

Linux 内核引导参数简介

作者：[金步国](#)

版权声明

本文作者是一位自由软件爱好者，所以本文虽然不是软件，但是本着 GPL 的精神发布。任何人都可以自由使用、转载、复制和再分发，但必须保留作者署名，亦不得对声明中的任何条款作任何形式的修改，也不得附加任何其它条件。您可以自由链接、下载、传播此文档，但前提是必须保证全文完整转载，包括完整的版权信息和作译者声明。

其他作品

本文作者十分愿意与他人共享劳动成果，如果你对我的其他翻译作品或者技术文章有兴趣，可以在如下位置查看现有作品的列表：

- [金步国作品列表](#)

BUG 报告，切磋与探讨

由于作者水平有限，因此不能保证作品内容准确无误，请在阅读中自行鉴别。如果你发现了作品中的错误，请您来信指出，哪怕是错别字也好，任何提高作品 质量的建议我都将虚心接纳。如果你愿意就作品中的相关内容与我进行进一步切磋与探讨，也欢迎你与我联系。联系方式：Email: csfrank@citiz.net ； QQ: 70171448 ； MSN: csfrank122@hotmail.com

概述

内核引导参数大体上可以分为两类：一类与设备无关、另一类与设备有关。内核源码树下的 Documentation/kernel-parameters.txt 文件列出了所有可用的引导参数，并指明了处理每个参数的具体文件。注意：对于模块而言，引导参数只能用于直接编译到核心里的模块，格式是使用“模块名.参数=值”模式指定，比如：usbcore.blinkenlights=1。动态加载的模块可以在 modprobe 命令行上指定相应的参数值，比如：modprobe usbcore.blinkenlights=1。

可以使用“modinfo -p \$ {module name}”命令显示可加载模块的所有可用参数。已经加载到内核中的模块会在 /sys/module/\$ {module name}/parameters/ 中显示出其参数，并且某些参数的值还可以在运行时通过“echo -n \$ {value} > /sys/module/\$ {module name}/parameters/\$ {param}”命令修改。

与设备有关的引导参数多如牛毛，需要你自己阅读内核中的相应驱动程序源码以获取其能够接受的引导参数。比如，如果你想知道可以向 AHA1542 SCSI 驱动程序传递哪些引导参数，那么就到 drivers/scsi 目录下寻找到 aha1542.c 文件，一般在前面 100 行注释里就可以找到所接受的引导参数说明。大多数参数是通过“__setup(.., ..)”函数设置的，逗号前的部分就是引导参数的名称，后面的部分就是处理这些参数的函数名。[提示]你可以在源码树的根目录下试一试

```
grep -r '\b__setup *(' *
```

命令。

[注意]多个参数之间用空格分割，而每个参数的值中不能包含空白，参数值是一个逗号分割的列表。

正确：ether=9,0x300,0xd0000,0xd4000,eth0 root=/dev/hda1
错误：ether = 9, 0x300, 0xd0000, 0xd4000, eth0 root = /dev/hda1

在内核运行起来之后，可以通过 cat /proc/cmdline 命令查看当初使用的引导参数以及相应的值。

所有引导参数都是大小写敏感的！

内核如何处理引导参数

绝大部分的内核引导参数的格式如下(每个参数的值列表中最多只能有十项)：

```
name[=value_1][,value_2]...[,value_10]
```

如果“name”不能被识别并且满足“name=value”的格式，那么则被解译为一个环境变量(比如“TERM=linux”或“BOOT_IMAGE=vm linux.bak”)，否则将被原封不动的传递给 init 程序(比如“single”)。

内核可以接受的参数个数没有限制，但是整个命令行的总长度(参数/值/空格全部包含在内)却是有限制的，一般是 256-4096 之间，定义在 include/asm/setup.h 中的 COMMAND_LINE_SIZE 宏中。

