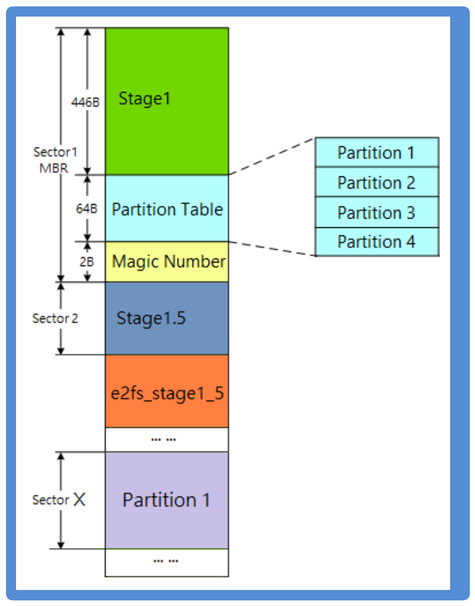
Linux开机启动图（systemd）

Bios引导阶段

1. Post（power On Self Test）
   - 检测电信号是否完整，外围设备是否正常
     - 如果正常产生一个中断int 13h
       该中断指向启动设备的可引导扇区
2. Runtime
   - 根据启动顺序，找到对应启动设备
     - 读取Mbr（主引导扇区）
       - Mbr（主引导记录）存储于磁盘的头部0柱面0磁头0扇区，大小为512bytes
         其中，446bytes用于存储bootloader程序，通常叫做引导镜像boot.img
         接下来64bytes用于存储分区表信息
         最后2bytes用于mbr的有效性检查。



Bootloader(grub2)阶段

Stage1

- 硬件初始化（看门狗、Ram、栈初始化等）
  为加载Stage1.5阶段代码准备ram空间
  将磁盘第2个扇区的内容加载的到内存
  - 使用汇编语言实现，Stage1/start1.s

Stage1.5

- Linux根文件系统是ext之类的文件系统，所以此阶段的目的是能够识别linux根文件系统，识别了文件系统，就能在磁盘上识别文件了，识别linux文件系统的代码存放在磁盘第3个扇区到第n个扇区，n的值取决于文件系统代码段的大小，此阶段就是把这些扇区的内容加载到内存，为stage2做准备
  - 使用汇编语言实现，Stage2/start.s

Stage2

- 读取Grub程序，根据/boot/grub2/grub.conf文件查找kernel信息

```
# DO NOT EDIT THIS FILE
# It is automatically generated by grub2-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub

menuentry 'SLES 15-SP1'  --class sles --class gnu-linux --class gnu
menuentry 'Advanced options for SLES 15-SP1'
```

```
# If you change this file, run 'grub2-mkconfig -o /boot/grub2/grub.cfg' afterwards to update
GRUB_DEFAULT=saved
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=8
GRUB_CMDLINE_LINUX_DEFAULT="splash=silent resume=/dev/systemvg/swaplv mitigations=auto quiet

GRUB_BACKGROUND=/boot/grub2/themes/SLE/background.png
GRUB_THEME=/boot/grub2/themes/SLE/theme.txt
SUSE_BTRFS_SNAPSHOT_BOOTING="true"
GRUB_DISABLE_OS_PROBER="true"
GRUB_ENABLE_CRYPTODISK="n"
```

- 显示系统菜单界面

- 加载Kernel程序到内存

```
suse1:/boot # ls -l vmlinuz initrd
lrwxrwxrwx 1 root root 26 Mar 22 09:35 initrd -> initrd-4.12.14-195-default
lrwxrwxrwx 1 root root 27 Mar 22 09:35 vmlinuz -> vmlinuz-4.12.14-195-default
```

```
Loading Linux 4.12.14-195-default ...
Loading initial ramdisk ...
```

Kernel启动阶段

系统将解压后的内核放置在内存之中，并调用start_kernel()函数来启动一系列的初始化函数，start_kenrel()定义在init/main.c中，真正的内核初始化过程是从这里才开始。

start_kernel()中主要执行了以下操作：
(l) 在屏幕上打印出当前的内核版本信息
(2) 执行setup_arch()，对系统结构进行设置
(3)执行sched_init()，对系统的调度机制进行初始化。先是对每个可用CPU上的runqueque进行初始化;然后初始化0号idle进程，即系统空闲时占据CPU的进程
(4)执行parse_early_param()和parsees_args()解析系统启动参数
(5)执行trap_in itQ，设置了系统中断向量表。0-19号用于CPU异常处理; 然后初始化系统调用向量; 最后调用cpu_init()完善对CPU的初始化，用于支持进程调度机制，包括设定相关寄存器等等
(6)执行rcu_init()，初始化系统中的Read-Copy Update互斥机制
(7)执行init_IRQ()函数，初始化用于外设的中断，完成对IDT的最终初始化过程
(8)执行init_timers()， softirq_init()和time_init()函数，分别初始系统的定时器机制，软中断机制以及系统日期和时间
(9)执行mem_init()函数，初始化物理内存页面的page数据结构描述符，完成对物理内存管理机制的创建
(l0)执行kmem_cache_init()，完成对通用slab缓冲区管理机制的初始化工作
(l1)执行fork_init()，计算出当前系统的物理内存容量能够允许创建的进程(线程)数量
(l2)执行proc_caches_init() , bufer_init()， unnamed_dev_init() .vfs_caches_init()， signals_init()等函数对各种管理机制建立起专用的slab缓冲区队列
(l3)执行proc_root_init()函数，对虚拟文件系统/proc进行初始化

