

# NFS启动Android用户指南

---

文件标识: RK-YH-YF-294

发布版本: V1.4.0

日期: 2023-06-27

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

## 免责声明

本文档按“现状”提供, 瑞芯微电子股份有限公司(“本公司”, 下同)不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因, 本文档将可能在未经任何通知的情况下, 不定期进行更新或修改。

## 商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标, 归本公司所有。

本文档可能提及的其他所有注册商标或商标, 由其各自所有者所有。

## 版权所有 © 2020 瑞芯微电子股份有限公司

超越合理使用范畴, 非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: [www.rock-chips.com](http://www.rock-chips.com)

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: [fae@rock-chips.com](mailto:fae@rock-chips.com)

前言

概述

本文档描述了Android设备通过网络从NFS服务器启动Android的实现方法。

产品版本

芯片名称	SDK版本
所有芯片通用	Android10/11/12/13/14 + Linux4.19/5.10

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	陈美友	2021-06-21	初始版本，支持Android10
V1.0.1	陈美友	2021-06-22	细微调整
v1.1.0	陈美友	2021-06-25	支持Android11
v1.2.0	陈美友	2021-08-12	支持Android12
v1.3.0	陈美友	2022-12-06	支持Android13
v1.4.0	陈美友	2023-06-27	支持Android14

# 目录

## NFS启动Android用户指南

1. 开发环境描述
2. Android SDK修改
  - 2.1 u-boot部分
  - 2.2 kernel部分
  - 2.3 Android部分
3. NFS Server部署
  - 3.1 PC上安装NFS Server
  - 3.2 rootfs
  - 3.3 Images制作
4. NFS Client部署
  - 4.1 分区更改
  - 4.2 固件烧写
  - 4.3 NFS启动

# 1. 开发环境描述

---

- Android设备: RK3399 IND EVB板
- Android SDK版本: 10/11/12/13/14
- PC端NFS服务器: ubuntu 20.04
- 设备端IP: 使用DHCP动态获取
- 服务器IP: 192.168.31.148

## 2. Android SDK修改

---

### 2.1 u-boot部分

主要是要支持:

1. 保存u-boot env到设备存储中

若你使用的并非rk3399芯片, 请参考补丁内容去修改目标芯片的配置

```
1 | cd $SDK/u-boot
2 | git am /path/to/u-boot/0003-Enable-save-env-to-block-device.patch
```

2. 支持把boot.img放到nfs服务器上而不是放在设备存储中

注意:

- Android12的u-boot已经有支持该功能, 不需再打下面的补丁
- 开机时候, u-boot需要从boot分区读取dtb信息, 因此boot分区依然需要烧写boot.img

```
1 | cd $SDK/u-boot
2 | git am /path/to/u-boot/0001-rockchip-rockchip-allow-update-resource-list-before-.patch
3 | git am /path/to/u-boot/0002-rockchip-board-reload-kernel-dtb-from-bootm-image.patch
```

修改完成后编译u-boot:

```
1 ./make.sh rk3399
2
3 也可以直接调用build.sh脚本编译uboot
4 ./build.sh -U
```

## 2.2 kernel部分

这部分主要是添加以下配置以支持nfs client功能:

```
1 CONFIG_NFS_FS=y
2 CONFIG_NFS_V3_ACL=y
3 CONFIG_NFS_V4=y
4 CONFIG_NFS_SWAP=y
5 CONFIG_NFS_V4_1=y
6 CONFIG_NFS_V4_2=y
7 CONFIG_NFS_V4_1_MIGRATION=y
8 CONFIG_ROOT_NFS=y
```

- Android-10

补丁如下:

```
1 cd $SDK/kernel/config
2 patch -p1 < /path/to/android10/kernel/0001-android-10-support-nfs-
  client.patch
```

修改完成后, 编译生成新的不含rootfs的boot.img

```
1 make ARCH=arm64 rockchip_defconfig android-10.config
2 make ARCH=arm64 rk3399-evb-ind-lpddr4-android-avb.img -j12
```