内核引导参数精选

由于引导参数多如牛毛，本文不可能涉及全部，因此下面只列出精选出来的一些(基于 2.6.22 内核)，与设备有关的基本上都被忽略了。

标记说明

并不是所有的参数都是永远可用的，只有在特定的模块存在并且相应的硬件也存在的情况下才可用。引导参数上面的方括号说明了其依赖关系，其中使用的标记解释如下：

| | |
|---------|---|
| ACPI | 高级配置与电源接口 |
| APIC | 高级可编程中断控制器 |
| HW | 相应的硬件设备存在 |
| IA-32 | IA-32(i386)体系结构 |
| X86-64 | X86-64 体系结构，更多参数在 Documentation/x86_64/boot-options.txt 中描述 |
| IOSCHED | 启用了多个 IO 调度器 |
| LIBATA | 启用了 Libata 驱动 |
| LOOP | 启用了 Loopback 设备 |
| NET | 启用了网络支持 |
| PCI | PCI 总线支持 |
| PNP | 即插即用支持 |
| PS2 | PS/2 支持 |
| SCSI | 许多 SCSI 设备的参数在 Documentation/scsi/ 中描述 |
| SMP | 对称多处理器 |
| USB | USB 支持 |
| USBHID | USB 人机界面设备 |
| VT | 虚拟终端(Virtual terminal) |

此外，下面的标记的含义与在逻辑上与上面的有所不同：

| | |
|-------|---------------------------|
| BUGS= | 用于在特定的体系结构上解决某些 CPU 的 bug |
| KNL | 是一个内核启动参数 |
| BOOT | 是一个引导程序参数 |

标记为“B O O T”的参数实际上由引导程序使用，对内核本身没有直接的意义。没有特别的需求，请不要修改此类参数的语法，更多信息请阅读 Documentation/i386/boot.txt 文档。

控制台

这些参数控制着控制台或内核日志，在何处显示内核调试信息和错误信息。

[KNL]
console= tty< N>
 设置输出控制台使用第 N 号虚拟控制台。
[IA-32,X86-64]
earlyprintk= vga
 在传统的控制台初始化之前，在 V G A 上显示内核日志信息。如果不使用此参数那么这些信息你可用永远没机会看见。
loglevel= {0|1|2|3|4|5|6|7}
 所有小于该数字的内核信息都将在控制台上显示出来。这个级别可以使用 klogd 程序或者修改 /proc/sys/kernel/printk 文件进行调整。取值范围是“0”(不显示任何信息)到“7”(显示所有级别的信息)。建议至少设为“4”。[提示]级别“7”要求编译时加入了调试支持。
[KNL]
initcall_debug
 跟踪所有内核初始化过程中调用的函数。有助于诊断内核在启动过程中死在了那个函数上面。

中断

这些参数影响内核与处理中断的硬件之间的接口。常见的中断控制器有两种：传统的 8259A 和新式的 APIC，前者也被称为“PIC”。8259A 只 适合单 CPU 的场合，而 APIC 则能够把中断传递给系统中的每个 CPU，从而充分挖掘 S M P 体系结构的并行性。所以 8259A 已经被淘汰了。

APIC 系统由 3 部分组成：APIC 总线、IO-APIC、本地 APIC。

每个 CPU 中集成了一个本地 APIC，负责传递中断信号到处理器。而 IO-APIC 是系统芯片组中一部分，负责收集来自 I/O 设备的中断信号并发送到本地 APIC。APIC 总线则是连接 IO-APIC 和各个本地 APIC 的桥梁。

[APIC,i386]
apic= {quiet|verbose|debug}
 在初始化 APIC 和 IO-APIC 组件的时候，显示调试信息的详细程度。默认是“quiet”。
[SMP,APIC]
noapic
 强制内核禁止使用 IO-APIC (输入输出高级可编程输入控制器)
[IA-32,APIC]
lapic
 强制内核启用 localAPIC，即使 BIOS 已经禁用了。
[IA-32,APIC]