start_kenrel()的结尾，内核通过kenrel_thread()创建出第一个系统内核线程(即l号进程)，systemd进程

systemd 进程是 init 进程的替代品，是所有进程的父进程。

```
susel:~ # ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.5  0.2 222568  8928 ?        Ss   23:24   0:06 /usr/lib/systemd/systemd
root           2  0.0  0.0      0     0 ?        S    23:24   0:00 [kthreadd]
root           4  0.0  0.0      0     0 ?        S<   23:24   0:00 [kworker/0:0H]
root           5  0.0  0.0      0     0 ?        S    23:24   0:00 [kworker/u256:0]
root           6  0.0  0.0      0     0 ?        S<   23:24   0:00 [mm_percpu_wq]
root           7  0.0  0.0      0     0 ?        S    23:24   0:00 [ksoftirqd/0]
root           8  0.0  0.0      0     0 ?        S    23:24   0:00 [rcu_sched]
root           9  0.0  0.0      0     0 ?        S    23:24   0:00 [rcu_bh]
root          10  0.0  0.0      0     0 ?        S    23:24   0:00 [migration/0]
```

之后的引导过程将由systemd完成，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，同时也对应于原始init进程的runlevel

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，仅提供一个 shell 用于交互。
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		systemd 总是通过 default.target 启动系统，这个目标态target是 multi-user.target 或 graphical.target 的一个符号链接的别名。

```
suse1:/usr/lib/systemd/system # ls *.target
basic.target            halt.target               network-online.target   rpcbind.target    sockets.target
bluetooth.target        hibernate.target          network-pre.target      runlevel0.target  sound.target
brltty.target           hybrid-sleep.target       network.target          runlevel1.target  suspend.target
brltty@.target          initrd-fs.target          nfs-client.target       runlevel2.target  swap.target
cryptsetup-pre.target   initrd-root-device.target nss-lookup.target       runlevel3.target  sysinit.target
cryptsetup.target       initrd-root-fs.target     nss-user-lookup.target  runlevel4.target  system-update.target
ctrl-alt-del.target     initrd-switch-root.target paths.target            runlevel5.target  time-sync.target
default.target          initrd.target             poweroff.target         runlevel6.target  timers.target
emergency.target        kexec.target              printer.target          shutdown.target   umount.target
exit.target             local-fs-pre.target       reboot.target           sigpwr.target     xvnc.target
final.target            local-fs.target           remote-fs-pre.target    sleep.target
getty.target            machines.target           remote-fs.target        slices.target
graphical.target        multi-user.target         rescue.target           smartcard.target
suse1:/usr/lib/systemd/system # ls -l runlevel?.target
lrwxrwxrwx 1 root root 15 Feb 15  2019 runlevel0.target -> poweroff.target
lrwxrwxrwx 1 root root 13 Feb 15  2019 runlevel1.target -> rescue.target
lrwxrwxrwx 1 root root 17 Feb 15  2019 runlevel2.target -> multi-user.target
lrwxrwxrwx 1 root root 17 Feb 15  2019 runlevel3.target -> multi-user.target
lrwxrwxrwx 1 root root 17 Feb 15  2019 runlevel4.target -> multi-user.target
lrwxrwxrwx 1 root root 16 Feb 15  2019 runlevel5.target -> graphical.target
lrwxrwxrwx 1 root root 13 Feb 15  2019 runlevel6.target -> reboot.target
```

```
suse1:/usr/lib/systemd/system # ls -l default.target
lrwxrwxrwx 1 root root 16 Feb 15  2019 default.target -> graphical.target
```

实际上第一个目标default.target是指向graphical.target的软链接目标

```
suse1:/usr/lib/systemd/system # cat graphical.target
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
```