- Android-11

补丁如下:

```
1 cd $SDK/kernel/config
2 patch -p1 < /path/to/android11/kernel/0001-android-11-support-nfs-
  client.patch
```

修改完成后, 编译生成新的不含rootfs的boot.img

```
1 make ARCH=arm64 rockchip_defconfig android-11.config
2 make ARCH=arm64 rk3399-evb-ind-lpddr4-android-avb.img -j12
```

- Android-12

补丁如下:

```

1 | cd $SDK/mkcombinedroot
2 | patch -p1 < /path/to/android12/kernel/0001-configs-support-nfs-
  | client.patch
3 |
4 | cd $SDK/kernel/config
5 | patch -p1 < /path/to/android12/kernel/0001-support-nfs-fs.patch

```

修改完成后，编译生成新的不含rootfs的boot.img

```

1 | make ARCH=arm64 rockchip_defconfig android-11.config
2 | make ARCH=arm64 rk3399-evb-ind-lpddr4-android-avb.img -j12

```

- Android-13

补丁如下：

```

1 | cd $SDK/mkcombinedroot
2 | patch -p1 < /path/to/android13/kernel/0001-configs-support-nfs-
  | client.patch
3 |
4 | cd $SDK/kernel/config
5 | patch -p1 < /path/to/android13/kernel/0001-support-nfs-file-system.patch

```

修改完成后，编译生成新的不含rootfs的boot.img

```

1 | make ARCH=arm64 rockchip_defconfig rk356x.config android-13.config
2 | make ARCH=arm64 rk3568-evb1-ddr4-v10.img -j12
3 |
4 | 也可以直接调用build.sh脚本完成编译
5 | ./build.sh -C

```

- Android-14

补丁如下：

```

1 | cd $SDK/mkcombinedroot
2 | patch -p1 < /path/to/android14/kernel/0001-configs-support-nfs-
  | client.patch
3 |
4 | cd $SDK/kernel/config
5 | patch -p1 < /path/to/android14/kernel/0001-support-nfs-file-system.patch

```

修改完成后，编译生成新的不含rootfs的boot.img

```

1 | 直接调用build.sh脚本编译kernel
2 | ./build.sh -C

```

## 2.3 Android部分

- init

修改init以支持nfs启动

- netd

避免android启动过程中对ethernet初始化导致nfs通信异常

补丁如下:

- Android-10:

```
1 cd $SDK/system/core
2 git am /path/to/android10/android/0001-init-Support-boot-up-from-NFS.patch
3 git am /path/to/android10/android/0002-fixed-fstab-for-nfs-boot-up.patch
4
5 cd $SDK/system/netd
6 git am /path/to/android10/android/0001-Skip-clear-interface-for-NFS-boot-up.patch
```

- Android-11:

```
1 cd $SDK/system/core
2 git am /path/to/android11/android/0001-init-support-boot-up-from-nfs.patch
3
4 cd $SDK/system/netd
5 git am /path/to/android11/android/0001-Skip-clear-interface-for-NFS-boot-up.patch
```

- Android-12:

```
1 cd $SDK/system/core
2 git am /path/to/android12/android/0001-init-support-bootup-from-nfs.patch
3
4 cd $SDK/system/netd
5 git am /path/to/android12/android/0001-skip-clear-interface-and-route-controller-init-for-N
```

- Android-13:

```
1 cd $SDK/system/core
2 git am /path/to/android13/android/0001-init-support-bootup-from-nfs.patch
3
4 cd $SDK/system/netd
5 git am /path/to/android13/android/0001-skip-clear-interface-and-route-controller-init-for-N
```

- Android-14:

```
1 cd $SDK/system/core
2 git am /path/to/android14/android/0001-init-support-bootup-from-nfs.patch
3
4 cd $SDK/system/netd
5 git am /path/to/android14/android/0001-skip-clear-interface-and-route-
  controller-init-for-N
```

