服务与进程实体化

- 前言：

还记得前面我们经常用到的 systemctl 指令吗？当初我用它的时候一脸懵*，其实它背后有很多奥秘的，这节我们就来揭开它的神秘面纱。

- 正文：

1、了解Linux的开机启动流程

在Linux里，不管是一个软件、一个服务、一个脚本，归根结底还是进程，既然一切都是进程，那到底是谁启动的呢？其实是 systemed。那么又是谁启动的systemd呢？这就得从Linux的开机说起了：

①计算机通电。

②通过BIOS检查和发现所有硬件设备以及找硬件的一个起点（引导设备）。操作系统存在硬盘上，说白了就是以硬盘为起点，然后按顺序启动引导设备。

③找到硬盘了，但是数据很多，只好看MBR了解一下硬盘的整体框架，看完框架才能开始读其他的数据。MBR是硬盘被加载后第一个被读到的数据。

④开始读其他需要的数据了，其他很多数据例如音乐电影之类的，操作系统不需要，那么系统就需要一个引导程序GRUB，由它负责找到操作系统数据的位置。

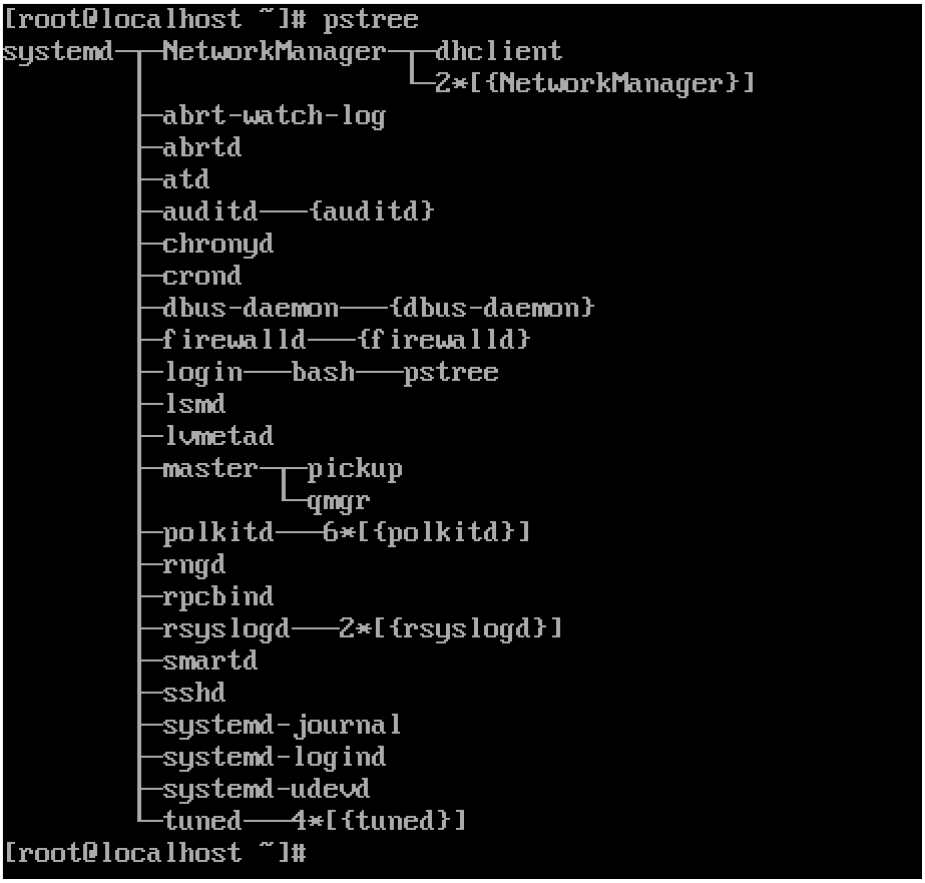
⑤GRUB会引导到Linux的内核上。我们平时说的Linux，其实原本就是指它的内核kernal，或者说驱动程式，驱动是为了调度硬件而存在的。其他所有东西都是内核的附属品。GRUB引导内核（各种驱动）启动，也就是Linux启动了。

