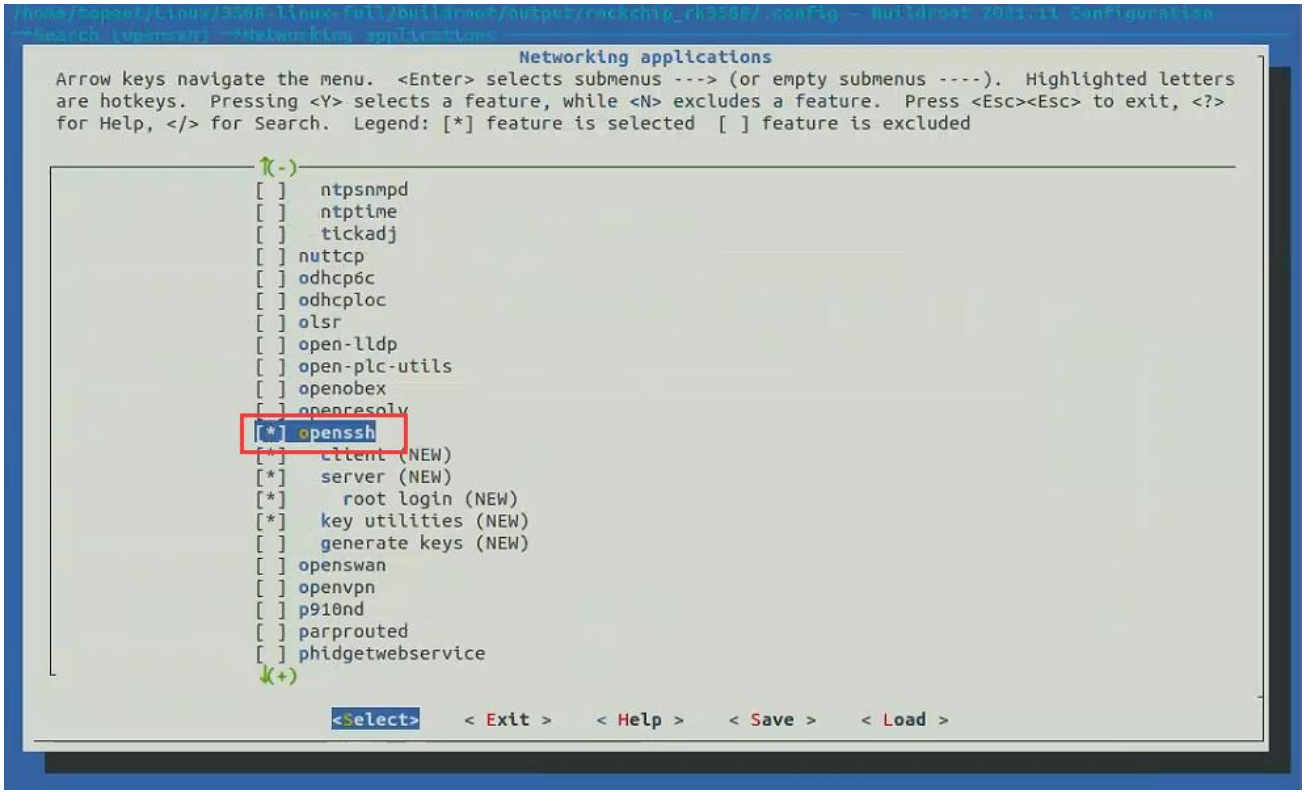
```
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$ make menuconfig
```

