用UEFI启动一台RISC-V机器

汪 流 | openEuler RISC-V SIG Committer

开放原子开源基金会 OPENATOM FOUNDATION | OpenEuler

- UEFI简介
- UEFI启动流程
- UEFI系统组成
- EDK2简介
- 在开发板用UEFI启动操作系统
- THANKS

UEFI简介

UEFI简介

什么是UEFI?

UEFI，即统一扩展固件接口（Unified Extensible Firmware Interface），是UEFI论坛发布的一种用于操作系统启动和平台固件之间的接口标准，没有提供UEFI的实现，它的实现是由其他公司或开源组织提供，例如TianoCore社区提供的EDK2。UEFI提供了一种更现代、灵活和功能丰富的固件接口，具有以下特点：

- 灵活性： UEFI支持更多的硬件架构和设备，使其适用于各种不同类型的计算机系统，包括传统PC、服务器、移动设备等。
- 图形界面： UEFI固件可以支持图形用户界面(GUI)，使用户能够更直观地与系统进行交互，例如在启动时选择引导设备或配置系统设置。
- 安全性： UEFI提供了一些安全功能，如安全启动(Secure Boot)，可以帮助防止恶意软件在系统启动过程中被加载。
- 扩展性： UEFI具有可扩展性，允许厂商和开发者开发并添加自定义的固件功能，以满足特定需求或支持新的技术。

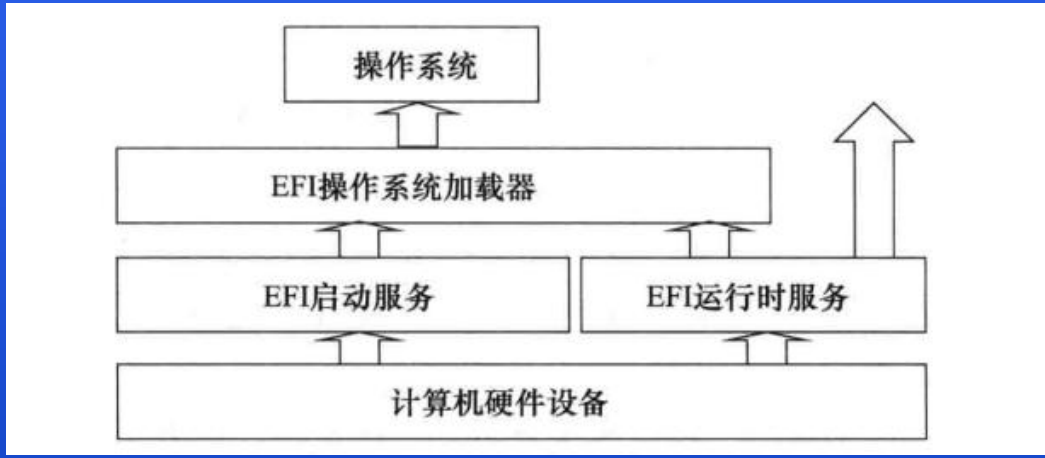
UEFI系统组成

UEFI系统组成

UEFI提供给操作系统的接口包括启动服务(Boot Services，BS)和运行时服务(Runtime Service，RT)以及隐藏在BS之后的丰富的接口(Protocol)。BS和RT以表的形式(C语言中的结构体)存在。UEFI驱动和服务以接口的形式通过BS提供给操作系统。

从操作系统加载器(OS Loader)被加载，到OS Loader执行ExitBootServices()的这段时间，是从UEFI环境向操作系统过渡的过程。在这个过程中，OS Loader可以通过BS和RT使用UEFI提供的服务，将计算机系统资源逐渐转移到自己手中，这个过程称为TSL(Transient System Load)。

当OS Loader完全掌握了计算机系统资源时，BS也就完成了它的使命。OS Loader调用ExitBootServices()结束BS并回收BS占用的资源，之后计算机系统进入UEFI Runtime阶段。在Runtime阶段只有运行时服务继续为OS提供服务，BS已经从计算机系统中销毁。

