openEuler操作系统启动管理

课程介绍

- 本课程主要介绍了Linux操作系统启动的基本流程：
- 学完本课程后，您将能够：了解系统启动的流程；掌握设置系统从哪个内核启动的方法；掌握管理系统启动服务的基本方法

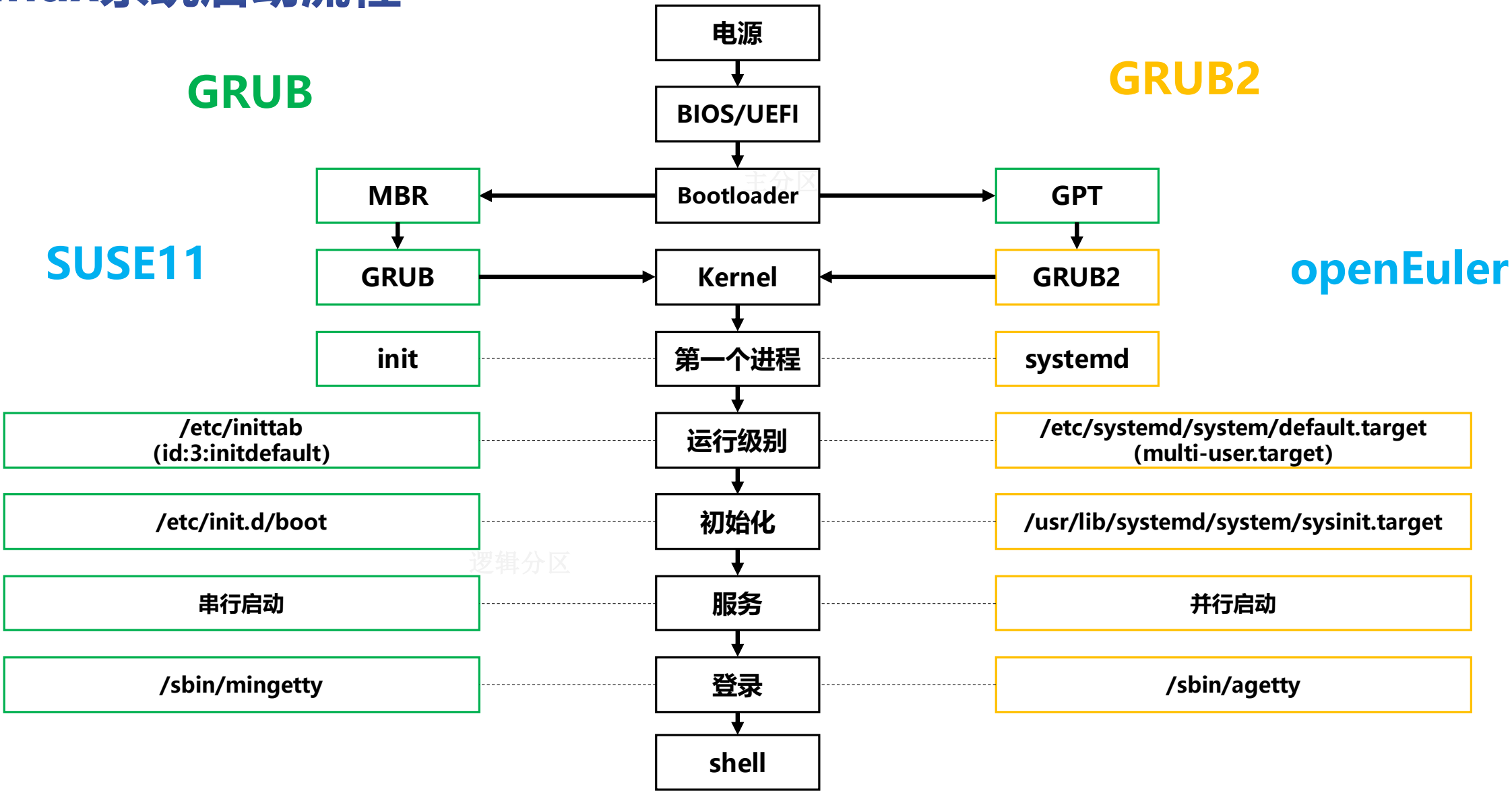
目录

CONTENT

openEuler

Linux系统启动流程

POST->Bios->Grub->kernel+initrd->systemd(init)->services



linux下查看系统引导方式

查看linux下是否有 “/sys/firmware/efi” 目录

- **如果不存在，则说明启动方式是 Legacy（BIOS）**

```
linux:~ # ll /sys/firmware/
total 0
drwxr-xr-x  4 root root 0 Nov 28 11:06 acpi
drwxr-xr-x  3 root root 0 Nov 27 18:15 edd
drwxr-xr-x 20 root root 0 Nov 27 18:15 memmap
```

