Linux 开机引导和启动过程详解

原创2017-08-25译者：penghusterLinux中国

你是否曾经对操作系统为何能够执行应用程序而感到疑惑？那么本文将为你揭开操作系统引导与启动的面纱。
-- David Both

本文导航

你是否曾经对操作系统为何能够执行应用程序而感到疑惑？那么本文将为你揭开操作系统引导与启动的面纱。

理解操作系统开机引导和启动过程对于配置操作系统和解决相关启动问题是至关重要的。该文章陈述了 GRUB2 引导装载程序[1]开机引导装载内核的过程和 systemd 初始化系统[2]执行开机启动操作系统的过程。

事实上，操作系统的启动分为两个阶段：和。引导阶段开始于打开电源开关，结束于内核初始化完成和 systemd 进程成功运行。启动阶段接管了剩余工作，直到操作系统进入可操作状态。

总体来说，Linux 的开机引导和启动过程是相当容易理解，下文将分节对于不同步骤进行详细说明。

- BIOS 上电自检（POST）
- 引导装载程序 (GRUB2)
- 内核初始化
- 启动 systemd，其是所有进程之父。

注意，本文以 GRUB2 和 systemd 为载体讲述操作系统的开机引导和启动过程，是因为这二者是目前主流的 linux 发行版本所使用的引导装载程序和初始化软件。当然另外一些过去使用的相关软件仍然在一些 Linux 发行版本中使用。

引导过程

引导过程能以两种方式之一初始化。其一，如果系统处于关机状态，那么打开电源按钮将开启系统引导过程。其二，如果操作系统已经运行在一个本地用户（该用户可以是 root 或其他非特权用户），那么用户可以借助图形界面或命令行界面通过编程方式发起一个重启操作，从而触发系统引导过程。重启包括了一个关机和重新开始的操作。

BIOS 上电自检（POST）

上电自检过程中其实 Linux 没有什么也没做，上电自检主要由硬件的部分来完成，这对于所有操作系统都一样。当电脑接通电源，电脑开始执行 BIOS（）的 POST（）过程。

在 1981 年，IBM 设计的第一台个人电脑中，BIOS 被设计为用来初始化硬件组件。POST 作为 BIOS 的组成部分，用于检验电脑硬件基本功能是否正常。如果 POST 失败，那么这个电脑就不能使用，引导过程也将就此中断。

BIOS 上电自检确认硬件的基本功能正常，然后产生一个 BIOS 中断[3] INT 13H，该中断指向某个接入的可引导设备的引导扇区。它所找到的包含有效的引导记录的第一个引导扇区将被装载到内存中，并且控制权也将从引导扇

区转移到此段代码。

引导扇区是引导加载器真正的第一阶段。大多数 Linux 发行版本使用的引导加载器有三种：GRUB、GRUB2 和 LILO。GRUB2 是最新的，也是相对于其他老的同类程序使用最广泛的。

GRUB2

GRUB2 全称是 GRand Unified BootLoader，Version 2（第二版大一统引导装载程序）。它是目前流行的大部分 Linux 发行版本的主要引导加载程序。GRUB2 是一个用于计算机寻找操作系统内核并加载其到内存的智能程序。由于 GRUB 这个单词比 GRUB2 更易于书写和阅读，在下文中，除特殊指明以外，GRUB 将代指 GRUB2。

GRUB 被设计为兼容操作系统多重引导规范[4]，它能够用来引导不同版本的 Linux 和其他的开源操作系统；它还能链式加载专有操作系统的引导记录。

GRUB 允许用户从任何给定的 Linux 发行版本的几个不同内核中选择一个进行引导。这个特性使得操作系统，在因为关键软件不兼容或其它某些原因升级失败时，具备引导到先前版本的内核的能力。GRUB 能够通过文件 `/boot/grub/grub.conf` 进行配置。（LCTT 译注：此处指 GRUB1）