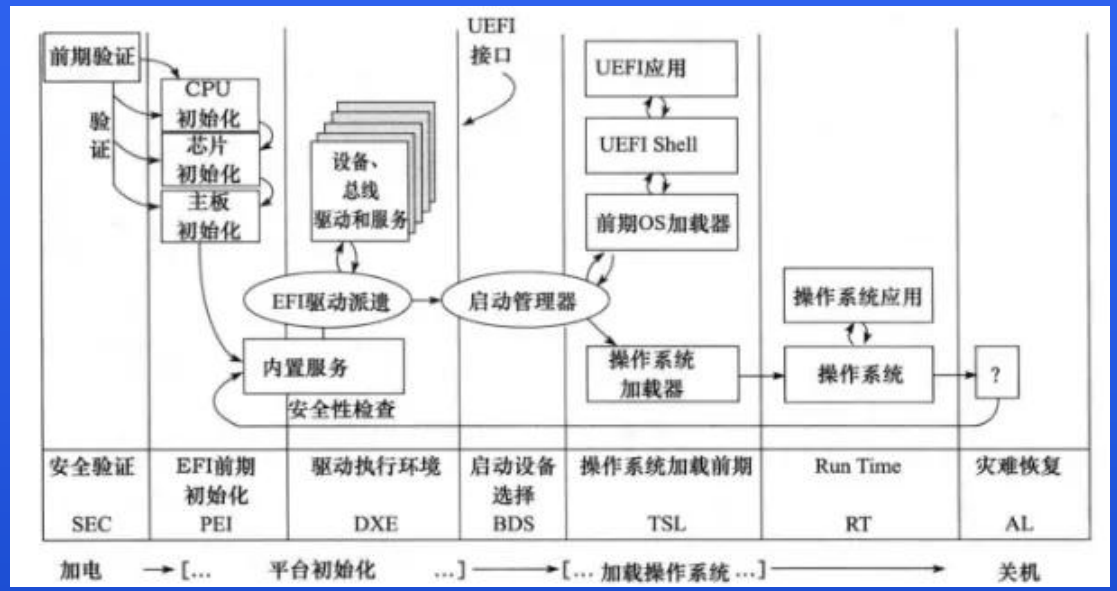


UEFI启动流程

UEFI启动流程

启动流程简介

UEFI系统的启动遵循UEFI平台初始化标准。UEFI 系统从加电到关机可分为7个阶段：
SEC（安全验证） ---> PEI（EFI前期初始化） ---> DXE（驱动执行环境）
---> BDS（启动设备选择） ---> TSL（操作系统加载前期）
---> RT（Run Time）
---> AL（灾难恢复）



前三个阶段（SEC、PEI、DXE）是UEFI初始化阶段，DXE阶段加载了大量的硬件驱动，DXE阶段结束后UEFI环境准备完毕。
后面四个阶段（BDS、TSL、RT、AL）属于UEFI规范，BDS阶段负责选择启动设备，TSL阶段主要负责运行操作系统加载器，这是控制权限向操作系统转移的一个阶段。
在TSL阶段中，当操作系统加载器调用ExitBootServices()服务，回收启动服务的资源。进入RT阶段。此时，操作系统已经被加载和运行，仅有运行时服务可以被访问。
最后，当系统硬件或操作系统出现严重错误不能继续运行时，固件会尝试修复错误，这时系统进入AL（After Life）阶段。

EDK2简介

EDK2简介

什么是EDK2?

EDK2 是一套用于开发和定制统一可扩展固件接口（UEFI）固件的开源开发工具。是UEFI规范的实现之一。它提供了一系列工具和库，帮助开发人员构建符合UEFI标准的固件，用于引导和管理计算机系统。具有以下特点：

- 开源性：EDK2是一个开源项目，这意味着开发者可以自由地访问和使用其源代码。这种开放性促进了代码的透明性和可定制性，使得开发者能够根据自己的需求进行修改和优化。
- 模块化设计：EDK2采用了模块化的设计方式，这使得代码更加清晰、易于维护，并且方便扩展。开发者可以根据自己的需要添加或删除模块，以满足特定的功能需求。
- 跨平台兼容性：EDK2旨在支持多种处理器架构，如x86、x64、ARM、RISC-V等，这使得它能够在不同的硬件平台上运行，为开发者提供了广泛的兼容性。
- UEFI规范支持：EDK2完全遵循UEFI规范，提供了丰富的UEFI服务和协议实现，这使得开发者能够轻松地构建符合UEFI标准的固件。
- 可扩展性：由于其模块化设计，EDK2允许开发者根据需要添加新的功能或修改现有功能，从而满足不断变化的市场需求和技术要求。。

