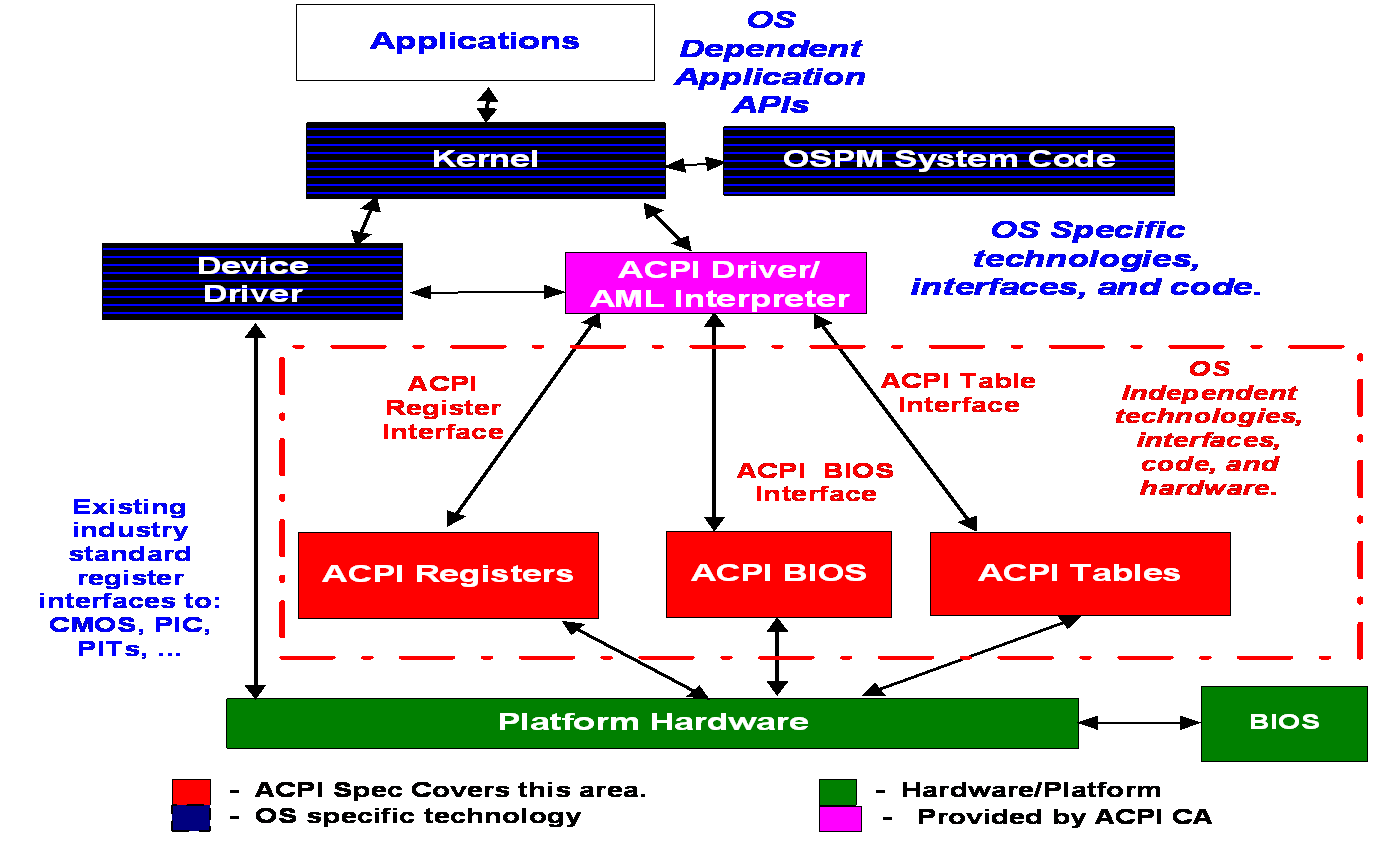
**ACPI简介\_Linux**

ACPI – the Advanced Configuration & Power Interface. ACPI是OS，BIOS和硬件之间的抽象层。它允许OS和平台独立的发展，比如新的OS可以控制老的平台，老的OS也可以控制新的平台而不需要额外的修改。ACPI的基本结构图如下：



计算机领域的一个基本方法是增加一个抽象层，从而使得抽象层的上下两层独立的发展，ACPI事实上也是借鉴类似的思想。ACPI的抽象主要是通过ACPI表中提供的信息来实现，这些信息包括ACPI寄存器、AML代码、配置信息等。

* ACPI寄存器，描述了和ACPI相关的寄存器。OS可以直接从ACPI表中取得这些寄存器的信息，因此不必了解具体的硬件配置。
* AML代码，一种类C的代码。由BIOS提供，OS的AML解释器（在Linux中叫ACPI CA）解释执行这些代码。这是ACPI抽象层的关键，下面会详细介绍。
* 配置信息，ACPI包含的配置信息很多，比如多处理器配置信息（MADT）、NUMA配置信息（SRAT、SLIT）、高精度定时器（HPET）等。