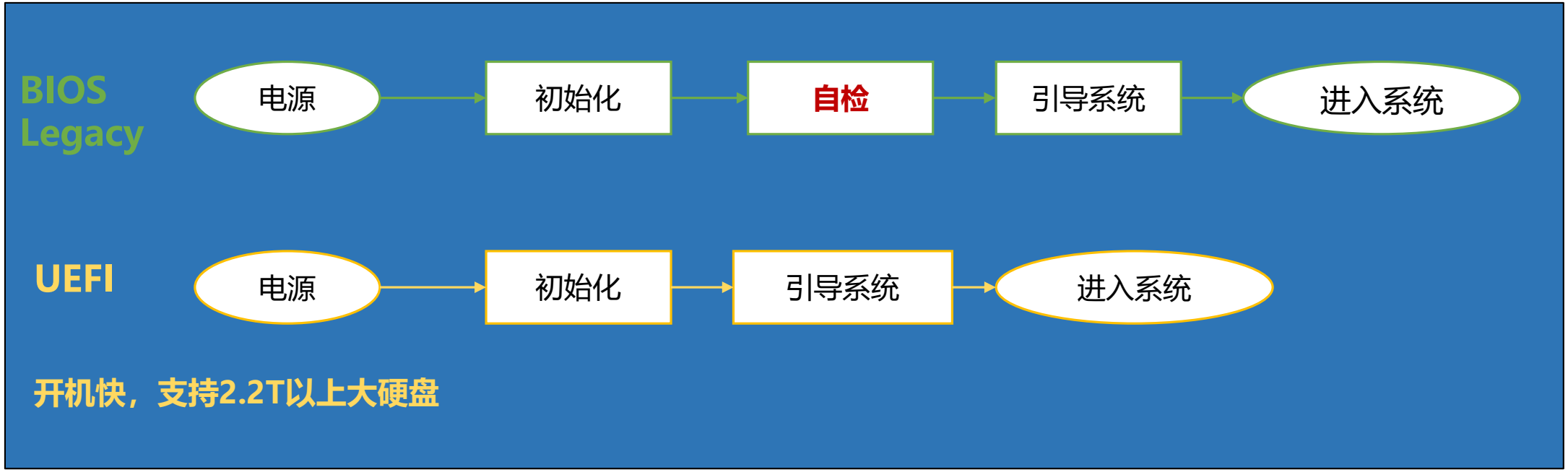
- **如果存在，则说明启动方式是UEFI：**

```
host-8-4-184-55:~ # ll /sys/firmware/
total 0
drwxr-xr-x 5 root root       0 Nov 28 10:22 acpi
drwxr-xr-x 2 root root       0 Nov 28 10:22 devicetree
drwxr-xr-x 3 root root       0 Nov 28 10:22 dmi
drwxr-xr-x 4 root root       0 Nov 28 10:22 efi
-r-------- 1 root root 2097152 Nov 28 10:22 fdt
```