在开发板用UEFI启动操作系统

在开发板用UEFI启动操作系统

SG2042 UEFI启动流程

ZSBL ---> FSBL ---> OpenSBI ---> EDK2 ---> GRUB2 ---> OS

ZSBL阶段：芯片初始化的第一阶段。主要工作是初始化DDR。

FSBL阶段：为ZSBL阶段DEBUG的串口初始化。建立内存映射表，用于初始化和配置操作系统或固件的内存管理。一些CPU功能的初始化。

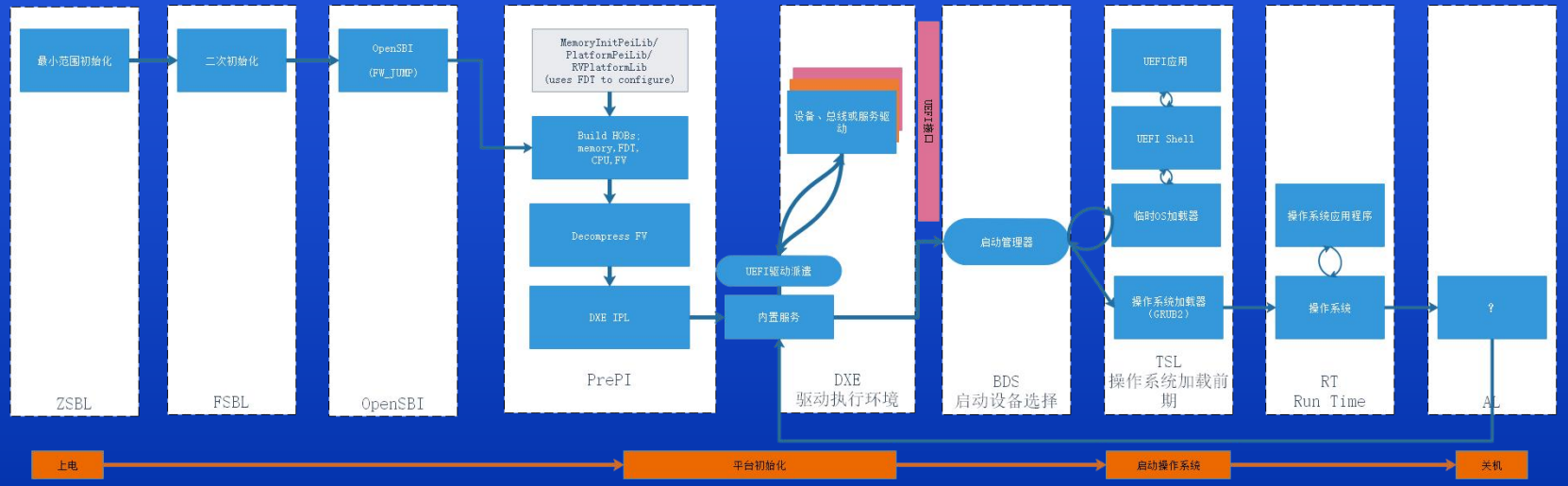
OpenSBI阶段：提供了SG2042的平台初始化条件，作为独立固件在edk2之前的初始化链接环境运行，以FW_PAYLOAD的形式存在，以M模式初始化系统，初始化SG2042的部分功能，并将后续EDK2运行环境置于S模式。

PrePI阶段：在内存、CPU、FV、堆栈和处理器smbios信息上进行一些处理，最终加载DXE核心并转移控制权。

DXE阶段：EDK2 DXE OpenSBI接口为所有DXE驱动程序调用SBI服务提供了统一的接口，DXE阶段主要增加了SG2042 SD卡读取驱动程序来读取SD卡的内容。

BDS阶段：主要任务是选择并加载操作系统的引导加载程序，启动操作系统。在S模式下运行，因为OpenSBI将模式转换为S模式，当BDS将权限交给操作系统时，必须是S模式，避免了另一种模式切换。

TSL阶段：使用edk2 Shell引导和启动SD卡中的GRUB2程序代码，使用GRUB加载分区中不同的OS操作系统。

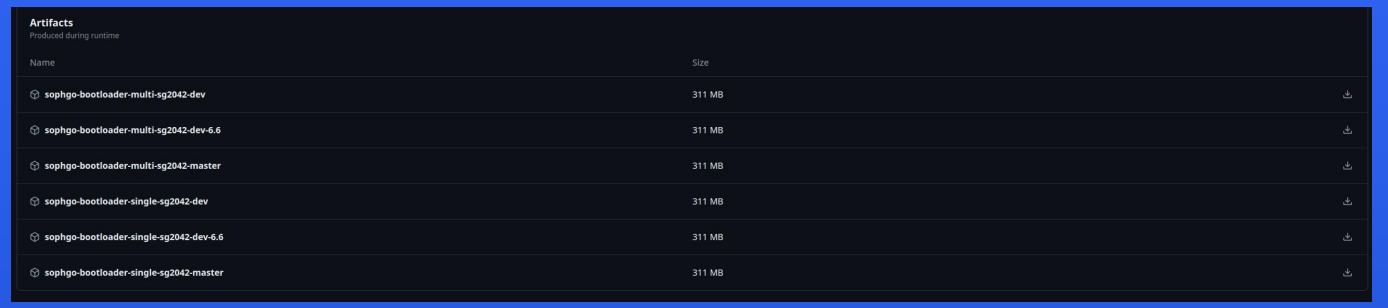


在开发板用UEFI启动操作系统

准备启动固件SD卡

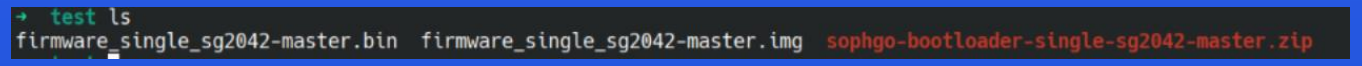
从算能的Github仓库里获取固件压缩包 sophgo-bootloader-single-sg2042-master.zip 并解压，会获得bin和img文件

```
# unzip sophgo-bootloader-single-sg2042-master.zip
```



