ACPI Implementation in Linux 2.6

The Small Sleeper

The new kernel has seen the continued advance of ACPI into the Linux mainstream. The standardized hardware configuration facilities should be familiar to most people from the kernel 2.4, but power management has now been enhanced with the addition of several sleep modes. BY TIMO HÖNIG

Esther Keller, visipix.com

What a success story. Almost any laptop, pre-configured PC, components or operating systems you look at claim to be ACPI-aware. Since the introduction of the first version of the Advanced Configuration and Power Interface way back in 1996, the specification has aimed to finally send off the APM (Advanced Power Management) standard and the inflexible plug & play BIOS into retirement.

ACPI is often seen merely as a replacement for APM. But that is not strictly true: the C for Configuration is an integral part of ACPI, which provides a uniform and (as far as this is feasible) operating system independent hardware setup interface. Linux 2.6 uses ACPI to route PCI interrupts. If you want to use the Advanced Programmable Interrupt Controller (APIC) on a single-CPU system, instead of the venerable PIC – to avoid resource conflicts or interrupt sharing – there is no alternative to ACPI. Also, the ACPI subsystem tells the kernel which PCI devices are hotplug capable, and how to configure these devices. The routines are to be found in *drivers/pci/hotplug/acpiphp_glue.c*.

ACPI uses the ACPI Source Language (ASL) and its compiled offshoot, the ACPI Machine Language to provide an abstraction layer for hardware-dependent functions. AML describes the hardware and the steps needed to access it. Every ACPI-compatible operating system has an AML interpreter for AML bytecode.

THE AUTHOR

Timo Hönig studies Computer Science in Augsburg, Germany. He has lived in the world of networks, desktops, Open Source and Unix for over ten years now. His current passion is an Open Source Project called FnFX, a Linux ACPI daemon for Toshiba laptops (http://fnfx.sourceforge.net).

Two-Layered Specification

The ACPI specification [1] divides the ACPI architecture into two layers. The first of them, low level, comprises the following ACPI architecture:

- ACPI tables
- ACPI BIOS
- ACPI registers

The ACPI tables describe the ACPI hardware and its configuration in AML. The central data structure of each ACPI system uses definition blocks to stipulate how to access the hardware. When a system boots, the ACPI BIOS stores the ACPI tables in memory. The ACPI BIOS is also involved in sleep and resume operations (Suspend-to-RAM, Suspend-to-Disk).

The second, or high, level is part of the operating system and uses the ACPI core and ACPI drivers to provide an API that the low level ACPI components can access. The AML interpreter is part of this level.

ACPI on Linux 2.6

The kernel configuration groups the ACPI power management functions [2] below *Power management options (ACPI, APM)* (see Figure 1). If you want to use Suspend-to-RAM , you will first need to select *Prompt for development and/or incomplete code/drivers* below *Code maturity level options* to display the *Sleep States* option with the other ACPI options.

On booting, the kernel calls *acpi_boot_init()* to access the ACPI tables and parses them via the ACPI BIOS. The Differentiated System Description Table

(DSDT) is of particular interest. It contains the specifications for ACPI-compatible hardware components and their configurations.

It is well-known that some manufacturers use faulty DSDTs on their systems. Even if the manufacturer does not provide a BIOS upgrade with an error-free DSDT, and the *Relaxed AML* kernel option (*CONFIG_ACPI_RELAXED_ AML*) is no help, don't panic. Debugged DSDTs for many systems with faulty DSDTs are available from [3]. The steps are described at [4].

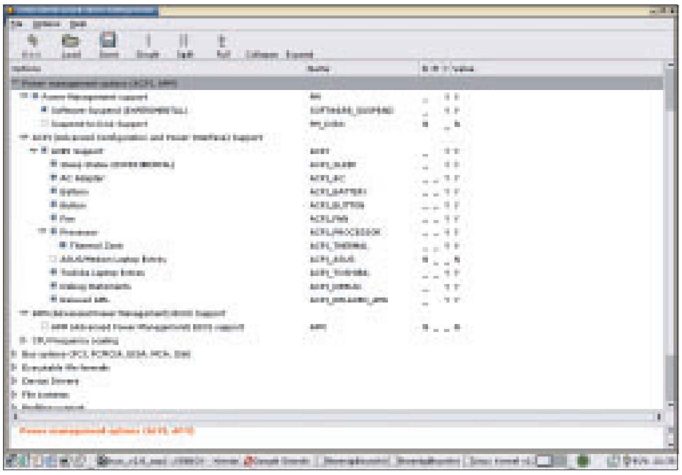

Figure 1: Kernel 2.6 options for ACPI-based power management.