GRUB1 现在已经逐步被弃用，在大多数现代发行版上它已经被 GRUB2 所替换，GRUB2 是在 GRUB1 的基础上重写完成。基于 Red Hat 的发行版大约是在 Fedora 15 和 CentOS/RHEL 7 时升级到 GRUB2 的。GRUB2 提供了与 GRUB1 同样的引导功能，但是 GRUB2 也是一个类似主框架（mainframe）系统上的基于命令行的前置操作系统（Pre-OS）环境，使得在预引导阶段配置更为方便和易操作。GRUB2 通过 `/boot/grub2/grub.cfg` 进行配置。

两个 GRUB 的最主要作用都是将内核加载到内存并运行。两个版本的 GRUB 的基本工作方式一致，其主要阶段也保持相同，都可分为 3 个阶段。在本文将以 GRUB2 为例进行讨论其工作过程。GRUB 或 GRUB2 的配置，以及 GRUB2 的命令使用均超过本文范围，不会在文中进行介绍。

虽然 GRUB2 并未在其三个引导阶段中正式使用这些名词，但是为了讨论方便，我们在本文中使用它们。

阶段 1

如上文 POST（上电自检）阶段提到的，在 POST 阶段结束时，BIOS 将查找在接入的磁盘中查找引导记录，其通常位于 MBR（），它加载它找到的第一个引导记录中到内存中，并开始执行此代码。引导代码（及阶段 1 代码）必须非常小，因为它必须连同分区表放到硬盘的第一个 512 字节的扇区中。 在传统的常规 MBR[5] 中，引导代码实际所占用的空间大小为 446 字节。这个阶段 1 的 446 字节的文件通常被叫做引导镜像（boot.img），其中不包含设备的分区信息，分区是一般单独添加到引导记录中。

由于引导记录必须非常的小，它不可能非常智能，且不能理解文件系统结构。因此阶段 1 的唯一功能就是定位并加载阶段 1.5 的代码。为了完成此任务，阶段 1.5 的代码必须位于引导记录与设备第一个分区之间的位置。在加载阶段 1.5 代码进入内存后，控制权将由阶段 1 转移到阶段 1.5。

阶段 1.5

如上所述，阶段 1.5 的代码必须位于引导记录与设备第一个分区之间的位置。该空间由于历史上的技术原因而空闲。第一个分区的开始位置在扇区 63 和 MBR（扇区 0）之间遗留下 62 个 512 字节的扇区（共 31744 字节），该区域用于存储阶段 1.5 的代码镜像 core.img 文件。该文件大小为 25389 字节，故此区域有足够大小的空间用来存储 core.img。

因为有更大的存储空间用于阶段 1.5，且该空间足够容纳一些通用的文件系统驱动程序，如标准的 EXT 和其它的 Linux 文件系统，如 FAT 和 NTFS 等。GRUB2 的 core.img 远比更老的 GRUB1 阶段 1.5 更复杂且更强大。这意味着 GRUB2 的阶段 2 能够放在标准的 EXT 文件系统内，但是不能放在逻辑卷内。故阶段 2 的文件可以存放于 `/boot` 文件系统中，一般在 `/boot/grub2` 目录下。

注意 `/boot` 目录必须放在一个 GRUB 所支持的文件系统（并不是所有的文件系统均可）。阶段 1.5 的功能是开始执行存放阶段 2 文件的 `/boot` 文件系统的驱动程序，并加载相关的驱动程序。

阶段 2

GRUB 阶段 2 所有的文件都已存放于 `/boot/grub2` 目录及其几个子目录之下。该阶段没有一个类似于阶段 1 与阶段 1.5 的镜像文件。相应地，该阶段主要需要从 `/boot/grub2/i386-pc` 目录下加载一些内核运行时模块。

GRUB 阶段 2 的主要功能是定位和加载 Linux 内核到内存中，并转移控制权到内核。内核的相关文件位于 `/boot` 目录下，这些内核文件可以通过其文件名进行识别，其文件名均带有前缀 vmlinuz。你可以列出 `/boot` 目录中的内容来查看操作系统中当前已经安装的内核。

GRUB2 跟 GRUB1 类似，支持从 Linux 内核选择之一引导启动。Red Hat 包管理器（DNF）支持保留多个内核版本，以防最新版本内核发生问题而无法启动时，可以恢复老版本的内核。默认情况下，GRUB 提供了一个已安装内核的预引导菜单，其中包括问题诊断菜单（recuse）以及恢复菜单（如果配置已经设置恢复镜像）。