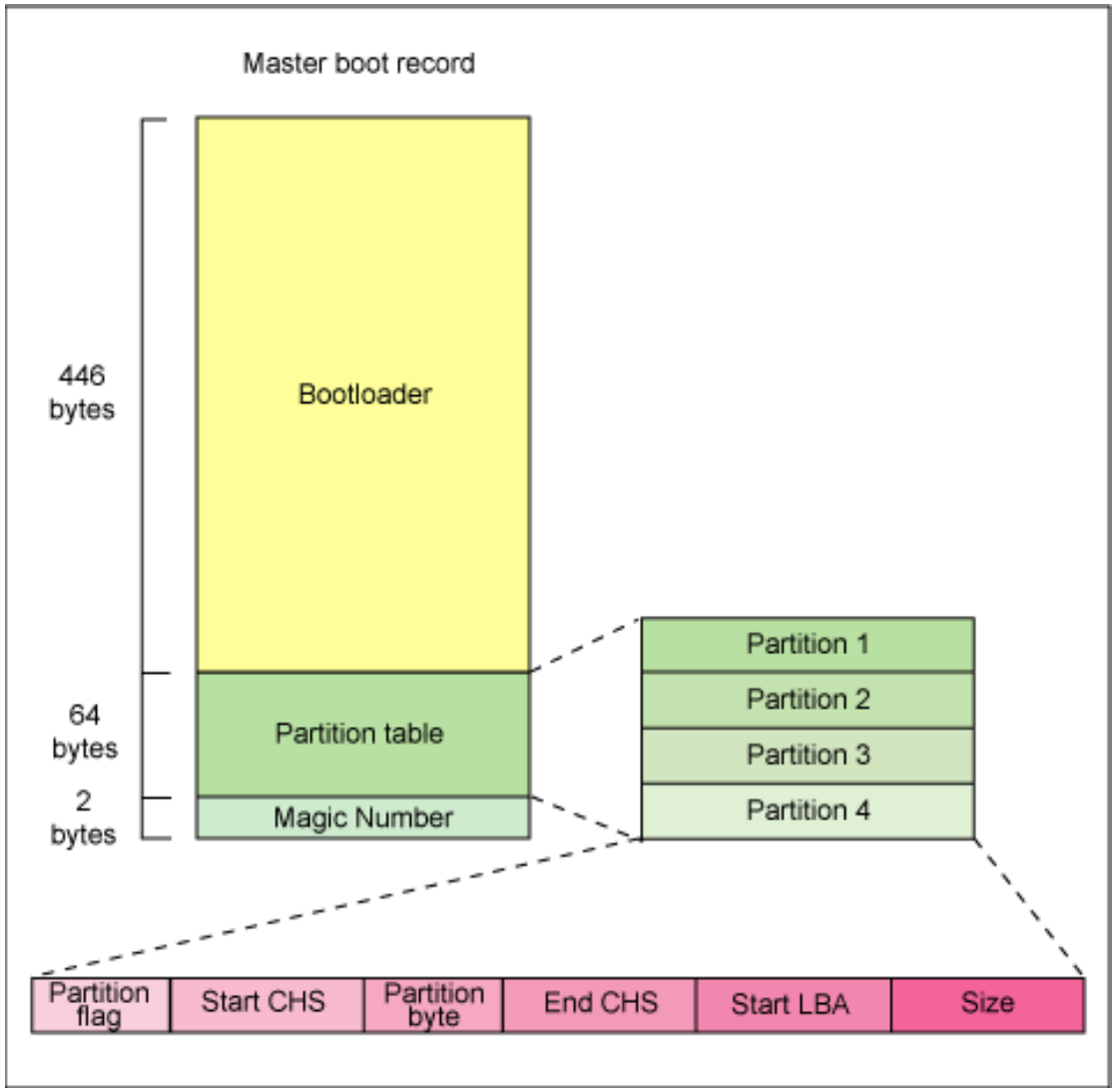
启动流程—BIOS和UEFI

两种BIOS启动模式

- **Legacy** 又叫传统BIOS（全称“Basic Input Output System”，中文名称“基本输入输出系统”），BIOS 程序是机器上电后运行的第一个程序，保存在 BIOS 芯片中，BIOS芯片是一个只读存储器（ROM）。在BIOS设置中我们经常会看到legacy support一词，表示支持传统的BIOS启动引导方式。Legacy 就是传统的BIOS启动，和MBR分区模式相互依存，可以进行MBR分区系统的安装，兼容性较强。但是MBR分区无法支持超过2T的硬盘，并且效率低下，终究是会淘汰的。
- **UEFI**（全称“Unified Extensible Firmware Interface”，中文名称“统一的可扩展固件接口”）是现在新式增强的BIOS启动引导模式，支持GPT（GUID）分区结构，也必须是GPT分区表格式才能启动。UEFI启动引导可加快启动速度，提高开机后操作系统的启动速度。UEFI 甚至还有图形界面，支持鼠标。但实际上，UEFI 最主要的功能和 BIOS Linux 启动过程一样，就是加载 bootloader，并转交控制权给 bootloader。



