RHEL7开机顺序

2015年6月6日　　8:58

1. 计算机接通电源，硬件自检（POST）。

2. BIOS选择引导设备(光盘、网络、硬盘、u盘)。通常为硬盘，BIOS在自检通过后会把控制权交给MBR。主引导记录（MBR）是一种特殊类型的引导扇区，MBR保存硬盘分区的信息和主引导程序(grub)用于加载已安装的操作系统~~(格式化分区命令不会擦除MBR信息，因为此特殊空间不属于任何分区)~~。MBR保存着系统的主引导程序和分区表（grub 446字节，分区表64字节）。

3. 从磁盘读取启动引导程序，然后将系统控制权交给启动引导程序GRUB(在RHEL7中通常是grub2)。GRUB允许用户在计算机启动时选择希望运行的操作系统(启动的配置菜单)。GRUB可用于选择不同的内核，也可用于向这些内核传递启动参数。

4. GRUB加载内核，内核执行systemd程序成为Linux系统的父进程。

   systemd开启和监督整个系统是基于unit的概念。系统初始化需要做的事情非常多。需要启动后台服务，比如启动 ssh 服务；需要做配置工作，比如挂载文件系统。这个过程中的每一步都被 systemd 抽象为一个配置单元，即 unit。可以认为一个服务是一个配置单元，一个挂载点是一个配置单元，一个交换分区的配置是一个配置单元等等。
   下面是一些常见的 unit 类型：
   1. service：守护进程的启动、停止、重启和重载是此类unit中最为明显的几个类型
   2. target ：此类 unit 为其他 unit 进行逻辑分组。它们本身实际上并不做什么，只是引用其他 unit 而已。这样便可以对 unit 做一个统一的控制。

   Systemd使用“target”来处理引导和服务管理过程。这些 systemd里的“target”文件被用于分组不同的引导单元以及启动同步进程。
   systemd执行的第一个目标是default.target。但实际上default.target是指向graphical.target的软链接。文件graphical.target的位置是/usr/lib/systemd/system/graphical.target。（注：这个目录下我们会看到很多后缀是service的文件，还有很多后缀是target的文件）。target可以控制多个后缀serivce的文件。default.target类似于一个快捷方式，最终指向的还是graphical.target。这个target默认的基本就好像init中的runlevel 5

   centos7表面是有“运行级别”这个概念，实际上是为了兼容以前的系统，每个所谓的“运行级别”都有对应的软连接指向，默认的启动级别时/etc/systemd/system/default.target,根据它的指向可以找到系统要进入哪个模式
   模式：
   0 ==> runlevel0.target, poweroff.target
   1 ==> runlevel1.target, rescue.target
   2 ==> runlevel2.target, multi-user.target
   3 ==> runlevel3.target, multi-user.target
   4 ==> runlevel4.target, multi-user.target
   5 ==> runlevel5.target, graphical.target
   6 ==> runlevel6.target, reboot.target

   ```
   # 启动级别target的启动级别和以前的对比：
   poweroff.target    类似于启动级别runlevel 0    # 系统停机状态，系统默认运行级别不能设为0，否则不能正常启动
   resuce.target      类似于启动级别runlevel 1    # 为单用户模式，就像Win下的安全模式，root权限，用于系统维护，禁止远程登陆
   multi-user.target 类似于启动级别runlevel 2  3 # 运行级别2没有NFS支持，运行级别3为标准的运行级
                      类似于启动级别runlevel   4  # 系统未使用，保留
   graphical.target   类似于启动级别runlevel 5    # X11控制台，登陆后进入图形GUI模式
   reboot.target      类似于启动级别runlevel 6    # 系统正常关闭并重启，默认运行级别不能设为6，否则不能正常启动

   [root@test ~]# ls -l /etc/systemd/system/default.target
   lrwxrwxrwx. 1 root root 36 5月  11 20:31 /etc/systemd/system/default.target ->
   /lib/systemd/system/graphical.target

   [root@test ~]# ls /etc/systemd/system/graphical.target.wants/
   # 这个target将自己的子单元放在目录“/etc/systemd/system/graphical.target.wants”里
   #：graphical.target脚本解读
   [Unit]
   Description=Graphical Interface    #  描述这个是一个图形化的接口
   Documentation=man:systemd.special(7)
   Requires=multi-user.target       # 相当于启动级别3
   After=multi-user.target
   Conflicts=rescue.target
   ```