`no lapic`
强制内核禁用 localAPIC，即使 BIOS 已经启用了。
[IA-32,SMP,KNL]
`no irqbalance`
禁止使用内核中的中断平衡逻辑
[HW]
`irqfixup`
用于修复基本的中断问题：当一个中断没有被处理时搜索所有可用的中断处理器。用于解决某些 firmware 缺陷。
[HW]
`irqpoll`
用于修复更进一步的中断问题：当一个中断没有被处理时搜索所有可用的中断处理器，并且对每个时钟中断都进行搜索。用于解决某些严重的 firmware 缺陷。
[IA-32]
`no irqdebug`
默认情况下，内核将探测并且禁止未处理的中断源，以免引起内核其他部分的响应问题，这个选项禁止该功能。

内存

[KNL,BOOT]
`highmem=nn[KMG]`
强制指定 highmem 区域的大小，甚至在默认没有 highmem 的机器上也能工作。这个选项还可以用来在大内存的机器上强制减少 highmem 区域的大小。内核使用低于 896M 的“直接映射物理内存”很方便，但使用大于 896M 的部分 (highmem) 却比较麻烦，所以系统在给用户进程分配内存时会优先使用 highmem。对于小于等于 1G 内存的用户来说，则无需关心这个问题。
[HW,IA-32]
`hugepages=<NUM>`
指定 HugeTLB 页的最大数量，仅在内核启用了 CONFIG_HUGETLBFS 之后才有效。大多数现代计算机体系结构提供对多页面大小的支持，比如 IA-32 结构支持 4K 和 4M (PAE 模式为 2M) 两种页面。因此 Linux 将物理内存划分成许多固定大小的页面 (默认大小为 4k)，每个页对应一个 page 结构，这些结构组成一个 mem_map 数组。TLB (Translation Lookaside Buffer) 是虚拟地址到物理地址的翻译缓冲区，这种缓冲区在处理器上是很宝贵的，操作系统总是尝试将有限的 TLB 资源发挥到极致。特别是能够轻松获得 若干 G 内存的时候 (≥ 4G)，这种优化就显得尤为关键。而 HugeTLB 特性则允许将某些页的尺寸增大到 4MB。用户可以使用 memmap 系统调用或者标准的 SYSv 共享内存调用 (shmget, shmatt) 来使用 hugepage。可以使用 `grep Huge /proc/meminfo` 命令查看是否开启了 hugepage 支持。
[KNL]
`ihash_entries=<NUM>`
内核会在内存中缓存一定数量的 inode 结构来加速文件访问，每个 inode 对应一个文件 (不同于文件系统中的 inode 概念)，包含文件访问权限/属主/组/大小/生成时间/访问时间/最后修改时间等信息。这些 inode 保存在一个哈希表中。这个值用于指定这个哈希表的最大项数。比如 1G 内存推荐为 16384，4G 及以上内存推荐 131072，但你可以根据自己硬盘上可能被访问的文件数量对默认值进行调整 (注意需要考虑哈希值的碰撞)。
[KNL,BOOT]
`max_addr=nn[KMG]`
内核将忽略在该物理地址以上的内存
[KNL,BOOT]
`mem=nn[KMG]`
强制指定内核使用多少数量的内存。缺乏远见设计的传统 BIOS 只能报告最大 64MB 内存。新的 e820 规范则突破了这个限制，使得 BIOS 可以正确报告大于 64MB 的内存。如果你在老旧的机器上使用大内存就需要指定这个参数 (最保险的做法是在实际内存的总数上减掉 1MB)。但有时候 e820 报告的数量并不准确，此时就需要使用下面的 memmap 参数精确指定内存映射 (此时就不要使用 “mem=” 了)。
[KNL,IA-32,X86_64]
`memmap=exactmap`
指定将要使用随后的 “memmap=nn@ss” 等参数进行精确的 E820 内存映射。比如对于一个 4G 内存的机器可能是：
“memmap=exactmap memmap=640K@0 memmap=4095M@1M”。
[KNL]
`memmap=nn[KMG]@ss[KMG]`
强制内核只使用从 ss 开始的 nn 长度的特定内存区域。可以多次使用以指定多个区域。
[IA-32,X86-64]
`noexec={on|off}`
允许 (on, 默认) 或禁止 (off) 内核将部分内存映射为 “不可执行” 区域。
[KNL,BUGS]
`reserve=nn[KMG]`
强制内核忽略 (预留) 一定量的 ID 内存区域
[KNL,BOOT]
`vmalloc=nn[KMG]`
强制指定 vmalloc 区域的大小。可用于增加 vmalloc 区域的最小尺寸 (x86 默认 128MB)，也可以用于减少 vmalloc 的大小，增加更多的空间用于直接映射内核 RAM。
`norandmaps`
默认内核随机化程序启动的地址，该选项禁用该功能。该选项等价于 “echo 0 > /proc/sys/kernel/randomize_va_space” 命令。