If you are out of luck, and cannot locate an error-free DSDT for your system, a manual repair may be possible. Intel provides an ASL compiler and the AML disassembler *iasl* [5]. This allows you to copy the faulty DSDT from the Proc filesystem (*cat /proc/acpi/dsdt > dsdt.aml*), and disassemble it using the AML compiler. The *iasl -d dsdt.aml* command will disassemble the DSDT and output the ASL source to *dtds.dsl* (see Listing 1).

The next step is to locate the errors in the ASL source, remove them, and re-compile (*iasl -tc dtds.dsl*). Try to eliminate any warnings and errors, by modifying the ASL source (*dtds.dsl*), before using the re-compiled DSDT. Tips on patching faulty DSDTs are available at [6].

After booting Linux 2.6 with ACPI support, it makes sense to inspect the ACPI entries on the proc and sys filesystems. You can use the entries in */proc/acpi* and */sys* to access the ACPI interface. Root privileges are required to write to these locations. If you have not mounted *sysfs*, enter the following to do so:

```
mkdir /sys
mount -t sysfs sysfs /sys
```

The *sysfs /sys sysfs defaults 0 0* entry in the */etc/fstab* file ensures that sysfs will mount automatically when you boot.

Sleep States

The individual sleep states (see Table 1) can be manipulated via */sys/power/state*. *cat /sys/power/state* will tell you the states your system supports. *echo -n "Sleep_State" /sys/ power/state* sets the system to the specified state.

Suspend-to-RAM (S3) switches off the whole system with the exception of the main memory: that is, the CPU, cache memory, the chipset and the peripheral devices (hard disks, USB and so on). To prevent loss of memory content, power is supplied to the RAM. Suspend-to-RAM theoretically allows the system to go to sleep, and wake up again, within a few seconds. My experience is that Suspend-to-RAM rarely works. A good thing that the kernel still has it tagged as *experimental*.

Suspend-to-Disk (S4) saves more power than any other sleep state, as it stores the current memory content, the registers, and the states of peripheral devices on the hard disk (Linux uses a large swap partition to do this) before completely halting the system. On re-booting, Suspend-to-Disk reads the content of main memory, and the previous system state, from the hard disk, and restores the previous state. Linux 2.6 supports Suspend-to-Disk independently of ACPI and APM. For comparison's sake: Linux 2.4 does not support Suspend-to-RAM at all, although Suspend-to-Disk is possible, using the kernel patches provided by the Swsusp [7] project.

Software Suspend and Suspend-to-Disk

Confusingly, Linux 2.6 has two alternative Suspend-to-Disk (S4) methods: Software Suspend (*CONFIG_SOFTWARE_ SUSPEND*) and Suspend-to-Disk (*CONFIG_PM_ DISK*). The former is tagged as

Listing 1: Disassembling a DSDT with *iasl*

```
[root@sunshine:~]$ cp /proc/acpi/dsdt ~/dsdt
[root@sunshine:~]$ iasl -d ./dsdt
Intel ACPI Component Architecture
ASL Optimizing Compiler / AML Disassembler version 20030918 [Sep 18 2003]
Copyright (C) 2000 - 2003 Intel Corporation
Supports ACPI Specification Revision 2.0b

Loading Acpi table from file dsdt
Acpi table [DSDT] successfully installed and loaded
Pass 1 parse of [DSDT]
Pass 2 parse of [DSDT]
Parsing Deferred Opcodes (Methods/Buffers/Packages/Regions)
..............................................
Parsing completed
Disassembly completed, written to "dsdt.dsl"
[root@sunshine:~]$ head dsdt.dsl
/*
 * Intel ACPI Component Architecture
 * AML Disassembler version 20030918
 *
 * Disassembly of dsdt, Thu Dec 11 17:28:16 2003
 */
DefinitionBlock ("DSDT.aml", "DSDT", 1, "TOSHIB", "2000 ", 537003284)
{
    Name (\_S0, Package (0x04)
    {
[root@sunshine:~]$
```

experimental. In contrast, Suspend-to-Disk support is a stable branch of Suspend-to-Disk.

Software Suspend assumes a *resume* boot prompt parameter – it is not possible to compile this into the kernel. The option tells the kernel the swap partition it should use to store the memory content and status information. It makes sense to create a permanent boot prompt parameter in the boot manager configuration file, for example *resume=/dev/hda2*.

The alternative Suspend-to-Disk support provided by Software Suspend allows you to specify the swap partition in the kernel parameter *pmdisk*. In contrast to *resume*, you can bind *pmdisk* to the kernel.