⑥Linux系统启动了第一个进程——init程序。之后其他进程由init创造。

init程序在发展中有很多版本，主流是upstart（CentOS 6.x）、systemd（CentOS 7.x）。

```
[root@localhost ~]# ps -ef | grep systemd
root          1     0  0 15:14 ?        00:00:01 /usr/lib/systemd/systemd --switched-root --system --
deserialize 22
root        543     1  0 15:14 ?        00:00:00 /usr/lib/systemd/systemd-journald
root        580     1  0 15:14 ?        00:00:00 /usr/lib/systemd/systemd-udevd
root        748     1  0 15:14 ?        00:00:00 /usr/lib/systemd/systemd-logind
dbus        761     1  0 15:14 ?        00:00:00 /usr/bin/dbus-daemon --system --address=systemd: --n
ofork --nopidfile --systemd-activation
root       1439  1381  0 15:19 tty1     00:00:00 grep --color=auto systemd
[root@localhost ~]# ps -ef | grep init
root       1441  1381  0 15:19 tty1     00:00:00 grep --color=auto init
[root@localhost ~]#
```

看第一项，可以看到systemd进程号是1，其父进程号是0；后面四条父进程号是1，是由systemd创造的。



可以用pstree进程树来查看父子关系，如图，也确实对应上了上上图中的关系。

2、CentOS 7.x 专属服务管理器 systemd

systemd这玩意吧，可以说是第一个启动的服务，也可以说是系统的一个独特的服务体系框架。其实systemd更多的功能是管理和维护所有服务。它的总体设计框架很大，感兴趣自行上网查看。

systemd Utilities 是systemd的自带工具，有了这些工具，就可以轻松地访问systemd。在这些工具中，首先就是systemctl，它可以让systemd去做各种各样的工作。

```
  UNIT                                LOAD   ACTIVE SUB     DESCRIPTION
  abrt-ccpp.service                   loaded active exited  Install ABRT coredump hook
  abrt-oops.service                   loaded active running ABRT kernel log watcher
  abrtd.service                       loaded active running ABRT Automated Bug Reporting Tool
  atd.service                         loaded active running Job spooling tools
  auditd.service                      loaded active running Security Auditing Service
  blk-availability.service            loaded active exited  Availability of block devices
  chronyd.service                     loaded active running NTP client/server
  crond.service                       loaded active running Command Scheduler
  dbus.service                        loaded active running D-Bus System Message Bus
  firewalld.service                   loaded active running firewalld - dynamic firewall daemon
  getty@tty1.service                  loaded active running Getty on tty1
● kdump.service                       loaded failed failed  Crash recovery kernel arming
  kmod-static-nodes.service           loaded active exited  Create list of required static device nod
  libstoragemgmt.service              loaded active running libstoragemgmt plug-in server daemon
  lvm2-lvmetad.service                loaded active running LVM2 metadata daemon
  lvm2-monitor.service                loaded active exited  Monitoring of LVM2 mirrors, snapshots etc
  lvm2-pvscan@8:17.service            loaded active exited  LVM2 PV scan on device 8:17
  lvm2-pvscan@8:18.service            loaded active exited  LVM2 PV scan on device 8:18
  lvm2-pvscan@8:19.service            loaded active exited  LVM2 PV scan on device 8:19
  lvm2-pvscan@8:2.service             loaded active exited  LVM2 PV scan on device 8:2
  network.service                     loaded active exited  LSB: Bring up/down networking
  NetworkManager-wait-online.service  loaded active exited  Network Manager Wait Online
  NetworkManager.service              loaded active running Network Manager
  polkit.service                      loaded active running Authorization Manager
  postfix.service                     loaded active running Postfix Mail Transport Agent
  rhel-dmesg.service                  loaded active exited  Dump dmesg to /var/log/dmesg
  rhel-domainname.service             loaded active exited  Read and set NIS domainname from /etc/sys
  rhel-import-state.service           loaded active exited  Import network configuration from initram
  rhel-readonly.service               loaded active exited  Configure read-only root support
  rngd.service                        loaded active running Hardware RNG Entropy Gatherer Daemon
  rpcbind.service                     loaded active running RPC bind service
  rsyslog.service                     loaded active running System Logging Service
  smartd.service                      loaded active running Self Monitoring and Reporting Technology
  sshd.service                        loaded active running OpenSSH server daemon
  sysstat.service                     loaded active exited  Resets System Activity Logs
lines 1-36
```