CPU

[BUGS=IA-32]
`cachesize=<NUM>`
强制指定 CPU L2 cache 的大小，单位是字节。

[KNL,BUGS=IA-32]
nm_i_watchdog= {0|1|2|3}
设置非屏蔽中断(NM I watchdog)的特性。“0”表示禁用NM I watchdog；“1”表示使用APIC；“2”表示使用localAPIC；“3”表示NM I watchdog有缺陷，因此被禁用。
[IA-32]
mce
nomce
启用/禁用Machine Check Exception功能。
[SMP]
maxcpus=<NUM>
明确指定一个SMP内核能够使用的最大CPU数量。最好使用“maxcpus=0”而不是“maxcpus=1”来禁用SMP功能。

Ram disk

[BOOT]
initrd=<filename>
指定initialram disk的位置
[RAM]
noinitrd
禁止使用任何initialRAM disk
[RAM]
ramdisk_blocksize=<NUM>
指定ram disk的块尺寸，默认是“1024”。
[RAM]
ramdisk_size=<NUM>
RAM disks的大小(kB)，默认为4096 (4MB)。

根文件系统

[KNL]
root=XXxx
告诉核心启动时以那个设备作为根文件系统使用，默认为编译内核时使用的设备。设备名由16进制主设备号(XX)与16进制次设备号(xx)组成，比如：root= B401 相当于从 /dev/uba1 启动；root= 0801 相当于从 /dev/sda1 启动。
[KNL]
rootdelay=<NUM>
挂载文件系统前延迟多少秒，当根文件系统在USB或FireWire设备上时常用。
[KNL]
rootflags=
设置根文件系统的挂载选项，比如“noatime,ro”。各种不同的文件系统所能使用的选项各不相同(比如Documentation/filesystems/xfs.txt)，也可以参考mount程序的选项。
[KNL]
rootfstype=
根文件系统的类型，比如“xfs”。

init

[KNL]
init=<full_path>
指定内核启动后运行的第一个程序的绝对路径。默认为“/sbin/init”。
[KNL]
rdinit=<full_path>
从ram disk中运行的第一个程序的绝对路径，默认为“/init”。指定的文件必须是在ram disk而不是在根文件系统中进行。
[KNL]
S
以单用户模式运行init，默认是多用户模式。