AML代码是抽象的关键。为了消除平台相关性，BIOS把平台相关的操作用AML代码来实现。OS不需要知道平台细节，它只是解释执行这些代码，在解释执行的过程中平台相关的操作就被执行了。ACPI规范定义了一些标准的AML函数，OS解释执行一个这样的标准函数就可以实现特定的功能。举个例子：

Scope (\_SB.PCI0.LPC) { OperationRegion (LPCS, PCI\_Config, 0x00, 0x0100) Field (LPCS, AnyAcc, NoLock, Preserve) { Offset (0x60),                                            PIRA,   8, } } Device (LNKA) {                     Method (\_DIS, 0, NotSerialized) {                                           Or (\\_SB.PCI0.LPC.PIRA, 0x80, \\_SB.PCI0.LPC.PIRA) } }

比如我们想禁止LNKA设备，ACPI规范定义了OS必须执行此设备的\_DIS函数。从硬件的角度来讲，禁止LNKA设备需要将某个PCI设备的配置空间的寄存器0×60的最高位置上。OS不需要知道硬件的细节，它解释执行\_DIS函数即可。上面的代码就是AML代码，由BIOS提供，语句‘Or (\\_SB.PCI0.LPC.PIRA, 0×80, \\_SB.PCI0.LPC.PIRA)’实际上就是给寄存器\\_SB.PCI0.LPC.PIRA置上最高位， 而PIRA就是PCI设备LPC的配置空间的寄存器0×60。从此段代码我们可以明显的看出BIOS以AML代码的形式隐藏（抽象）了硬件的细节，从而使得OS看到的是一个平台无关的硬件。

更多的ACPI预定义的函数可以在ACPI规范中找到，可以在[http://www.acpi.info](http://www.acpi.info/" \o "http://www.acpi.info)下载最新的规范。

Linux/ACPI实现中使用的AML解释器是ACPICA －the ACPI Component Architecture. 可以从[http://www.intel.com/technology/iapc/acpi/downloads.htm](http://www.intel.com/technology/iapc/acpi/downloads.htm" \o "http://www.intel.com/technology/iapc/acpi/downloads.htm)得到。它包含一个AML解释器（Linux kernel包含了这个解释器，很多其他OS也是如此，比如BSD、opensolaris等），一个编译器（将ACPI Source Language （ASL）编译成AML代码）和一些测试工具。

最新的Linux/ACPI的代码可以使用git在[http://www.kernel.org/git/?p=linux/kernel/git/lenb/linux-acpi-2.6.git](http://www.kernel.org/git/?p=linux/kernel/git/lenb/linux-acpi-2.6.git" \o "http://www.kernel.org/git/?p=linux/kernel/git/lenb/linux-acpi-2.6.git)得到。

## 为什么学习ACPI

ACPI是Intel（i386，x86\_64，IA64）平台的标准固件规范，绝大部分OS需要从BIOS得到的信息都可以从ACPI得到，并且现在的趋势是未来的任何新的特性相关的信息都只能从ACPI得到。ACPI的内容庞杂，学习ACPI至少可以帮助我们理解：

* 配置信息。这些信息从legacy PNP设备的配置，到多处理器，到NUMA，比如现在的Multiple Core的信息就只能从ACPI得到。Linux启动很多代码就是处理这些配置信息，比如APIC，IOAPIC设置等。
* ACPI相关设备。主要是笔记本电脑相关的设备，包括电源按钮，电池，外接电源，风扇，热键等。
* 底层硬件。比如PCI中断路由，chipset（主要是南桥PCI-to-LPC bridge)操作等。
* 电源管理。ACPI定义的电源管理包括CPU的电源管理（调频率P-state，idle C-state，throtting T-state），设备电源管理(D-state)，系统电源管理（Suspend-to-Ram, Suspend-to-Disk, power off）等。
* 设备热插拔。ACPI用一种统一的方式来描述设备的热插拔，这样的设备从单一的PCI设备，到笔记本电脑的Docking Station，到整个PCI hierarchy，CPU，Memory，甚至整个NUMA节点。

可以说要理解现代PC平台必须了解ACPI。

# 解决ACPI问题的小窍门

首先可以看看是否这是一个regression，如果以前版本的Linux kernel可以工作，但新的不行，则是一个regression，可以测试不同的kernel从而找出哪个版本引入了bug。使用git-bisect是一个好的选择，它可以帮你定位到哪个patch导致了regression。一些git-bisect相关的资料如下： <http://www.stardust.webpages.pl/files/handbook/handbook-en.pdf> <http://www.kernel.org/pub/software/scm/git/docs/tutorial.html> <http://www.kernel.org/pub/software/scm/git/docs/git-bisect.html>