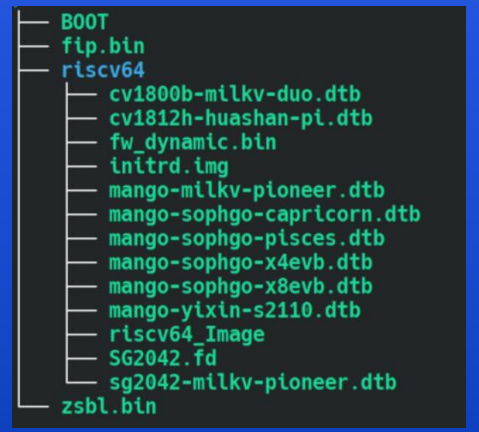
将img文件烧录至SD卡上

```
# dd if=firmware_single_sg2042-master.img   of=/dev/sda bs=512K
iflag=fullblock oflag=direct conv=fsync status=progress
```



修改文件，将EDK2编译的产物SG2042.fd文件替换riscv64_Image文件，切换为UEFI启动

```
# mount /dev/sda1  /mnt/
# pushd /mnt/riscv64/
# mv riscv64/riscv64_Image riscv64/riscv64_Image.backup
# mv riscv64/SG2042.fd riscv64/riscv64_Image
# popd
# umount /mnt
```



在开发板用UEFI启动操作系统

创建镜像文件

创建16G的openEuler-sg2042.img文件，并对其进行分区和格式化

```
# dd if=/dev/zero of=openEuler-sg2042.img bs=1M count=16384
# loopdev=$(losetup -fP --show openEuler-sg2042.img)
# sgdisk -n 1:0:+512M -n 2:0:+512M -n 3:0:0  -t 1:ef00 -t 2:ea00 -t 3:8300  -c 1:efi -c 2:boot -c 3:oERV ${loopdev}
# mkfs.fat -F 32 ${loopdev}p1
# mkfs.fat -F 32 ${loopdev}p2
# mkfs.ext4 ${loopdev}p3
```

```
[root@SnailArch test]# sgdisk -n 1:0:+512M -n 2:0:+512M -n 3:0:0  -t 1:ef00 -t 2:ea00 -t 3:8300  -c 1:efi -c 2:boot -c 3:oERV ${loopdev}
Creating new GPT entries in memory.
The operation has completed successfully.
[root@SnailArch test]# sgdisk -p ${loopdev}
Disk /dev/loop0: 33554432 sectors, 16.0 GiB
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 2D1A7424-F807-4068-BBCE-BE509EC06D8A
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 33554398
Partitions will be aligned on 2048-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         1050623   512.0 MiB   EF00  efi
   2         1050624         2099199   512.0 MiB   EA00  boot
   3         2099200        33554398   15.0 GiB    8300  oERV
[root@SnailArch test]# mkfs.fat -F 32 ${loopdev}p1
mkfs.fat 4.2 (2021-01-31)
[root@SnailArch test]# mkfs.fat -F 32 ${loopdev}p2
mkfs.fat 4.2 (2021-01-31)
[root@SnailArch test]# mkfs.ext4 ${loopdev}p3
mke2fs 1.47.0 (5-Feb-2023)
丢弃设备块：完成
创建含有 3931899 个块（每块 4k）和 983040 个 inode 的文件系统
文件系统 UUID：6513303a-082c-4093-bd74-04a0b574514f
超级块的备份存储于下列块：
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

正在分配组表：完成
正在写入 inode表：完成
创建日志（16384 个块）：完成
写入超级块和文件系统账户统计信息：已完成
```

在开发板用UEFI启动操作系统

rootfs与挂载

获取openEuler RISC-V的rootfs压缩包

wget https://mirror.iscas.ac.cn/openeuler-sig-riscv/openEuler-RISC-V/testing/2403LTS-test/v1/openeuler-rootfs.tar.gz

将rootfs压缩包解压至镜像的root分区，并挂载其他分区

mount ${loopdev}p3 /mnt
tar xf openeuler-rootfs.tar.gz -C /mnt
rm -fr /mnt/boot/*
mount ${loopdev}p2 /mnt/boot
mkdir /mnt/boot/efi
mount ${loopdev}p1 /mnt/boot/efi
mount -o bind /dev/ /mnt/dev/
mount -t proc proc /mnt/proc
mount -t sysfs sysfs /mnt/sys/
mount -t devpts pts /mnt/dev/pts

```
[root@SnailArch test]# mount ${loopdev}p3 /mnt
[root@SnailArch test]#  tar xf openeuler-rootfs.tar.gz -C /mnt
[root@SnailArch test]#  rm -fr /mnt/boot/*
[root@SnailArch test]# mount ${loopdev}p2 /mnt/boot
[root@SnailArch test]# mkdir /mnt/boot/efi
[root@SnailArch test]# mount ${loopdev}p1 /mnt/boot/efi
[root@SnailArch test]#  mount -o bind /dev/ /mnt/dev/
[root@SnailArch test]# mount -t proc proc  /mnt/proc
[root@SnailArch test]# mount -t sysfs sysfs  /mnt/sys/
[root@SnailArch test]# mount -t devpts pts /mnt/dev/pts
[root@SnailArch test]# mount | grep mnt
/dev/loop0p3 on /mnt type ext4 (rw,relatime)
/dev/loop0p2 on /mnt/boot type vfat (rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro)
/dev/loop0p1 on /mnt/boot/efi type vfat (rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro)
dev on /mnt/dev type devtmpfs (rw,nosuid,relatime,size=8012268k,nr_inodes=2003067,mode=755,inode64)
proc on /mnt/proc type proc (rw,relatime)
sysfs on /mnt/sys type sysfs (rw,relatime)
pts on /mnt/dev/pts type devpts (rw,relatime,mode=600,ptmxmode=000)
[root@SnailArch test]#
```