ACPI

[HW,ACPI,X86-64,i386]
acpi= {force|off|ht|strict|noirq}
ACPI的总开关。force表示强制启用；off表示强制禁用；noirq表示不要将ACPI用于IRQ路由；ht表示只运行足够的ACPI来支持超线程；strict表示降低对不严格遵循ACPI规格的平台兼容性。
acpi_sleep= {s3_bios,s3_mode}
ACPI休眠选项。在从S3状态(挂起到内存)恢复的时候，硬件需要被正确的初始化。这对大多数硬件都不成问题，除了显卡之外，因为显卡是由BIOS初始化的，内核无法获取必要的恢复信息(仅存在于BIOS中，内核无法读取)。这个选项允许内核以两种方式尝试使用ACPI子系统来恢复显卡的状态。
[HW,ACPI]
acpi_sci= {level|edge|high|low}
ACPI系统控制终端触发器模式(System Control Interrupt trigger mode)。
[HW,ACPI]
acpi_irq_balance
使ACPI对中断请求进行平衡，在APIC模式下为默认值
[HW,ACPI]

```
acpi_irq_nobalance
    ACPI不对中断请求进行平衡(默认), PIC 模式下为默认值
[HW,ACPI]
acpi_irq_pci=< irq>,< irq>...
    如果启用了 irq_balance 则将列出的中断号标记为已经被 PCI子系统使用, 可用于屏蔽某些中断。
[HW,ACPI]
acpi_os_name=
    告诉 ACPI BIOS 操作系统的名称。常常用来哄骗某些老旧的 BIOS 以为运行的是 Windows 系统。比如“Microsoft 2001”表示
    WinXP, “Microsoft Windows”表示 Win98。
[HW,ACPI]
acpi_serialize
    强制串行化 ACPI 机器语言(ACPI Machine Language)方法, 操作系统使用这种语言与 BIOS 打交道。
[HW,ACPI]
acpi_use_timer_override
    对于某些有毛病的 Nvidia NF5 主板需要使用此选项才能正常使用, 不过此时 HPET 将失效。
[IA-32,X86-64]
acpi_pm_good
    跳过 pm timer 的 bug 检测, 强制内核假设这台机器的 pm timer 没有毛病。用于解决某些有缺陷的 BIOS。
[KNL,ACPI]
memmap=nn[KMG]#ss[KMG]
    将从 ss 开始的 nn 长度的内存区域标记为 ACPI 数据。
[KNL,ACPI]
memmap=nn[KMG]$ss[KMG]
    将从 ss 开始的 nn 长度的内存区域标记为“保留”。
[ACPI]
pnpcpi=off
    禁用 ACPI 的即插即用功能, 而使用 PNP BIOS 来代替。
[HW,ACPI]
processor.max_cstate= {0|1|2|3|4|5|6|7|8|9}
    无视 ACPI 表报告的值, 强制制定 CPU 的最大 C-state 值。这里的数字必须是一个有效的 C-state 值, 比如 Yonah 处理器支持“
    0-4”五个级别: C0 为正常状态, 其他则为不同的省电模式(数字越大表示 CPU 休眠的程度越深/越省电)。“9”表示超越所有的 DMI
    黑名单限制。你的 CPU 的 95% 的时间应该处于最深度的 idle 状态。
processor.no_cst
    不使用_CST 方法来侦测 C-state 值, 而是使用传统的 FADT 方法。
```

SCSI

这里只列出了 SCSI 子系统的通用参数。其他特定于某一种 SCSI 驱动的参数并未列出, 请在 Documentation 目录下的 kernel-parameters.txt 文件中和 scsi 目录下寻找它们。

```
[SCSI]
max_luns=
    限制 SCSI 的最大逻辑单元号(LUN, logical unit number)。取值范围在 1 到 2^32-1 之间。
[SCSI]
max_report_luns=
    限制系统能够接受的最大逻辑单元号(LUN)。取值范围在 1 到 16384 之间。
```

PCI

```
[PCI]
pci_option[option...]
    off
        [IA-32]不检测 PCI 总线, 也就是关闭所有 PCI 设备。
    bios
        [IA-32]强制使用 PCI BIOS 而不是直接访问硬件, 这表示内核完全信任 BIOS(大多数情况下它并不可信)。仅在你的机器有
        一个不标准的 PCI host bridge 的时候才用。
    nobios
        [IA-32]强制直接访问硬件而不使用 PCI BIOS, 2.6.13 之后这是默认值。如果你确定在内核引导时的崩溃是由 BIOS 所致就
        可以使用它。
    conf1
        [IA-32]强制硬件设备使用 PCI Configuration Mechanism 1 访问 PCI Memory 以与内核中的驱动程序进行通信。
    conf2
        [IA-32]强制硬件设备使用 PCI Configuration Mechanism 2 访问 PCI Memory 以与内核中的驱动程序进行通信。
    nommconf
        [IA-32,X86_64]禁止为 PCI Configuration 使用 MMCONFIG 表。
    nomsi
        [MSI]如果启用了 PCI MSI 内核配置选项, 那么可以使用这个参数在系统范围内禁用 MSI 中断。
    nosort
        [IA-32]不在检测阶段根据 PCI BIOS 给出的顺序对 PCI 设备进行排序。进行这样的排序是为了以与早期内核兼容的方式获取
        设备序号。
    biosirq
        [IA-32]使用 PCI BIOS 调用来获取中断路由表。这些调用在不少机器上都有缺陷, 会导致系统在使用过程中挂起。但是在某
        些机器上却是唯一获取中断路由表的手段。如果内核无法分配 IRQ 或者发现了第二个 PCI 总线, 就可以尝试使用这个选项
```