## 系统不能启动

试试kernel参数”acpi=off”，如果此参数没有带来任何改变，那么这不是一个ACPI bug。反之，则这很可能是一个ACPI bug。 确定了是一个ACPI bug后，我们还有其他一些参数来更进一步的区分到底是ACPI哪个部分的bug。

acpi=ht

这个参数和"acpi=off"几乎一样，它禁止了除多处理器配置相关的内容以外的所有ACPI功能。如果acpi=off正常，但acpi=ht 不正常, 则解析ACPI 表或者Linux SMP的代码有bug.

pci=noacpi

禁止使用ACPI来处理任何PCI相关的内容，包括PCI root bus的枚举和PCI设备中断路由。

acpi=noirq

禁止使用ACPI来处理PCI设备中断路由，和pci=noacpi的区别是它允许使用ACPI来枚举PCI root bus.

pnpacpi=off

禁止使用ACPI来枚举PNP设备，比如串口、PS2键盘鼠标等。

noapic

禁止使用io-apic来做设备中断路由，这样做的效果之一是ACPI返回的中断路由表将是针对PIC（8259）的。

nolapic

禁止使用Local-APIC和IO-APIC。

## 设备中断相关的问题

出现中断问题的可能性很多，比如驱动程序有bug。由ACPI导致的最常见的中断问题是kernel打出：”irqXX: nobody cared!”。这意味着kernel收到一个中断，但是没有驱动程序来处理此中断。Kernel会将此中断禁止，从而导致挂在此中断上的所有设备都停止工作。pci=noacpi, acpi=noirq, pnpacpi=off, noapic, nolapic这些参数可以帮助隔离一些问题。另外一个有用的参数是”irqpoll”，出现上面的中断问题时，它可以使kernel自动探测哪个设备发出了中断。这个参数对于调试那些中断路由有问题的系统很有用。

## Suspend to RAM问题

STR的一个常见问题是Resume回来后黑屏，但是系统并没有死掉，比如可以通过网络访问系统或者键盘灯工作正常。可以试试kernel参数acpi\_sleep=s3\_bios/s3\_mode，它会尝试将显示器打开。如果不行可以试试vbetools（<http://www.srcf.ucam.org/~mjg59/vbetool>），在resume回来后输入 $ vbetool post 为了方便，你可以在你的STR脚本中调用此命令。

STR的另一个常见问题是系统没法resume回来，你可以试试acpi\_sleep=s3\_beep。如果你听不到电脑的扬声器产生的声音，那么resume的代码完全没被执行。这可能是BIOS的原因，也可能是Linux的原因，目前还没有太好的办法处理。反之，很有可能是Linux driver的原因，你可以尝试尽可能少的加载驱动程序，只保留最基本的驱动，比如硬盘驱动。

## Suspend to Disk 问题

TBD

## ACPI debug参数

参数是acpi.debug\_level and acpi.debug\_layer。如果打开debug参数，ACPI可以产生很多详细的运行输出。这些输出可以帮助我们定位出错的原因。

对于debug\_layer和debug\_level，include/acpi/acoutput.h里面包含了很多值，这些值决定了Linux/ACPI输出信息的详细程度和内容范围。acpi.debug\_level和 acpi.debug\_layer是kernel参数，也可以在系统运行时改变这些值，它们是/sys/module/acpi/parameters/debug\_{level,layer}。

注意，这些输出信息可能很快就将kernel的ring buffer用完，你可能需要使用log\_buf\_len=XY来增加ring buffer的大小。使用serial console (Documentation/serial-console.txt)来得到kernel输出是一个好的方法。如果你的笔记本电脑没有串口，可以试试netconsole (Documentation/networking/netconsole.txt)。

## 使用定制的 DSDT

DSDT (Differentiated System Description Table)是一个主要的ACPI表，它包含了很多AML代码。因为BIOS的bug，这些代码本身可能有错。Linux提供的一种方法能让你使用定制的DSDT表，这对于调试很有帮助。让kernel使用定制的DSDT步骤如下：

    首先要得到原始的DSDT表（后面的章节会介绍acpidump等工具）:     $ acpidump > acpidump.out     $ acpixtract DSDT acpidump > DSDT.dat     这样我们就得到了DSDT表的二进制文件，将它反汇编     $ iasl -d DSDT.dat     我们会得到一个AML代码文件，你可以修改它     $ vi DSDT.dsl     然后重新编译     $ iasl -tc DSDT.dsl     把它拷贝到kernel source中     $ cp DSDT.hex $SRC/include/

加入下面几行到你的kernel配置文件（.config）:

CONFIG\_STANDALONE=n CONFIG\_ACPI\_CUSTOM\_DSDT=y CONFIG\_ACPI\_CUSTOM\_DSDT\_FILE=”DSDT.hex”