修改完成后，使用 `make -j16` 进行编译，生成新的固件

### 3. NFS Server部署

下面关于nfs server以及nfs client的配置，基于以下这些策略：

1. 每个设备的nfsroot文件夹独立

每个设备都使用各自的nfsroot文件夹，文件夹以设备的编号命名，比如设备编号是 `76`，则nfsroot文件夹是 `76`

2. nfsroot文件夹中存放ramdisk和各个image，目录结构如下：

```
1  nfsroot/76/
2  ├── apex
3  ├── debug_ramdisk
4  ├── dev
5  ├── fstab.rk30board
6  ├── images
7  |   ├── boot.img
8  |   ├── cache.img
9  |   ├── metadata.img
10 |   ├── super.raw.img
11 |   └── userdata.img
12 ├── init
13 ├── metadata
14 ├── mnt
15 ├── proc
16 ├── sys
17 └── system
```

以下配置以设备编号 `76` 为例进行说明



## 3.1 PC上安装NFS Server

- 安装nfs server

```
1 | sudo apt-get install -y nfs-kernel-server
```

- 配置nfs共享文件夹

创建NFS共享文件夹

```
1 | NFSROOT=/home/cmy/temp/nfs/nfsroot/76
2 | mkdir -p $NFSROOT
```

修改 /etc/exports

```
1 | /home/cmy/temp/nfs/nfsroot/76 192.168.31.*
   | (rw,sync,no_root_squash,no_subtree_check)
```

重启nfs service

```
1 | sudo systemctl restart nfs-kernel-server
```

查看nfs共享目录

```
1 | showmount -e
```

## 3.2 rootfs

rootfs使用Android编译生成的ramdisk

```
1 | # 这里你需要根据自己实际的android sdk版本选择对应的ramdisk
2 | cp -a $SDK/out/target/product/rk3399_Android10/ramdisk/* $NFSROOT/
3 | mkdir $NFSROOT/images
```

`$NFSROOT/images` 目录下存放所需要的固件，如下所述

## 3.3 Images制作

先进入rootfs的images目录下，再开始制作image

```
1 | cd $NFSROOT/images
```

- super.raw.img

Android编译生成的super.img，需要使用 `simg2img` 转换为raw image格式

```
1 sudo apt install simg2img
2
3 # 这里你需要根据自己实际的android sdk版本选择对应的super.img
4 simg2img $SDK/out/target/product/rk3399_Android10/super.img
   ./super.raw.img
```

- boot.img

如果你是从设备分区中加载kernel，则不需要在nfs服务器端放置boot.img。

如果你是从nfs服务器加载kernel，则需要放置一个boot.img在nfs服务器端。

boot.img需要包含kernel和resource，但不要包含android's rootfs，可以使用kernel目录下编译生成的boot.img

```
1 cp $SDK/kernel/boot.img ./
```

**注意：**如果你使用了带有rootfs的boot.img，那么设备开机时候不会从nfs挂载rootfs，而是使用boot.img自带的rootfs，从而导致启动失败；你应该使用不含rootfs的boot.img

正常从NFS挂载rootfs会有如下这句LOG：

```
1 [ 4.129940] VFS: Mounted root (nfs4 filesystem) on device 0:19.
```

- metadata.img/userdata.img/cache.img

```
1 dd if=/dev/zero of=metadata.img bs=1M count=16
2 dd if=/dev/zero of=userdata.img bs=1M count=4096
3 dd if=/dev/zero of=cache.img bs=1M count=384
4 mkfs.ext4 cache.img
```

1. 此处创建的userdata是4GB大小，若不够用，可再增大些。

2. 对于android12，在系统启动过程中不会自动对cache分区进行格式化，导致我们想要 `remount` 时候，执行 `adb remount` 将会失败；所以在创建cache.img时候需将其格式化为ext4格式

## 4. NFS Client部署

### 4.1 分区更改

因多个分区已放在NFS Server上，设备中的分区需要进行裁剪，修改parameter.txt文件去掉以下分区：

```
1 userdata
2 super
3 metadata
4 cache
5 backup
6 recovery
```