解决问题。

rom
[IA-32]为扩展 ROM 分配地址空间。使用此选项要小心，因为某些设备在 ROM 与其它资源之间共享地址解码器。

pirqaddr= 0xA A A A A
[IA-32]指定物理地址位于 F0000h-100000h 范围之外的 PIRQ 表 (通常由 BIOS 产生)的物理地址。

lastbus= N
[IA-32]扫描所有总线，直到第 N 个总线。如果内核找不到第二条总线的时候，你就需要使用这个选项明确告诉它。

assign-busses
[IA-32]总是使用你自己指定的 PCI总线号 (而不是 firmware 提供的)。

usepirqmask
[IA-32]优先使用可能存在于 BIOS \$PIR 表中的 IRQ 掩码。某些有缺陷的 BIOS 需要这个选项，特别是在 HP Pavilion N5400 和 OmniBook XE3 笔记本上。如果启用了 ACPIIRQ 路由的话，将不会考虑这个选项的设置。

noacpi
[IA-32]不为 IRQ 路由或者 PCI扫描使用 ACPI。

routeirq
为所有 PCI设备执行 IRQ 路由。这个通常在 pci_enable_device()中执行，所有这是一个解决不调用此函数的 bug 驱动程序的临时解决方法。

bfsort
按照宽度优先的顺序对 PCI设备进行排序。进行这样的排序是为了以与 2.4 内核兼容的方式获取设备序号。

nobfsort
不按照宽度优先的顺序对 PCI设备进行排序。

cbiosize= nn [KM G]
从 CardBus bridge 的 IO 窗口接受的固定长度的总线空间 (bus space)，默认值是 256 字节。

cbmemsize= nn [KM G]
从 CardBus bridge 的 memory 窗口接受的固定长度的总线空间 (bus space)，默认值是 64MB。

网络

[NET]
netdev= < irq>,< io>,< mem_start>,< mem_end>,< name>
网络设备参数。具体细节取决于不同的驱动程序，请参考各自的驱动程序文档。该选项通常不用于 PCI/USB 等即插即用网卡。

[KNL,NET]
rhash_entries=
设置内核路由缓冲区哈希表的大小，仅供内核网络专家使用。

[NET]
shapers=
设置内核允许使用的最大网络 Shaper(限制网络速率的虚拟网络设备)。

[KNL,NET]
thash_entries=
设置内核允许使用的 TCP 链接哈希表的大小。

硬件

[USB]
noub
禁用 USB 子系统。

定时器

[386,x86-64]
enable_timer_pin_1
disable_timer_pin_1
启用/禁用 APIC 定时器的 PN1，可以用于解决某些有 bug 的芯片组 (特别是 ATI 芯片组)。内核将尽可能自动探测正确的值。

[IA32/X86_64]
enable_8254_timer
disable_8254_timer
启用/禁用 在通过 IO-APIC 对 IRQ0 (时钟中断)进行路由之外，还通过 8254 进行路由。内核将尽可能通过检测来选择正确的值。

[IA-32,HPET]
hpet= disable
禁用 HPET，转而使用 PIT。

[GENERATOR_TIME,IA-32,X86-64,ACPI]
clocksource= {hpet|pit|tsc|acpi_pm}
强制指定 clocksource 来取代默认值。

其他杂项

[VT]
default_utf8= {0|1}
在系统范围内为将所有 tty 默认设置为 UTF-8 模式。“1”表示 UTF-8 模式，默认值为“0”。

[IO SCHED]
elevator= {“anticipatory”|“cfq”|“deadline”|“noop”}
指定默认 IO 调度器

[LOOP]

max_loop=< 1-256>
最大允许挂载的 loopback 设备数。

[KNL]

panic=< seconds>
在内核发生 panic 之后 reboot 之前等候的秒数。默认值“0”表示不重启而停顿在那里。