我们可以在配置界面添加一些工具，按需求定制系统，我们可以输入/进入搜索界面，然后输入要查找的内容。比如搜索 openssh,按回车进行搜索。

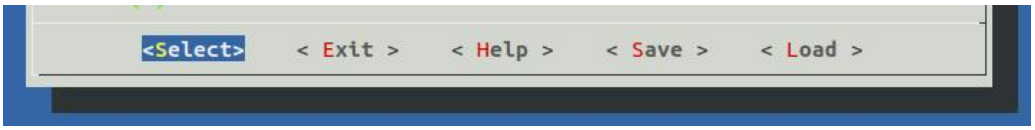


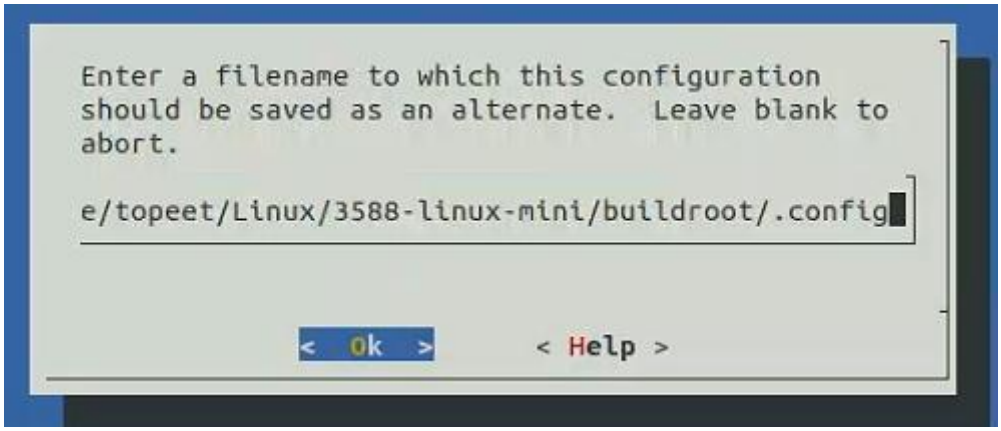


选择 1 跳转到对应的界面，按空格选中配置：



配置完成之后，按→或者←移动光标到 save，然后按回车保存到.config；然后移动到 Exit 按回车退出。





然后输入以下命令保存配置文件,将修改的.config 文件保存到默认的配置文件 configs/rockchip_rk3568_defconfig 中。

```
make savedefconfig
```

```
topeet@ubuntu:~/Linux/3588-linux-mini/buildroot$ make savedefconfig
/home/topeet/Linux/3588-linux-mini/buildroot /home/topeet/Linux/3588-linux-mini/buildroot/../prebuilts/gcc/linux-x86/
arm/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf GCC10
arm64 arm64 arm64
arm64
topeet@ubuntu:~/Linux/3588-linux-mini/buildroot$
```

也可以直接使用 cp 命令镜像拷贝覆盖，

```
cp .config configs/rockchip_rk3588_defconfig
```

```
topeet@ubuntu:~/Linux/3588-linux-mini/buildroot$
topeet@ubuntu:~/Linux/3588-linux-mini/buildroot$ cp .config configs/rockchip_rk3588_defconfig
```

接下来可以参考 06 编译手册编译 buildroot 系统。

1.12.3 buildroot 编译

在配置好 Buildroot 之后，输入“make”命令进行编译。在编译的时候会执行以下过程。

1 下载源码

2 配置，编译，安装交叉编译工具链

3 配置，编译，安装选择的软件包

4 按选择的格式生成根文件系统。

我们也可以输入“make <package>”命令单独编译某个软件包，软件包的编译主要包括下载，解压，打补丁，配置，编译，安装等过程。

1 buildroot 会根据配置 package/<package>/<package>.mk，自动从网络获取对应的软件包，包括一些第三方库，插件工具等，然后放在 dl 目录下。

2 软件包解压之后会放在 output/rockchip_rk3588/build/<package>-<version>目录下。

3 补丁放在 package/<packgae>/ 目录，buildroot 会在解压软件包后为其打上相应的补丁。如果要修改源码，可以通过打补丁的方式进行修改。

4 配置

5 编译

6 安装

编译完成之后，会将需要的编译生成文件拷贝到 output/rockchip_rk3588/target/ 目录下。

1.12.4 buildroot 重新构建

当使用图形化配置工具修改了系统配置的时候，buildroot 不会检测应该重建系统的哪些部分。在某种情况下，buildroot 应该重新编译重建整个系统或者重新编译某个软件包。当出现以下几种情况时，需要重新编译重建。