其中：

- userdata/super/metadata/cache所对应的image都已经放在nfs server上面，不必再烧写到设备中；
- boot分区是个例外，因为u-boot启动过程中需要用到resource信息，因而boot.img目前要烧写到设备分区当中，以供u-boot所用；但如果你希望设备开机时候从服务器上加载kernel，则还需要把boot.img拷贝一份到 `$NFSROOT/images` 目录下。
- 而NFS启动方式下不支持OTA升级和恢复出厂设置等功能，因而去掉backup/recovery

修改后的parameter.txt可参考补丁包里面的parameter.txt

## 4.2 固件烧写

设备端需要烧写的images：

```
1 MiniLoaderAll.bin
2 parameter.txt
3 uboot.img
4 trust.img
5 dtbo.img
6 vbmeta.img
7 boot.img
```

对于Android 11/12/13/14版本，还需要烧写 `baseparameter.img`

## 4.3 NFS启动

在u-boot启动阶段按 `Ctrl+C` 键可进入命令行模式（需要连接调试串口），然后输入以下命令设置参数并保存到设备存储中，最后重启设备即可，此后不必再输入这些命令。

**注意：**在调试阶段，建议不要保存参数到存储；等到调试成功以后，再保存参数到存储中；为此，调试阶段可把 `saveenv` 命令更改为 `boot` 命令即可

```

1 setenv client_id 76
2 setenv boot_nfs 'setenv autoload no; dhcp; setenv serverip 192.168.31.148;
  setenv nfsroot $serverip:/home/cmy/temp/nfs/nfsroot/$client_id; setenv
  bootargs root=/dev/nfs nfsroot=$nfsroot,v4.2,tcp rw
  ip=$ipaddr:$serverip:$gatewayip:$netmask::eth0:off init=/init
  androidboot.selinux=permissive androidboot.hardware=rk30board
  androidboot.nfsboot=1'
3 setenv bootcmd 'run boot_nfs; ${bootcmd}'
4 saveenv

```

如果你想要uboot从服务器端加载kernel，而不是从设备存储中加载kernel，需要对前面的命令修改如下：

```

1 setenv client_id 76
2 setenv boot_nfs 'setenv autoload no; dhcp; setenv serverip 192.168.31.148;
  setenv nfsroot $serverip:/home/cmy/temp/nfs/nfsroot/$client_id; setenv
  bootargs root=/dev/nfs nfsroot=$nfsroot,v4.2,tcp rw
  ip=$ipaddr:$serverip:$gatewayip:$netmask::eth0:off init=/init
  androidboot.selinux=permissive androidboot.hardware=rk30board
  androidboot.nfsboot=1; setenv loadaddr 0x10000000; nfs $loadaddr
  ${nfsroot}/images/boot.img; bootm $loadaddr'
3 setenv bootcmd 'run boot_nfs;'
4 saveenv

```

上述命令中的nfsroot需要根据你的实际情况进行修改。

对于Android 13/14版本，还需要在 `bootargs` 环境变量中添加 `androidboot.boot_devices` 属性，比如芯片是rk3568，该属性值是：

```

1 androidboot.boot_devices=fe310000.sdhci,fe330000.nandc

```

具体芯片的这个属性值，可以在Android SDK中通过如下命令获得：

```

1 source build/envsetup.sh
2 lunch
3 get_build_var BOARD_KERNEL_CMDLINE | grep -o "androidboot.boot_devices=
  [[:graph:]]*"

```

所以，对于rk3568 android13平台，追加 `androidboot.boot_devices` 属性后，最终的u-boot命令如下：

```

1 setenv client_id 76
2 setenv boot_nfs 'setenv autoload no; dhcp; setenv serverip 192.168.31.148;
  setenv nfsroot $serverip:/home/cmy/temp/nfs/nfsroot/$client_id; setenv
  bootargs root=/dev/nfs nfsroot=$nfsroot,v4.2,tcp rw
  ip=$ipaddr:$serverip:$gatewayip:$netmask::eth0:off init=/init
  androidboot.selinux=permissive androidboot.hardware=rk30board
  androidboot.nfsboot=1 androidboot.boot_devices=fe310000.sdhci,fe330000.nandc'
3 setenv bootcmd 'run boot_nfs; ${bootcmd}'
4 saveenv

```