阶段 2 加载选定的内核到内存中，并转移控制权到内核代码。

内核

内核文件都是以一种自解压的压缩格式存储以节省空间，它与一个初始化的内存映像和存储设备映射表都存储于 `/boot` 目录之下。

在选定的内核加载到内存中并开始执行后，在其进行任何工作之前，内核文件首先必须从压缩格式解压自身。一旦内核自解压完成，则加载 systemd[6] 进程（其是老式 System V 系统的 init[7] 程序的替代品），并转移控制权到 systemd。

这就是引导过程的结束。此刻，Linux 内核和 systemd 处于运行状态，但是由于没有其他任何程序在执行，故其不能执行任何有关用户的功能性任务。

启动过程

启动过程紧随引导过程之后，启动过程使 Linux 系统进入可操作状态，并能够执行用户功能性任务。

systemd

systemd 是所有进程的父进程。它负责将 Linux 主机带到一个用户可操作状态（可以执行功能任务）。systemd 的一些功能远较旧式 init 程序更丰富，可以管理运行中的 Linux 主机的许多方面，包括挂载文件系统，以及开启和管理 Linux 主机的系统服务等。但是 systemd 的任何与系统启动过程无关的功能均不在此文的讨论范围。

首先，systemd 挂载在 `/etc/fstab` 中配置的文件系统，包括内存交换文件或分区。据此，systemd 必须能够访问位于 `/etc` 目录下的配置文件，包括它自己的。systemd 借助其配置文件 `/etc/systemd/system/default.target` 决定 Linux 系统应该启动达到哪个状态（或）。`default.target` 是一个真实的 target 文件的符号链接。对于桌面系统，其链接到 `graphical.target`，该文件相当于旧式 systemV init 方式的 runlevel 5。对于一个服务器操作系统来说，`default.target` 更多是默认链接到 `multi-user.target`，相当于 systemV 系统的 runlevel 3。 `emergency.target` 相当于单用户模式。

（LCTT 译注：“target” 是 systemd 新引入的概念，目前尚未发现有官方的准确译名，考虑到其作用和使用的上下文环境，我们认为翻译为“目标态”比较贴切。以及，“unit” 是指 systemd 中服务和目标态等各个对象/文件，在此依照语境译作“单元”。）

注意，所有的和均是 systemd 的。

如下表 1 是 systemd 启动的和老版 systemV init 启动的对比。这个 systemd 目标态别名 是为了 systemd 向前兼容 systemV 而提供。这个目标态别名允许系统管理员（包括我自己）用 systemV 命令（例如 `init 3`）改变运行级别。当然，该 systemV 命令是被转发到 systemd 进行解释和执行的。

< 如显示不全，请左右滑动 >

SystemV 运行级别	systemd 目标态	systemd 目标态别名	描述
	halt.target		停止系统运行但不切断电源。
0	poweroff.target	runlevel0.target	停止系统运行并切断电源.
S	emergency.target		单用户模式，没有服务进程运行，文件系统也没挂载。这是一个最基本的运行级别，仅在主控制台上提供一个 shell 用于用户与系统进行交互。
1	rescue.target	runlevel1.target	挂载了文件系统，仅运行了最基本的服务进程的基本系统，并在主控制台启动了一个 shell 访问入口用于诊断。
2		runlevel2.target	多用户，没有挂载 NFS 文件系统，但是所有的非图形界面的服务进程已经运行。
3	multi-user.target	runlevel3.target	所有服务都已运行，但只支持命令行接口访问。
4		runlevel4.target	未使用。
5	graphical.target	runlevel5.target	多用户，且支持图形界面接口。
6	reboot.target	runlevel6.target	重启。
	default.target		这个是总是 multi-user.target 或 graphical.target 的一个符号链接的别名。systemd 总是通过 default.target 启动系统。default.target 绝不应该指向 halt.target、 poweroff.target 或 reboot.target。