1 更改目标体系结构配置时，要完全编译重建

2 更改工具链时，要完全编译重建

3 将其他软件包添加到配置时，不一定需要完全编译重建

4 从配置中删除某个软件包时，buildroot 不会自动删除此软件包的安装文件，需要完全编译重建时才可以删除这些文件

5 更改软件包的子选项时，不会自动重建软件包

6 对根文件系统框架进行更改时，需要完全重建

那么如何重新编译，完全重建呢？

方法一：

直接删除 buildroot 的编译输出目录，之后重新进行配置，编译。

```
rm -rf output/
```

```
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$ rm -rf output/
```

方法二：

执行如下命令，会删除编译输出，之后重新进行配置，编译。

```
make clean all
```

```
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$ make clean all
```

在开发过程中，如果修改了某个软件包的源码，buildroot 是不会重新编译此软件包的。编译软件包有俩种方法。

方式一：

```
make <package>-rebuild
```

方式二，删除软件包的编译输出目录，然后重新编译

```
rm -rf output/rockchip_rk3588/build/<package>-<version>
make <package>
```

1.12.5 新增自定义的软件包

在开发过程中，Buildroot 自带的软件包有时可能无法满足我们的需求，所以有时候需要添加自定义的软件包。Buildroot 支持多种格式的软件包，包括 generic-package、cmake-package、autotools-package 等，我们以 generic-package 举例说明。

首先输入以下命令创建软件包目录

```
cd /home/topeet/Linux/3588_linux/buildroot
mkdir -p package/rockchip/topeet_demo
```

在 topeet_demo 目录下添加 Config.in 文件，Config.in 文件内容如下所示：

```
config BR2_PACKAGE_TOPEET_DEMO
        bool "Simple topeet Demo"
```

在 topeet_demo 目录下添加 topeet_demo.mk 文件，topeet_demo.mk 内容如下所示：

```
ifeq ($(BR2_PACKAGE_TOPEET_DEMO), y)

        TOPEET_DEMO_VERSION:=2.0.0
        TOPEET_DEMO_SITE=$(TOPDIR)/../external/topeet_demo/src
        TOPEET_DEMO_SITE_METHOD=local

define TOPEET_DEMO_BUILD_CMDS
        $(TARGET_MAKE_ENV) $(MAKE) CC=$(TARGET_CC) CXX=$(TARGET_CXX) -C
$(@D)
endef

define TOPEET_DEMO_CLEAN_CMDS
        $(TARGET_MAKE_ENV) $(MAKE) -C $(@D) clean
endef

define TOPEET_DEMO_INSTALL_TARGET_CMDS
        $(TARGET_MAKE_ENV) $(MAKE) -C $(@D) install
endef

define TOPEET_DEMO_UNINSTALL_TARGET_CMDS
        $(TARGET_MAKE_ENV) $(MAKE) -C $(@D) uninstall
endef
```

```
$(eval $(generic-package))
endif
```

创建源码目录,上文的 Makefile 文件里已经指定了源码目录 external/topeet_demo/src。

```
cd /home/topeet/Linux/rk356x_linux/buildroot/package/rockchip/
mkdir -p external/topeet_demo/src
```

编写源码 topeet_demo.c，在 topeet_demo/src/ 下添加 topeet_demo.c：

topeet_demo.c 内容如下所示：

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
        printf("Hello World!\n");
        return 0;
}
```

编写 Makefile，在 topeet_demo/src/ 下添加 Makefile,Makefile 内容如下所示：

```
DEPS =
OBJ = topeet_demo.o
CFLAGS =
%.o: %.c $(DEPS)
        $(CC) -c -o $@ $< $(CFLAGS)

topeet_demo: $(OBJ)
        $(CXX) -o $@ $^ $(CFLAGS)