此目标依赖于multi-user.target，此目标想做的事情是执行display-manager.service

```
suse1:/usr/lib/systemd/system # cat display-manager.service
[Unit]
Description=X Display Manager
Requires=var-run.mount
Conflicts=getty@tty7.service plymouth-quit.service
After=ypbind.service gpm.service time-sync.target winbind.service acpid.service
Wants=remote-fs.target dbus.socket systemd-user-sessions.service systemd-logind.service
After=remote-fs.target dbus.socket systemd-user-sessions.service systemd-logind.service
OnFailure=plymouth-quit.service

[Service]
Type=forking
PIDFile=/var/run/displaymanager.pid
ExecStart=/usr/lib/X11/display-manager start
ExecStop=/usr/lib/X11/display-manager stop
ExecReload=/usr/lib/X11/display-manager reload
KillMode=process
```

systemd执行的第一个target是default.target，
此文件位于/usr/lib/systemd/system下，
且systemd执行的所有target都在此目录下

每个target有一个在其配置文件中描述的Requires，systemd 需要首先启动其所需Requires，这些Requires 是 Linux 主机运行在特定的级别所要求的服务。

```
suse1:/usr/lib/systemd/system # cat multi-user.target
[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
```

```
suse1:/usr/lib/systemd/system/multi-user.target.wants # ls
after-local.service
dbus.service
getty.target
plymouth-quit-wait.service
plymouth-quit.service
systemd-logind.service
systemd-update-utmp-runlevel.service
systemd-user-sessions.service
```

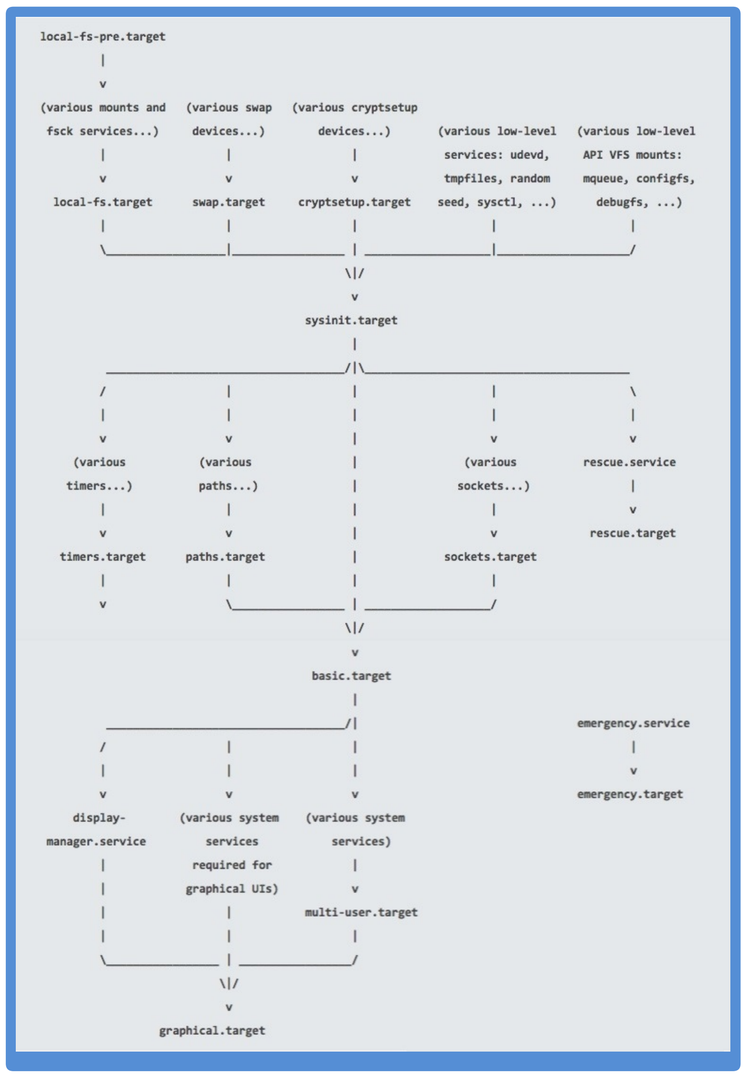
```
suse1:/usr/lib/systemd/system # cat basic.target
[Unit]
Description=Basic System
Documentation=man:systemd.special(7)
Requires=sysinit.target
Wants=sockets.target timers.target paths.target slices.target
After=sysinit.target sockets.target paths.target slices.target tmp.mount

RequiresMountsFor=/var /var/tmp
Wants=tmp.mount
```

```
suse1:/usr/lib/systemd/system # cat 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
```

```
[Unit]
Description=Remount Root and Kernel File Systems
Documentation=man:systemd-remount-fs.service(8)
Documentation=https://www.freedesktop.org/wiki/Software/systemd/APIFileSystems
DefaultDependencies=no
Conflicts=shutdown.target
After=systemd-fsck-root.service
Before=local-fs-pre.target local-fs.target shutdown.target
Wants=local-fs-pre.target
ConditionPathExists=/etc/fstab

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/lib/systemd/systemd-remount-fs
```



当配置文件中所有的Requires都加载并运行后，系统也就成功执行。