表 1 老版本 systemV 的 运行级别与 systemd 与或目标态别名的比较

每个有一个在其配置文件中描述的依赖集，systemd 需要首先启动其所需依赖，这些依赖服务是 Linux 主机运行在特定的功能级别所要求的服务。当配置文件中所有的依赖服务都加载并运行后，即说明系统运行于该目标级别。

systemd 也会查看老式的 systemV init 目录中是否存在相关启动文件，若存在，则 systemd 根据这些配置文件的内容启动对应的服务。在 Fedora 系统中，过时的网络服务就是通过该方式启动的一个实例。

如下图 1 是直接从 bootup 的 man 页面拷贝而来。它展示了在 systemd 启动过程中一般的事件序列和确保成功的启动的基本的顺序要求。

sysinit.target 和 basic.target 目标态可以被视作启动过程中的状态检查点。尽管 systemd 的设计初衷是并行启动系统服务，但是部分服务或功能目标态是其它服务或目标态的启动的前提。系统将暂停于检查点直到其所要求的服务和目标态都满足为止。

sysinit.target 状态的到达是以其所依赖的所有资源模块都正常启动为前提的，所有其它的单元，如文件系统挂载、交换文件设置、设备管理器的启动、随机数生成器种子设置、低级别系统服务初始化、加解密服务启动（如果一个或者多个文件系统加密的话）等都必须完成，但是在 sysinit.target 中这些服务与模块是可以并行启动的。