在开发板用UEFI启动操作系统

安装软件包

安装适配sg2042设备的kernel和grub相关软件包

```
# chroot /mnt
# export PATH=$PATH:/usr/sbin/
# yum-config-manager --add-repo https://build-repo.tarsier-infra.isrc.ac.cn/Factory:/Board:/SG2042/openEuler_24.03_mainline_riscv64/
# echo "priority=1" >> /etc/yum.repos.d/build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_.repo
# yum makecache
# yum install  --nogpgcheck  kernel-6.6.0-19.0.0.21.oe2403 kernel-headers-6.6.0-19.0.0.21.oe2403.riscv64 linux-firmware grub2-efi-riscv64  efibootmgr grub2-efi-riscv64-modules
```

```
[root@SnailArch /]# yum-config-manager --add-repo https://build-repo.tarsier-infra.isrc.ac.cn/Factory:/Board:/SG2042/openEuler_24.03_mainline_riscv64/
添加仓库自：https://build-repo.tarsier-infra.isrc.ac.cn/Factory:/Board:/SG2042/openEuler_24.03_mainline_riscv64/
[root@SnailArch /]# echo "priority=1" >> /etc/yum.repos.d/build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_.repo
[root@SnailArch /]# yum makecache
created by dnf config-manager from https://build-repo.tarsier-infra.isrc.ac.cn/Factory:/Board:/SG2042/openEuler_24.03_mainline_riscv64/                3.7 kB/s
OS                                                                                                                                                  5.3 kB/s
EPOL                                                                                                                                                5.7 kB/s
extra                                                                                                                                               5.5 kB/s
update                                                                                                                                              4.0 kB/s
everything                                                                                                                                          5.2 kB/s
debuginfo                                                                                                                                           4.3 kB/s
Metadata cache created.
[root@SnailArch /]# yum install  --nogpgcheck  kernel-6.6.0-19.0.0.21.oe2403 kernel-headers-6.6.0-19.0.0.21.oe2403.riscv64 linux-firmware grub2-efi-riscv64  efibootmgr grub2-efi-riscv64-modules
Last metadata expiration check: 0:01:49 ago on Fri 26 Apr 2024 11:48:06 PM CST.
Package linux-firmware-20231111-1.oe2403.noarch is already installed.
Dependencies resolved.
================================================================================================================================================================
 Package                                 Architecture            Version                                 Repository
================================================================================================================================================================
Installing:
 efibootmgr                              riscv64                 18-4.oe2403                             OS
 grub2-efi-riscv64                       riscv64                 1:2.12-9.oe2403                         build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_
 grub2-efi-riscv64-modules               noarch                  1:2.12-9.oe2403                         build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_
 kernel                                  riscv64                 6.6.0-19.0.0.21.oe2403                  build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_
 kernel-headers                          riscv64                 6.6.0-19.0.0.21.oe2403                  build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_
Installing dependencies:
 efi-filesystem                          noarch                  4-9.oe2403                              OS
 efivar-libs                             riscv64                 38-4.oe2403                             OS
 gettext                                 riscv64                 0.22-2.oe2403                           OS
 grub2-common                            noarch                  1:2.12-9.oe2403                         build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_
 grub2-tools                             riscv64                 1:2.12-9.oe2403                         build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_
 grub2-tools-extra                       riscv64                 1:2.12-9.oe2403                         build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_
 grub2-tools-minimal                     riscv64                 1:2.12-9.oe2403                         build-repo.tarsier-infra.isrc.ac.cn_Factory_Board_SG2042_openEuler_24.03_mainline_riscv64_
 os-prober                               riscv64                 1.81-1.oe2403                           OS

Transaction Summary
================================================================================================================================================================
Install  13 Packages

Total download size: 81 M
Installed size: 174 M
Is this ok [y/N]: y
Downloading Packages:
(1/13): grub2-common-2.12-9.oe2403.noarch.rpm                                                                                                       4.2 MB/s
(2/13): grub2-efi-riscv64-2.12-9.oe2403.riscv64.rpm                                                                                                 908 kB/s
(3/13): grub2-tools-2.12-9.oe2403.riscv64.rpm                                                                                                        11 MB/s
(4/13): grub2-tools-extra-2.12-9.oe2403.riscv64.rpm                                                                                                 7.6 MB/s
(5/13): grub2-tools-minimal-2.12-9.oe2403.riscv64.rpm                                                                                               6.0 MB/s
(6/13): grub2-efi-riscv64-modules-2.12-9.oe2403.noarch.rpm                                                                                          1.5 MB/s
(7/13): kernel-headers-6.6.0-19.0.0.21.oe2403.riscv64.rpm                                                                                           2.8 MB/s
(8/13): efi-filesystem-4-9.oe2403.noarch.rpm                                                                                                         12 kB/s
(9/13): efibootmgr-18-4.oe2403.riscv64.rpm                                                                                                          144 kB/s
```