启动流程—MBR



主引导记录(MBR,Master Boot Record)：位于硬盘第1个扇区(0道0柱面1扇区)，共512字节，由3部分组成：

- 主引导程序(boot loader)占446个字节；
- Partition table区(分区表)，即DPT，占64个字节，硬盘中分区有多少以及每一分区的大小都记在其中；
- magicnumber，占2个字节，固定为55AA。

MBR是不属于任何一个操作系统， 可以通过dd命令进行MBR的读取、写入、删除等操作。

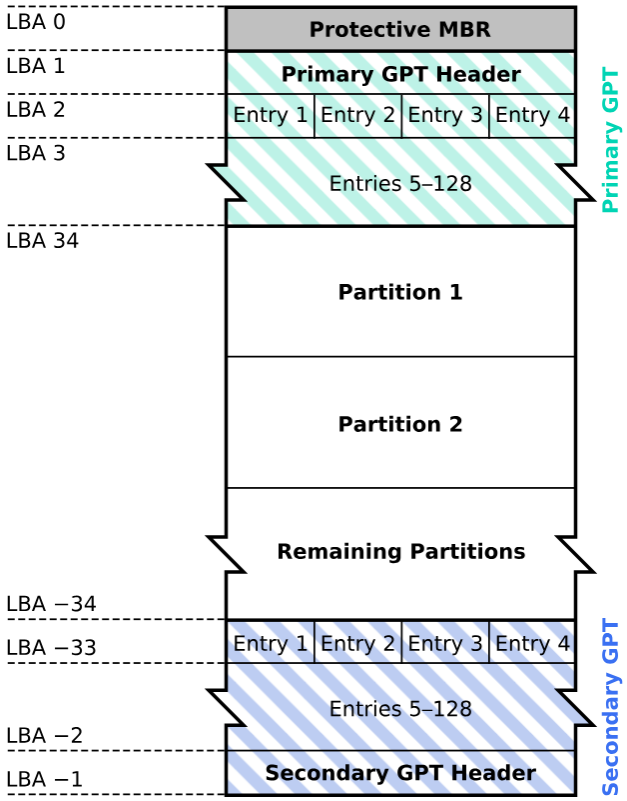
补充说明：

1. 一块硬盘最多4个主分区（Primary Partition）
2. 一块硬盘最多1个扩展分区（Extented Partition）
3. 逻辑分区（Logical Partion）的分区信息是写在扩展分区上
4. 磁盘编号1~4是留给主分区，5以后的编号留给逻辑分区