sysinit.target 启动所有的低级别服务和系统初具功能所需的单元，这些都是进入下一阶段 basic.target 的必要前提。

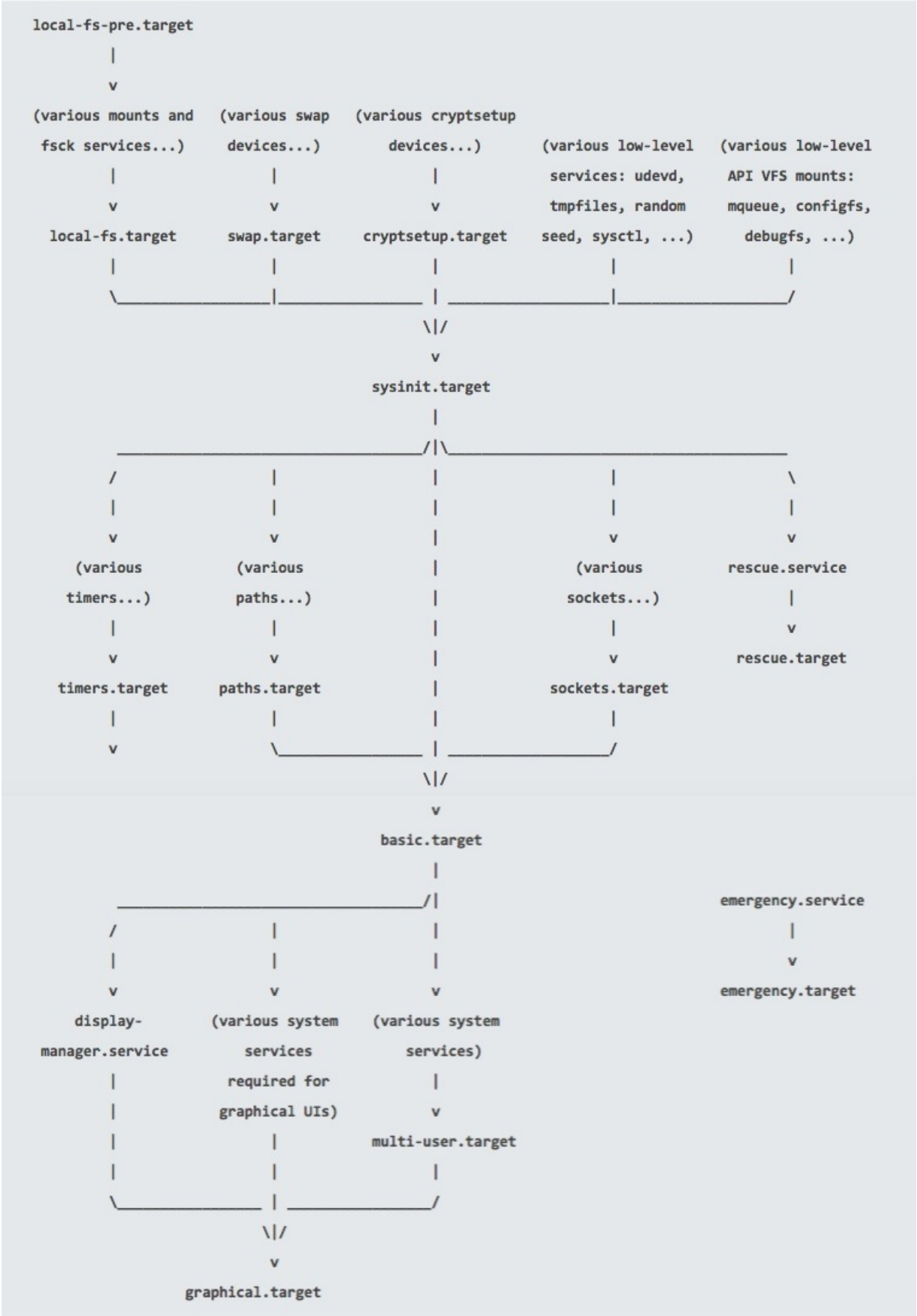

图 1：systemd 的启动流程

在 `sysinit.target` 的条件满足以后，systemd 接下来启动 `basic.target`，启动其所要求的所有单元。 `basic.target` 通过启动下一目标态所需的单元而提供了更多的功能，这包括各种可执行文件的目录路径、

通信 sockets，以及定时器等。

最后，用户级目标态（`multi-user.target` 或 `graphical.target`）可以初始化了，应该注意的是 `multi-user.target` 必须在满足图形化目标态 `graphical.target` 的依赖项之前先达成。

图 1 中，以 `*` 开头的目标态是通用的启动状态。当到达其中的某一目标态，则说明系统已经启动完成了。如果 `multi-user.target` 是默认的目标态，则成功启动的系统将以命令行登录界面呈现于用户。如果 `graphical.target` 是默认的目标态，则成功启动的系统将以图形登录界面呈现于用户，界面的具体样式将根据系统所配置的显示管理器[9]而定。

故障讨论

最近我需要改变一台使用 GRUB2 的 Linux 电脑的默认引导内核。我发现一些 GRUB2 的命令在我的系统上不能用，也可能是我使用方法不正确。至今，我仍然不知道是何原因导致，此问题需要进一步探究。

`grub2-set-default` 命令没能在配置文件 `/etc/default/grub` 中成功地设置默认内核索引，以至于期望的替代内核并没有被引导启动。故在该配置文件中我手动更改 `GRUB_DEFAULT=saved` 为 `GRUB_DEFAULT=2`，2 是我需要引导的安装好的内核文件的索引。然后我执行命令 `grub2-mkconfig > /boot/grub2/grub.cfg` 创建了新的 GRUB 配置文件，该方法如预期的规避了问题，并成功引导了替代的内核。

结论

GRUB2、systemd 初始化系统是大多数现代 Linux 发行版引导和启动的关键组件。尽管在实际中，systemd 的使用还存在一些争议，但是 GRUB2 与 systemd 可以密切地配合先加载内核，然后启动一个业务系统所需要的系统服务。

尽管 GRUB2 和 systemd 都比其前任要更加复杂，但是它们更加容易学习和管理。在 man 页面有大量关于 systemd 的帮助说明，freedesktop.org 也在线收录了完整的此帮助说明[9]。下面有更多相关信息链接。

附加资源

- GNU GRUB[10] (Wikipedia)
- GNU GRUB Manual[11] (GNU.org)
- Master Boot Record[12] (Wikipedia)
- Multiboot specification[13] (Wikipedia)
- systemd[14] (Wikipedia)
- systemd bootup process[15] (Freedesktop.org)
- systemd index of man pages[16] (Freedesktop.org)

作者简介：

David Both 居住在美国北卡罗纳州的首府罗利，是一个 Linux 开源贡献者。他已经从事 IT 行业 40 余年，在 IBM 教授 OS/2 20余年。1981 年，他在 IBM 开发了第一个关于最初的 IBM 个人电脑的培训课程。他也曾在 Red Hat 教授 RHCE 课程，也曾供职于 MCI worldcom，Cico 以及北卡罗纳州等。他已经为 Linux 开源社区工作近 20 年。