编译kernel，运行，你的dmesg中应该有如下输出： Table [DSDT] replaced by host OS

使用这种方法，你可以修正DSDT的bug。这种方法带来的一个有用的debug方法是：将ACPI的debug选项打开，然后在你的DSDT中加入类似如下的语句： Store(”hello world!”, Debug) Store(Local0, Debug) 即将某个变量存储到特殊的目标Debug中。加入了这样语句后的函数被kernel解释执行时你可以看到如下输出： [ACPI Debug] String: [0x0C] “hello world!” [ACPI Debug] Integer: 0×00000042 由此我们可以在AML代码级别进行调试。

# 报告ACPI bug

Linux/ACPI社区使用kernel bugzilla来跟踪bug 。[http://bugzilla.kernel.org/enter\_bug.cgi?product=ACPI](http://bugzilla.kernel.org/enter_bug.cgi?product=ACPI" \o "http://bugzilla.kernel.org/enter_bug.cgi?product=ACPI)。这个网站主要是跟踪base kernel的bug，如果你有特定发行版的bug，不要发到这个网站。Linux/ACPI有自己的邮件列表（[linux-acpi@vger.kernel.org](mailto:linux-acpi@vger.kernel.org" \o "linux-acpi@vger.kernel.org)），你也可以在那里讨论问题。另外，Intel的Linux/ACPI组也有一个邮件地址（[acpi@linux.intel.com](mailto:acpi@linux.intel.com" \o "acpi@linux.intel.com)），如果你的问题不方便公开，可以发到这儿。

如果你报告一个bug，请提交如下信息：

1. 产生bug的kernel版本
2. 以前的kernel有没有这样的bug。如果这是一个regression，最近可以工作的kernel版本是什么。如果你能使用git-bisect找到哪个patch带来regression，那问题基本上就等于解决了。
3. 出错的kernel和最近工作kernel的dmesg信息。你可能需要使用serial console来得到这些信息。
4. 如果这是中断相关的问题，可能的话请提供kernel出错和工作的时候/proc/interrupts的输出。/sbin/lspci –vvv和/sbin/lspci -xxx的输出也很有用。
5. 请提供acpidump的输出。Acpidump是一个工具，它可以将系统中的ACPI表打出来。你可以在<http://ftp.kernel.org/pub/linux/kernel/people/lenb/acpi/utils/>找到这个工具。注意acpidump输出的是BIOS的表，不同的BIOS版本可能会有不同的表。
6. 如果我们发现BIOS有问题，我们可以将此系统列入黑名单，在这种情况下需要提供dmidecode（通常在/usr/sbin/下）工具的输出。
7. 产生bug的kernel配置文件

## 如何使用ACPI工具

<http://ftp.kernel.org/pub/linux/kernel/people/lenb/acpi/utils/>，按照包里面的README编译。使用步骤如下：导出所有的表，这些表都是二进制的 $ acpidump > acpidump.out

上面的输出包含了很多个ACPI表，如果你希望将它们分离开，使用 $ acpixtract -a acpidump.out

反汇编某个表 $ iasl -d TABLE.dat 这样就得到类C的AML代码。

# 参考文档

ACPI in Linux – Myths vs. Reality(OLS 2007) paper: <https://ols2006.108.redhat.com/2007/Reprints/brown_1-Reprint.pdf> presentation: <http://ftp.kernel.org/pub/linux/kernel/people/lenb/acpi/doc/OLS2007-acpi-myths-web/>

ACPI in Linux – Architecture, Advances, and Challenges(OLS 2005) paper: <http://www.linuxsymposium.org/2005/linuxsymposium_procv1.pdf> presentation: <http://ftp.kernel.org/pub/linux/kernel/people/lenb/acpi/doc/ACPI_OLS_2005.pdf>

The State of ACPI in the Linux Kernel(OLS 2004) <http://ftp.kernel.org/pub/linux/kernel/people/lenb/acpi/doc/Reprint-Brown-OLS2004.pdf>

# TODO列表

1. Suspend/resume的稳定性。Suspend-to-ram在很多笔记本电脑上不能工作。很多驱动程序没有实现.suspend/.resume方法或者实现有问题。
2. Hotkey的支持。很多笔记本电脑厂商使用完全不同的方法来支持hotkey，现在Linux支持IBM，Asus，Toshiba等。但是还有很多厂商的不支持，即使支持的厂商也有很多笔记本型号不支持。
3. 运行时设备电源管理。Linux还缺乏一个框架在系统运行时对设备进行电源管理，例如在某个设备空闲时将它关闭而不影响整个系统的运行。
4. Device model方面的改进。Linux仍然缺乏一个好的机制将ACPI设备和它对应的物理设备统一起来处理。
5. Bugzilla上有很多ACPI的bug