在开发板用UEFI启动操作系统

配置grub

添加grub配置项，安装grub引导加载程序并生成grub配置文件

# vim /etc/default/grub
GRUB_DISABLE_LINUX_UUID=false
GRUB_DISABLE_LINUX_PARTUUID=false
GRUB_CMDLINE_LINUX="selinux=0 console=ttyS0,115200 earlycon"
# grub2-install --removable
# grub2-mkconfig -o /boot/grub2/grub.cfg

```
[root@SnailArch /]# grub2-install --removable
正在为 riscv64-efi 平台进行安装。
安装完成。没有报告错误。
[root@SnailArch /]# grub2-mkconfig -o /boot/grub2/grub.cfg

正在生成 grub 配置文件 ...
找到 Linux 镜像：/boot/vmlinuz-6.6.0-19.0.0.21.oe2403.riscv64
找到 initrd 镜像：/boot/initramfs-6.6.0-19.0.0.21.oe2403.riscv64.img
找到 Linux 镜像：/boot/vmlinuz-0-rescue-5b8956880ad041089d559bed8c534973
找到 initrd 镜像：/boot/initramfs-0-rescue-5b8956880ad041089d559bed8c534973.img
警告： os-prober 将不会运行并检测其它可引导分区。
这些分区上的系统将不会被添加至 GRUB 引导配置中。
请检查 GRUB_DISABLE_OS_PROBER 文档条目以了解详情。
正在添加 UEFI 固件设置的引导菜单项……
完成
```

在开发板用UEFI启动操作系统

更新fstab配置

获取UUID添加至fstab配置文件中

```
# efi_uuid=$(blkid -s UUID -o value /dev/loop0p1)
# boot_uuid=$(blkid -s UUID -o value /dev/loop0p2)
# root_uuid=$(blkid -s UUID -o value /dev/loop0p3)
# echo "UUID=$boot_uuid   /boot   vfat    defaults,noatime,x-systemd.device-timeout=300s,x-systemd.mount-timeout=300s 0 0" >> /etc/fstab
# echo "UUID=$efi_uuid   /boot/efi   vfat    defaults,noatime,x-systemd.device-timeout=300s,x-systemd.mount-timeout=300s 0 0" >> /etc/fstab
# echo "UUID=$root_uuid   /   ext4    defaults,noatime,x-systemd.device-timeout=300s,x-systemd.mount-timeout=300s 0 0" >> /etc/fstab
```

```
[root@SnailArch /]#  efi_uuid=$(blkid -s UUID -o value /dev/loop0p1)
[root@SnailArch /]# boot_uuid=$(blkid -s UUID -o value /dev/loop0p2)
[root@SnailArch /]#  root_uuid=$(blkid -s UUID -o value /dev/loop0p3)
[root@SnailArch /]# echo "UUID=$boot_uuid   /boot   vfat    defaults,noatime,x-systemd.device-timeout=300s,x-systemd.mount-timeout=300s 0 0" >> /etc/fstab
[root@SnailArch /]# echo "UUID=$efi_uuid   /boot/efi   vfat    defaults,noatime,x-systemd.device-timeout=300s,x-systemd.mount-timeout=300s 0 0" >> /etc/fstab
[root@SnailArch /]# echo "UUID=$root_uuid   /   ext4    defaults,noatime,x-systemd.device-timeout=300s,x-systemd.mount-timeout=300s 0 0" >> /etc/fstab
[root@SnailArch /]# cat /etc/fstab
UUID=8C66-2EC8   /boot   vfat    defaults,noatime,x-systemd.device-timeout=300s,x-systemd.mount-timeout=300s 0 0
UUID=8C0D-D236   /boot/efi   vfat    defaults,noatime,x-systemd.device-timeout=300s,x-systemd.mount-timeout=300s 0 0
UUID=6513303a-082c-4093-bd74-04a0b574514f   /   ext4    defaults,noatime,x-systemd.device-timeout=300s,x-systemd.mount-timeout=300s 0 0
[root@SnailArch /]#
```

在开发板用UEFI启动操作系统

系统盘完成

退出chroot环境，卸载之前配置的挂载

```
# exit
# umount -l /mnt/dev/pts/
# umount -l /mnt/dev
# umount /mnt/proc/
# umount /mnt/sys
# umount /mnt/boot/efi/
# umount /mnt/boot
# umount /mnt
# losetup -d ${loopdev}
```