启动流程—GPT

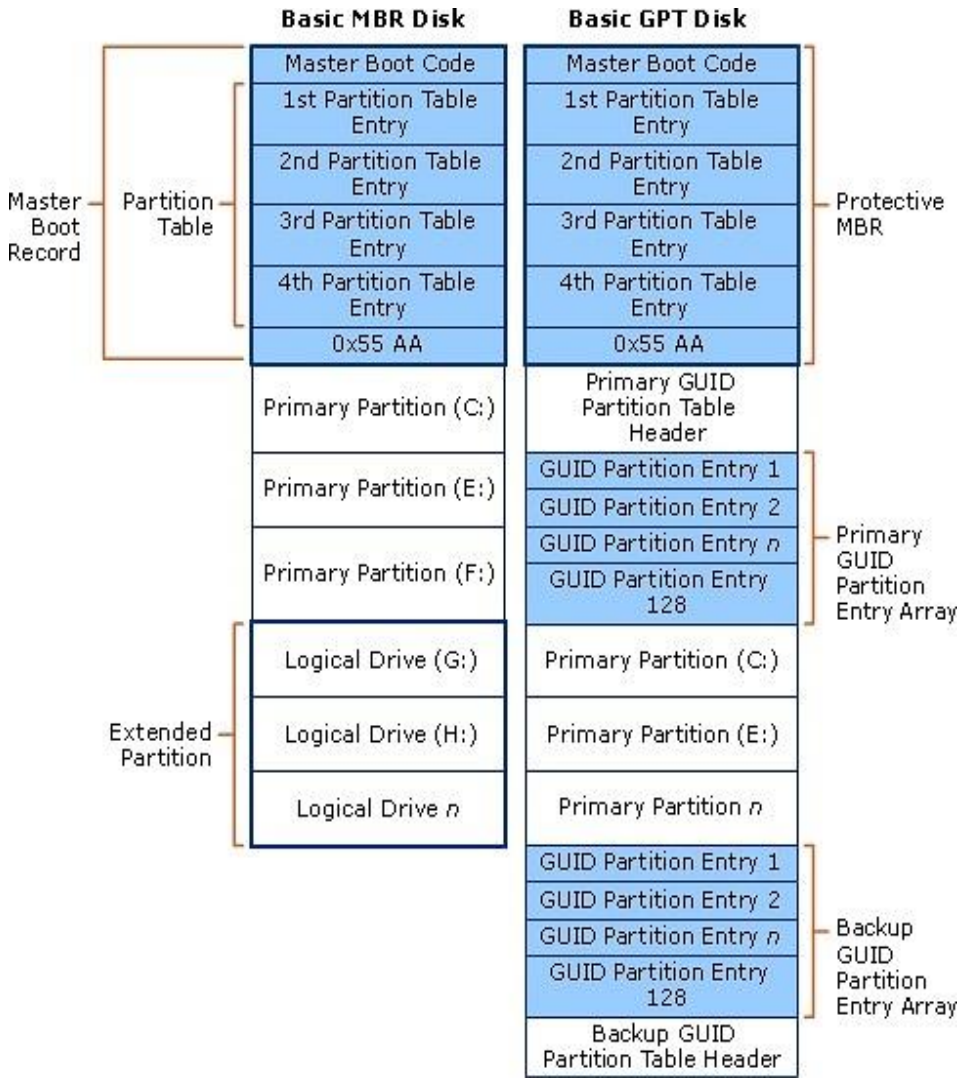
GPT

GPT 使用 LBA 编址，每个 LBA 512B。原来 MBR 占用的第一个扇区现在是 LBA0。GPT 使用了 34 个 LBA 区块来记录分区信息。除了前面 34 个 LBA 之外，整个磁盘的最后 33 个 LBA 也拿来作为另一个备份。具体结构如下：



- LBA0 (MBR 兼容区块) 与 MBR 模式相近，前 446B 还是 bootloader，后 66B 换成一个特殊标记，表明此磁盘是 GPT 分区
- LBA1 (GPT 表头记录) 记录分区表本身的位置与大小，备份分区放置的位置， 以及校验码 (CRC32)
- LBA2-33 (实际记录分区信息) 从 LBA2 区块开始，每个 LBA 都可以记录 4 个分区信息，默认的情况下，总共可以记录 128=4*32 个分区信息

启动流程—分区表（MBR和GPT）



MBR分区，全称"Master Boot Record"，中文名称"主引导记录"。
GPT分区，全称"GUID Partition Table"，中文名称"全局唯一标识磁盘分区表"。

对于现在的系统来说，分区的类型千百种，但对于磁盘分区的layout来说，最常接触的只是三种而已：

- MBR(Master Boot Record)
- GPT(Globe Partition Table)
- Apple Partition(Mixed Partition Table)

- MBR分区表:系统从硬盘启动，硬盘中存放boot loader stage1程序的扇区被称为MBR，它是启动硬盘(如/dev/sda)的首个扇区。
- GPT分区表：对于GPT磁盘采用带有主备分区表结构的GUID分区表。这个结构分别保存在磁盘开头和结尾的部分。相比MBR采用扇区Sector来识别的方法，GPT采用logical Block Address(LBA)来识别。
- Protective MBR: 在GPT磁盘中，LBA 0的位置存放着第一个结构Protective MBR，紧跟着，在LBA 1的位置存放这主GPT头。随后是GUID的分区Entry信息。
- 在GPT分区表的最开头（LBA0），处于兼容性考虑仍然存储了一份传统的MBR，用来防止不支持GPT的硬盘管理工具错误识别并破坏硬盘中的数据，这个MBR也叫做叫做保护MBR。

名称	MBR分区	GPT分区
分区表的位置	0号扇区	1～33号扇区
最大磁盘容量	2TB	8ZB
最大分区数	4个主分区	128
分区工具	fdisk	parted