这是执行 systemctl list-units --type=services 后的结果。systemd下的每一个进程、服务、配置，都管其叫做一个单位（unit），该命令列出了type（类型）为 service 的所有项，往下翻翻，其中会有我们熟悉的 sshd.service 。另外，systemctl 动词（如start） 名称，一类的命令，允许我们指示一个服务开启关闭暂停等。

以sshd服务为例，看看它背后到底干了什么：

```
[root@localhost ~]# systemctl cat sshd.service
# /usr/lib/systemd/system/sshd.service
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
[root@localhost ~]# ps -ef | grep sshd
root      1138     1  0 15:14 ?        00:00:00 /usr/sbin/sshd -D
root      1475  1381  0 15:39 tty1     00:00:00 grep --color=auto sshd
[root@localhost ~]# systemctl stop sshd.service
[root@localhost ~]# ps -ef | grep sshd
root      1485  1381  0 15:39 tty1     00:00:00 grep --color=auto sshd
[root@localhost ~]# _
```

ExecStart一栏指出，sshd服务开启时，只做了一件事——执行一条sshd的命令，ps可以查出来。当关闭这个服务后，这个进程就不存在了。

systemctl status + 服务名称 用来查看某一个服务当前的状态如何。

当然只看状态时看不出来一个服务哪里出错了，所以这里要用到 journalctl -xe 来查看日志信息。journal是systemd提供的又一重要工具，使用它，可以上下翻页查看历史Unit启动运行信息，如果有问题，可以在上面查看问题出在哪里，默认优先显示最近的操作。

更多的相关命令可以百度百度，这里就不赘述了。

3、systemd的启动设置

在Linux里，设置某一个服务是否开机启动是十分重要的一个环节。systemctl enable/disable 服务名称，开机启动/关闭服务指令。

一个服务要是想被启动，必须要有对应的配置文件，systemd下的服务也是如此。而systemd下的服务配置文件都保存在/usr/lib/systemd/system里。但是Linux在启动时只关注/etc/systemd/system这个目录，所以/usr下的配置文件要生成软链接放在/etc下才能做到开机启动。

所以归根结底，enable/disable 说白了就是在 /etc/systemd/system 里生成/删除一个对应服务的软链接而已。另外，想查看自启的服务，可以systemctl list-unit-files | grep enabled查看。

4、结合target加深理解运行级别

还记得前面提到的运行级别吗？还记得每一个运行级别后面的.target后缀吗？

一个target其实就是一组units，即一个target包含了一组进程、服务、配置等。unit可以理解为实在存在的东西，而target是它们的集合体。

```
[root@localhost ~]# ls /etc/systemd/system
basic.target.wants                                local-fs.target.wants
dbus-org.fedoraproject.FirewallD1.service         multi-user.target.wants
dbus-org.freedesktop.nm-dispatcher.service        network-online.target.wants
default.target                                    sockets.target.wants
default.target.wants                              sysinit.target.wants
dev-virtio\x2dports-org.qemu.guest_agent.0.device.wants  system-update.target.wants
getty.target.wants
[root@localhost ~]# find /etc/systemd/system/ -name 'sshd.service'
/etc/systemd/system/multi-user.target.wants/sshd.service
[root@localhost ~]# systemctl cat sshd.service
# /usr/lib/systemd/system/sshd.service
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
[root@localhost ~]# _
```

之前说过开机启动服务都在/etc/systemd/system文件夹里，ssh服务是开机启动的，但为什么里面没有它的软链接捏？我们通过看到这一大截的.target文件，估计也差的八九不离十了，然后find一看，果然sshd服务在target里，还是那个多用户运行级别里。开机时启动的时multi-user组，顺带就把里面的sshd给启动了。另外，在定义一个systemd Unit时，它的配置文件会指明这个unit属于哪个target（install项里）。

所以说，所谓的运行级别，不过是把一个target组设置成开机启动而已（完全可以用其他的target来作为开机运行级别，但为了安全还是别这么干）。systemd下还定义许多的targets，可以通过 systemctl list-units --type=target来查看。

- 后记：

很简单的一节，算是对前面知识的一个补充。