烧录至SSD硬盘上

```
# sudo dd if=openEuler-sg2042.img  of=/dev/sda bs=512K iflag=fullblock oflag=direct conv=fsync status=progress
```

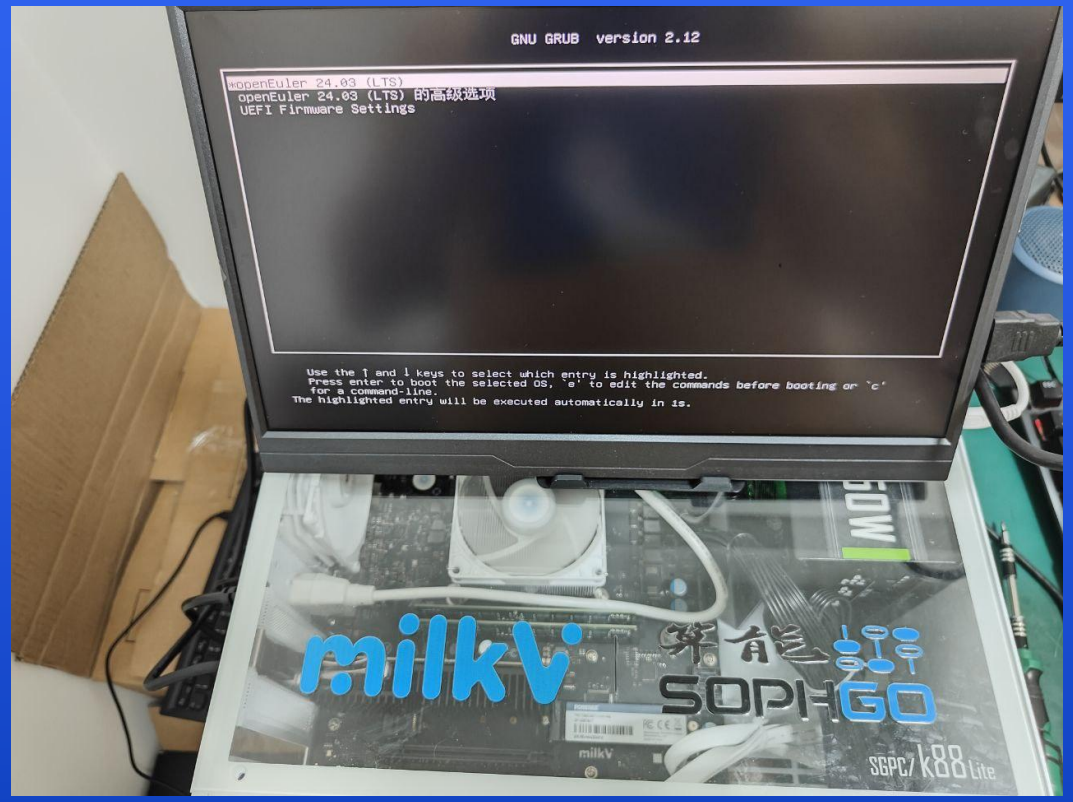
在开发板用UEFI启动操作系统

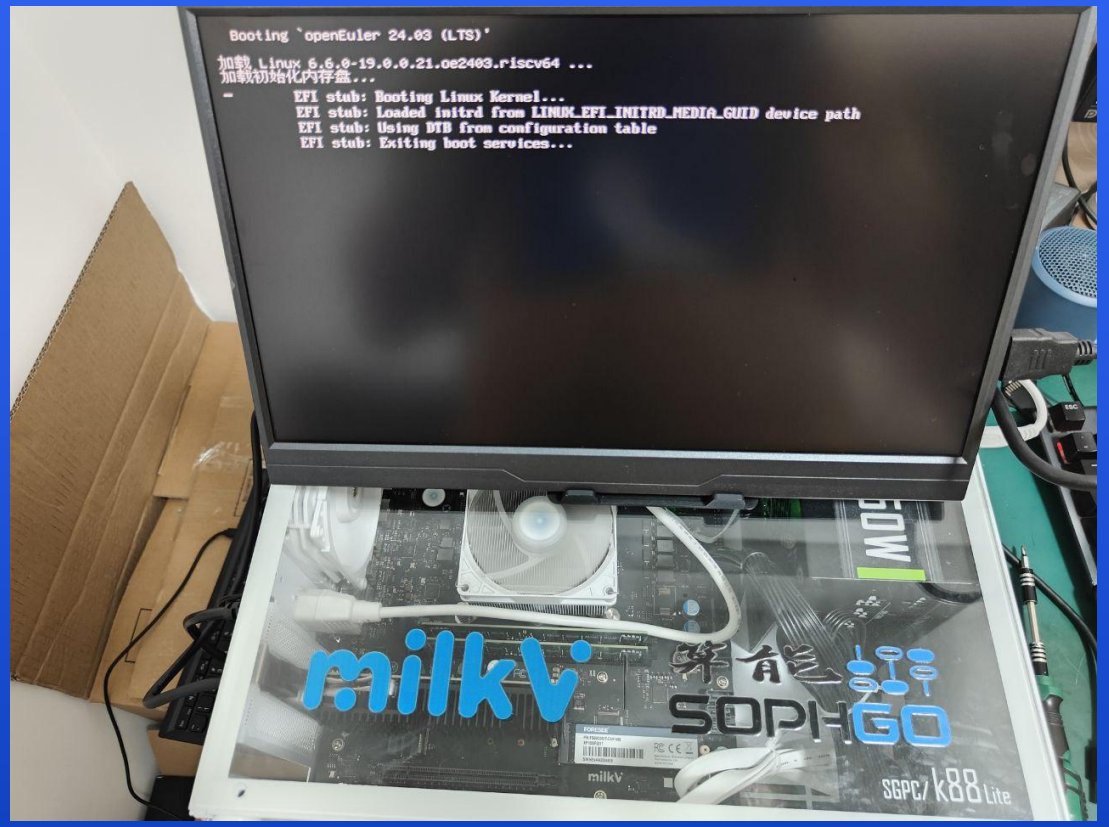
设备安装SD卡和SSD硬盘后，开机验证

```
SOPHGO ZSBL
sg2042:v0.3

sg2042 work in single socket mode
chip0 ddr info: raw data=0x5050505,
    ddr0 size:0x800000000
    ddr1 size:0x800000000
    ddr2 size:0x800000000
    ddr3 size:0x800000000
SD initializing 100000000Hz (transfer frequency at 25000000Hz)
sd card init ok
open 0:riscv64/conf.ini failed
conf.ini should start with "[sophgo-config]"
have no conf.ini file
rv boot from sd card
SD initializing 100000000Hz (transfer frequency at 25000000Hz)
sd card init ok
0:riscv64/fw_dynamic.bin file size is 270032
0:riscv64/riscv64_Image file size is 7995392
0:riscv64/initrd.img file size is 15773284
0:riscv64/mango-milkv-pioneer.dtb file size is 44575
sd read file ok
chip0 ddr node in dtb:
    base:0x0000000000, len:0xc0000000
    base:0x0100000000, len:0x70000000
    base:0x0800000000, len:0x800000000
    base:0x1000000000, len:0x800000000
    base:0x1800000000, len:0x800000000
use default mac address
main core sbi jump to 0x0, dynamic info:40019860

OpenSBI v1.2
Build time: 2024-03-28 02:30:20 +0000
Build compiler: gcc version 13.2.0 ()
  ____                    ____  ____ ___
 / __ \                  / ___||  _ \_ _|
| |  | |_ __   ___ _ __ | (___ | |_) | |
| |  | | '_ \ / _ \ '_ \ \___ \|  _ <| |
| |__| | |_) |  __/ | | |____) | |_) | |_
 \____/| .__/ \___|_| |_|_____/|___/|___|
       | |
       |_|

Platform Name             : Sophgo Mango
Platform Features         : medeleg
Platform HART Count       : 64
Platform IPI Device       : aclint-mswi