启动流程—MBR和GPT查看和转换

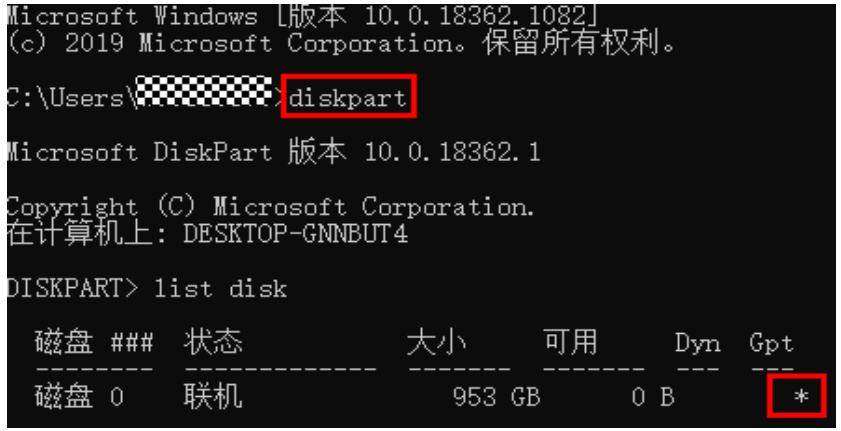
- **openEuler查看MBR方式引导的分区：**

```
# parted -l /dev/sda
Model: LSI Logical Volume (scsi)
Disk /dev/sda: 299GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

# partprobe -s
/dev/sda: msdos partitions 1 2 3
```

- **Windows下查看硬盘引导分区格式**

使用diskpart查看磁盘信息，GPT位置“*”表示使用GPT格式，因此是用UEFI引导：



- **openEuler查看GPT方式引导的分区：**

```
# parted -l /dev/vda
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 50.5GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

# partprobe -s
/dev/vda: gpt partitions 1 2 3
```

引导分区格式转换

转换前需要确认磁盘头剩余扇区必须大于33个。
例如下面vda盘是从2048扇区开始，可以做到无损转换。

```
# fdisk -l /dev/vda
......
Device     Boot   Start      End  Sectors  Size Id Type
/dev/vda1  *       2048  12595199 12593152    6G 83 Linux
/dev/vda2      12595200 104615935 92020736 43.9G 8e Linux LVM
```

- **转换成mbr格式**
  # parted /dev/sdb mklabel msdos -s
- **转化成gpt格式**
  # parted /dev/sdb mklabel gpt -s

转换前请备份好MBR及磁盘头部信息，也可以使用交互式完成格式装换。

随堂测

- 1. 在openEuler中查看系统引导方式?

openEuler

目录

CONTENT

openEuler

系统启动过程中GRUB选择内核的界面

以上是内核选择的菜单界面，如果系统上安装了多个内核版本，则可以通过光标上下选择，则系统会以对应内核版本启动。
以上菜单的配置文件 grub.cfg
UEFI启动模式下的文件路径是 /boot/efi/EFI/openEuler/grub.cfg
传统的legacy BIOS启动模式下的文件路径是 /boot/grub2/grub.cfg

GRUB配置文件 grub.cfg

grub.cfg配置文件中指定了引导的目标内核文件和目标 initrd文件，以及内核启动时设置的参数。
指定内核文件路径以及内核的启动参数：

linux /vmlinuz-5.10.0-60.18.0.50.oe2203.x86_64 root=/dev/mapper/openeuler-root ro resume=/dev/mapper/openeuler-swap rd.lvm.lv=openeuler/root rd.lvm.lv=openeuler/swap crashkernel=512M

指定initrd文件路径：

initrd /initramfs-5.10.0-60.18.0.50.oe2203.x86_64.img

如果系统上安装了多个内核版本，则会存在多个 menuentry，里面指定了对应的内核和initrd文件路径。

```
menuentry 'openEuler (5.10.0-60.18.0.50.oe2203.x86_64) 22.03 LTS' --class openeuler --class gnu-linux --class gnu --class os --u
nrestricted $menuentry_id_option 'gnulinux-5.10.0-60.18.0.50.oe2203.x86_64-advanced-1a22f498-07c1-4390-8604-05af3e2cb7b0' {
        load_video
        set gfxpayload=keep
        insmod gzio
        insmod part_gpt
        insmod ext2
        search --no-floppy --fs-uuid --set=root 87ed55e0-38e0-44cb-8f37-d6fd312b696c
        echo    'Loading Linux 5.10.0-60.18.0.50.oe2203.x86_64 ...'
        linux   /vmlinuz-5.10.0-60.18.0.50.oe2203.x86_64 root=/dev/mapper/openeuler-root ro resume=/dev/mapper/openeuler-swap rd
.lvm.lv=openeuler/root rd.lvm.lv=openeuler/swap crashkernel=512M
        echo    'Loading initial ramdisk ...'
        initrd  /initramfs-5.10.0-60.18.0.50.oe2203.x86_64.img
}
```