.PHONY: clean
clean:
        rm -f *.o *~ topeet_demo

.PHONY: install
```

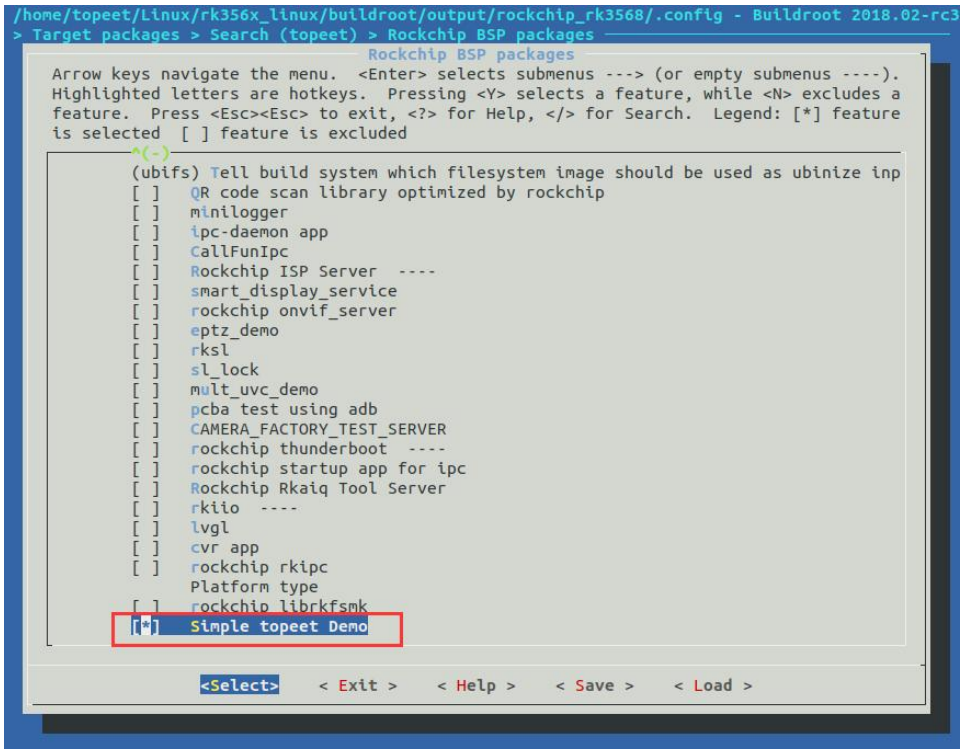
```
install:
        cp -f topeet_demo $(TARGET_DIR)/usr/bin/


.PHONY: uninstall
uninstall:
        rm -f $(TARGET_DIR)/usr/bin/topeet_demo
```

修改上一级 Config.in，在 buildroot/package/rockchip/Config.in 末尾添加一行：

```
source "package/rockchip/topeet_demo/Config.in"
```

```
source "package/rockchip/rkaiq_tool_server/Config.in"
source "package/rockchip/rkiio/Config.in"
source "package/rockchip/lvgl/Config.in"
source "package/rockchip/cvr_app/Config.in"
source "package/rockchip/rkipc/Config.in"
source "package/rockchip/rkfsmk/Config.in"
source "package/rockchip/topeet_demo/Config.in"
endif
```

然后参考 1.12.2 小节，打开 menuconfig 图形化配置，如下图所示，找到 topeet_demo 并选中配置。



输入以下命令编译 topeet_demo

```
make topeet_demo
```

输入以下命令打包进根文件系统

```
make
```

若修改源码，重新编译软件包，输入以下命令：

```
make topeet_demo-rebuild
```

1.12.6 buildroot overlay 功能

rootfs-overly 是一个很不错的功能，如果我们想要添加或者修改文件系统的某个文件，我们可以使用 overlay 功能，overlay 功能会在编译完成之后将指定的文件覆盖到文件系统的指定目录。

比如说我们要在根文件系统的/etc/目录下添加文件 topeet-test，操作步骤如下所示：

对于 iTOP-RK3588 来说，默认已经添加了 overlay 根目录，如下所示：

```
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$ ls board/rockchip/rk3588/fs-overlay/
8723du.ko  etc  quectel-CM  rockchip-test  rtk_btusb.ko  usr
topeet@ubuntu:~/Linux/3588-linux-full/buildroot$
```

添加文件到覆盖目录，在 fs-overlay 目录下新建/etc/topeet-test 文件，然后重新编译 buildroot，烧写编译好的镜像，启动 iTOP-RK3588 之后，可以看到已添加的文件 etc/topeet-test。