Platform Timer Device     : aclint-mtimer @ 50000000Hz
Platform Console Device   : uart8250
Platform HSM Device       : ---
Platform PMU Device       : ---
Platform Reboot Device    : mango-reset
Platform Shutdown Device  : mango-reset
Platform Suspend Device   : ---
Firmware Base             : 0x0
Firmware Size             : 1376 KB
Firmware RW Offset        : 0x40000
Runtime SBI Version       : 1.0

Domain0 Name              : root
Domain0 Boot HART         : 6
Domain0 HARTs             : 0*,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*
Domain0 Region00          : 0x0000007094000000-0x0000007094003fff M: (I,R,W) S/U: ()
Domain0 Region01          : 0x0000000000000000-0x0000000000003ffff M: (R,X) S/U: ()
Domain0 Region02          : 0x0000000000000000-0x0000000001ffffff M: (R,W) S/U: ()
Domain0 Region03          : 0x00000070ac000000-0x00000070ac3fffff M: (I,R,W) S/U: ()
Domain0 Region04          : 0x0000000000000000-0xffffffffffffffff M: (R,W,X) S/U: (R,W,X)
Domain0 Next Address      : 0x0000000002000000
Domain0 Next Arg1         : 0x000000004009fd90
Domain0 Next Mode         : S-mode
Domain0 SysReset          : yes
Domain0 SysSuspend        : yes

Boot HART ID              : 6
Boot HART Domain          : root
Boot HART Priv Version    : v1.11
Boot HART Base ISA        : rv64imafdcvx
Boot HART ISA Extensions  : time
Boot HART PMP Count       : 8
Boot HART PMP Granularity : 2048
Boot HART PMP Address Bits: 38
Boot HART MHPM Count      : 29
Boot HART MIDELEG         : 0x0000000000020222
Boot HART MEDELEG         : 0x000000000000b109
PROGRESS CODE: V03040003 I0
```

在开发板用UEFI启动操作系统

验证





在开发板用UEFI启动操作系统

验证

THANKS

开放原子开源基金会 OPENATOM FOUNDATION | OpenEuler