查看系统默认启动的内核

查看系统默认启动的内核：

```
# awk -F\' '$1=="menuentry " {print $2}' /boot/efi/EFI/openEuler/grub.cfg
openEuler (5.10.0-60.18.0.40.oe2103.x86_64) 22.03 LTS                    # menuentry0
openEuler (5.10.0-60.18.0.50.oe2203.x86_64) 22.03 LTS                    # menuentry1
```

上述结果输出表明，该Euler系统共有2个内核启动菜单入口，包括一个普通内核和一个救援内核。

```
# grub2-editenv list
saved_entry= openEuler (5.10.0-60.18.0.40.oe2103.x86_64) 22.03 LTS
```

上述结果输出表明，当前系统默认启动内核是menuentry0 ， 内核版本号“5.10.0-60.18.0.40.oe2103.x86_64” 。

设置系统从哪个内核启动

修改默认启动内核：

```
# grep GRUB_DEFAULT /etc/default/grub
GRUB_DEFAULT=saved
# grub2-set-default " openEuler (5.10.0-60.18.0.50.oe2203.x86_64) 22.03 LTS"
# grub2-editenv list
saved_entry= openEuler (5.10.0-60.18.0.50.oe2203.x86_64) 22.03 LTS
```

grub2-set-default命令必须要/etc/default/grub 中配置GRUB_DEFAULT=saved。

重启系统：

```
# reboot
```

重启系统后将会以新内核启动

目录

CONTENT

openEuler

如何管理系统启动服务

简介

systemd是在Linux下，与SysV和LSB初始化脚本兼容的系统和服务管理器。systemd使用socket和D-Bus来开启服务，提供基于守护进程的按需启动策略，支持快照和系统状态恢复，维护挂载和自挂载点，实现了各服务间基于从属关系的一个更为精细的逻辑控制，拥有更高的并行性能。

提供了更快的启动速度

systemd提供了并行启动能力，采用了socket/D-Bus activation等技术启动服务，带来了更快的启动速度。
为了减少系统启动时间，systemd的目标是：
1、尽可能启动更少的进程。
2、尽可能将更多进程并行启动。

使用如下命令查看系统启动服务的依赖关系

```
# systemctl list-dependencies -l
```

返回结果如右图所示，systemd梳理了系统启动服务之间的依赖关系，
没有依赖的服务之间，是并行启动的。

如何管理系统启动服务

查询系统服务是否开机自启动

systemctl is-enabled <系统服务名>

下面以系统的日志服务rsyslog服务举例

```
[root@localhost ~]# systemctl is-enabled rsyslog
enabled
[root@localhost ~]#
```

如果回显是 enabled，表示该服务已经设置了开机自启动。如果回显是 disabled，则表示未设置开机自启动。

设置系统服务开机不启动

systemctl disable <系统服务名>

下面以系统的日志服务rsyslog服务举例

```
[root@localhost ~]# systemctl disable rsyslog
Removed /etc/systemd/system/multi-user.target.wants/rsyslog.service.
[root@localhost ~]#
```

设置系统服务开机启动

systemctl enable <系统服务名>

下面以系统的日志服务rsyslog服务举例

```
[root@localhost ~]# systemctl enable rsyslog
Created symlink /etc/systemd/system/multi-user.target.wants/rsyslog.service → /usr/lib/systemd/system/rsyslog.service.
[root@localhost ~]#
```