Wants=display-manager.service
AllowIsolate=yes

[Install]
Alias=default.target
通过这个default.target这个脚本，指向新的默认启动级别

启动multi-user.target而这个target将自己的子单元放在目录“/etc/systemd/system/multi-user.target.wants”里。这个target为多用户支持设定系统环境。非root用户会在这个阶段的引导过程中启用；防火墙相关的服务也会在这个阶段启动。
"multi-user.target"会将控制权交给另一层“basic.target”。
#：multi-user.targe脚本解读
[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes

[Install]
Alias=default.target

5.mutli-user.target将控制权交给basic.target。Basic.target用于启动普通服务，主要是图形化管理的服务。它通过目录（/etc/systemd/system/basic.target.wants）来决定启动哪些service。basic.target之后将控制权交给sysinit.target.
[Unit]
Description=Basic System
Documentation=man:systemd.special(7)
Requires=sysinit.target
Wants=sockets.target timers.target paths.target slices.target
After=sysinit.target sockets.target timers.target paths.target slices.target
RefuseManualStart=yes

6. "sysinit.target"会启动重要的系统服务例如系统挂载，内存交换空间和设备，内核补充选项等等。sysinit.target在启动过程中会传递给local-fs.target。这个target单元的内容如下面截图里所展示。
根据basic内容的说明，它把控制权交给sysinit.target。
[Unit]
Description=System Initialization
Documentation=man:systemd.special(7)
Conflicts=emergency.service emergency.target
Wants=local-fs.target swap.target
After=local-fs.target swap.target emergency.service emergency.target
RefuseManualStart=yes

7. sysinit在启动过程中，将控制权传递给了local-fs。local-fs.target。local-fs.target这个target不启动与用户相关的服务，它仅仅处理底层核心服务，它会根据/etc/fstab和/etc/initab来执行
[Unit]
Description=Local File Systems
Documentation=man:systemd.special(7)
After=local-fs-pre.target
DefaultDependencies=no
Conflicts=shutdown.target
OnFailure=emergency.target
OnFailureIsolate=no

systemd 相比 init 的优点
1. 启动速度快、各服务平行运行。systemd 提供了比 upstart 更激进的并行启动能力。systemd 的目标是：尽可能启动更少的进程和尽可能将更多进程并行启动。
2. systemd 提供按需启动能力。当sysvinit系统初始化的时候，它会将所有可能用到的后台服务进程全部启动运行。并且系统必须等待所有的服务都启动就绪之后，才允许用户登录。这种做法有两个缺点：首先是启动时间过长，其次是系统资源浪费。某些服systemd 可以提供按需启动的能力，只有在某个服务被真正请求的时候才启动它。
3. systemd 自带日志服务journald。
4. 实现事务性依赖关系管理。系统启动过程是由很多的独立工作共同组成的，这些工作之间可能存在依赖关系，比如挂载一

个 NFS 文件系统必须依赖网络能够正常工作。systemd 虽然能够最大限度地并发执行很多有依赖关系的工作，但是类似"挂载NFS"和"启动网络"这样的工作还是存在天生的先后依赖关系，无法并发执行。对于这些任务，systemd 维护一个"事务一致性"的概念，保证所有相关的服务都可以正常启动而不会出现互相依赖，以至于死锁的情况。
5. 采用 linux 的 cgroups 跟踪和管理进程的生命周期。systemd 利用了 Linux 内核的特性即 cgroups 来完成跟踪的任务。当停止服务时，通过查询 cgroups ，systemd 可以确保找到所有的相关进程，从而干净地停止服务。
6. 自动挂载的管理。systemd内建了自动挂载服务。

参考文献 https://www.cnblogs.com/swordxia/p/4521428.html 走进Linux之systemd启动过程
参考文献 https://www.linuxidc.com/Linux/2018-03/151291.htm 从 init 系统说起
参考文献 https://www.linuxidc.com/Linux/2014-11/109232.htm systemd详解