如何管理系统启动服务

查看具体服务的启动依赖关系

使用以下命令，查看具体服务的配置文件，其中主要字段如下解释：

systemctl cat <系统服务名>

和依赖关系有关的字段的含义如下：

字段名	含义
Description	简短描述
Requires	当前 Unit 依赖的其他 Unit，如果它们没有运行，当前 Unit 会启动失败
Wants	与当前 Unit 配合的其他 Unit，如果它们没有运行，当前 Unit 不会启动失败
BindsTo	与 Requires 类似，它指定的 Unit 如果退出，会导致当前 Unit 停止运行
Before	如果该字段指定的 Unit 也要启动，那么必须在当前 Unit 之后启动
After	如果该字段指定的 Unit 也要启动，那么必须在当前 Unit 之前启动
Conflicts	这里指定的 Unit 不能与当前 Unit 同时运行
Condition...	当前 Unit 运行必须满足的条件，否则不会运行
Assert...	当前 Unit 运行必须满足的条件，否则会报启动失败

```
[root@localhost ~]# systemctl cat rsyslog
# /usr/lib/systemd/system/rsyslog.service
[Unit]
Description=System Logging Service
;Requires=syslog.socket
Wants=network.target network-online.target
After=network.target network-online.target
Documentation=man:rsyslogd(8)
Documentation=https://www.rsyslog.com/doc/
```

如何管理系统启动服务

Target和运行级别

systemd用目标（target）替代了运行级别的概念，提供了更大的灵活性，如您可以继承一个已有的目标，并添加其他服务，来创建自己的目标。下表列举了systemd下的目标和常见runlevel的对应关系。

运行级别	systemd目标（target）	描述
0	runlevel0.target，poweroff.target	关闭系统。
1, s, single	runlevel1.target，rescue.target	单用户模式。
2, 4	runlevel2.target，runlevel4.target，multi-user.target	用户定义/域特定运行级别。默认等同于3。
3	runlevel3.target，multi-user.target	多用户，非图形化。用户可以通过多个控制台或网络登录。
5	runlevel5.target，graphical.target	多用户，图形化。通常为所有运行级别3的服务外加图形化登录。
6	runlevel6.target，reboot.target	重启系统。
emergency	emergency.target	紧急Shell。

如何管理系统启动服务

使用如下命令，查看当前系统默认启动目标

$ systemctl get-default

```
[root@localhost ~]# systemctl get-default
multi-user.target
[root@localhost ~]#
```

使用如下命令，查看当前系统所有的启动目标

$ systemctl list-units --type=target

```
[root@localhost ~]# systemctl list-units --type=target
  UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
  basic.target           loaded active active Basic System
  getty.target           loaded active active Login Prompts
  local-fs-pre.target    loaded active active Preparation for Local File Systems
  local-fs.target        loaded active active Local File Systems
  multi-user.target      loaded active active Multi-User System
  network-online.target loaded active active Network is Online
  network-pre.target     loaded active active Preparation for Network
  network.target         loaded active active Network
  nfs-client.target      loaded active active NFS client services
  paths.target           loaded active active Path Units
  remote-fs-pre.target   loaded active active Preparation for Remote File Systems
  remote-fs.target       loaded active active Remote File Systems
  rpc_pipefs.target      loaded active active rpc_pipefs.target
  rpcbind.target         loaded active active RPC Port Mapper
  slices.target          loaded active active Slice Units
  sockets.target         loaded active active Socket Units
  sshd-keygen.target     loaded active active sshd-keygen.target
  swap.target            loaded active active Swaps
  sysinit.target         loaded active active System Initialization
  timers.target          loaded active active Timer Units
```

改变系统运行级别

改变默认目标

使用root用户，执行如下命令改变系统默认的目标：

systemctl set-default name.target

```
[root@localhost ~]# systemctl set-default multi-user.target
[root@localhost ~]#
```

使用root用户，执行如下命令，改变当前目标

systemctl isolate name.target

使用root用户，执行如下命令，切换到救援模式

systemctl isolate rescue.target

这个命令等同于 systemctl rescue

使用root用户，执行如下命令，切换到紧急模式

systemctl isolate emergency.target

这个命令等同于 systemctl emergency

> 说明：
> 用户需要重启系统，从紧急模式进入正常模式。

随堂测

- 1. 如何设置系统从救援模式启动？ （多选题）

  A. systemctl isolate rescue.target

  B. systemctl isolate emergency.target

  C. systemctl emergency

  D. systemctl rescue

- 2.系统的rsyslog服务是否随开机自启动。（判断题）

openEuler

Thank you

openEuler