Table 1: ACPI Sleep States in Linux 2.6

ACPI Sleep State	Sleep_State off	Description
Standby (S1)	*standby*	System empowered, no processing by CPU.
Suspend-to-RAM (S3)	*mem*	System switched off except for main memory.
Suspend-to-Disk (S4)	*disk*	System switched off, main memory and status are saved to disk.

Note: States S0 (Running) and S5 (Soft Off) are not ACPI sleep states.

Insomnia

Suspend-to-RAM and Suspend-to-Disk both assume that the current drivers are suspend and resume-aware. The driver model was modified to accomplish this on Linux 2.6. Device drivers are required to provide *suspend* and *resume* functions that the kernel will call at the beginning or end of a sleep state.

If the system still hangs, or fails to resume, the following workaround might help. Working as root, first inspect the kernel logfiles, then re-compile the driver that is causing the issue as a module – if this has not been done previously. Before calling *echo Sleep_State > /sys/power/state*, first remove the driver module, and re-instate it after resuming. The whole suspend process is easily scripted. Listing 2 shows an example that helped perform Suspend-to-RAM gracefully, although the USB host controller (*ohci-hcd.o* driver module) had previously caused kernel panic on resuming.

Listing 2: Suspend script

```
#!/bin/sh
rmmod usblp usb-storage hid;  # Remove USB Module
rmmod ohci-hcd;  # Remove Host Controller Module
rmmod usbcore;  # Remove USB Core Module

echo -n "mem" > /sys/power/state;  # Suspend to RAM

modprobe usbcore;  # Load USB Core Module
modprobe ohci-hcd;  # Load Host Controller Module
modprobe usblp usb-storage hid;  # Load USB Module

exit 0;
```

If you are experiencing difficulty both with Suspend-to-Disk (Software Suspend or Suspend-to-Disk), and on resuming, you can edit the kernel parameters manually. *resume=noresume* or *pmdisk=off* will prevent the kernel from attempting to restore the previous state on booting. The Software Suspend variant additionally requires you re-initialize the swap partition, *mkswap /dev/hda2* in our example, and then re-assign it to the system: *swapon /dev/hda2*.

Dynamic Clock Speed for Mobile CPUs

More recent mobile CPUs can reduce or increase their clock speeds depending on the current CPU load (Intel Speedstep and AMD Power Now!). To leverage this feature with Linux 2.6, enable *cpufreq* as shown in Figure 2 and make sure you select the correct *cpufreq* driver. The *cpufreq* interface changed during development of Linux 2.5, and is to be found in sysfs in the final release. If you intend to use a kernel compiled with these features, you should use *dmesg | grep cpufreq* to check if the kernel was able to initialize the feature (see Listing 3).

If *cpufreq* quits with an error message, you should first re-check that you have the correct *cpufreq* driver. As strange as this may sound, we had to enable Plug & Play BIOS kernel support (*CONFIG_PNP* and *CONFIG_PNPBIOS*) in our lab to get the *cpufreq* ACPI driver to run.

Entries

Assuming that *cpufreq* has initialized correctly, the following entries should be available below */sys/devices/system/cpu/cpu0/cpufreq* in sysfs: *cpuinfo_max_ freq* and *cpu_ min_freq* contain the maximum and minimum processor speeds for the current CPU. However, *scaling_max_freq* and *scaling_min _freq* define the clock speeds between which the CPU will actually scale. *echo speed > scaling_max_freq* and *echo speed > scaling_min_freq* allow root to change the values. *speed* must be a value between *cpuinfo_max_ freq* and *cpuinfo_min_freq*.

The *scaling_available_governors* entry tells you what kind of clock speed manipulation strategies the system provides: *powersave*, *userspace*, or *perform- ance*. The currently enabled strategy is stored in *scaling_governor*. Root can enter *echo Governor > scaling_governor* to enable a different strategy, where *Governor* must be one of the three known strings defined in *scaling_available_governors*.

To remove the need for command line echoes, the Cpudyn [8] project has developed a daemon that uses the *cpufreq* interface. Cpudyn helps you modify the clock speed strategies of modern CPUs to reflect your own requirements.

ACPI Throttling

Many CPUs that do not support dynamic clock speed manipulation can use ACPI-based throttling instead. This typically has a positive effect on heat and noise levels, and might be a good idea for servers with nothing to do outside of office hours.

The CPU will need to support throttling. To find out whether your CPU has this support, check */proc/acpi/processor/CPUx/ throttling* for a list of available throttling states. The *T0* state tells the CPU to run at full speed. In any other throttling state the CPU is slowed down by the percentage defined in */proc/acpi/processor/CPUx/throttling*.

ACPI for Laptops

The ACPI features described so far are especially interesting for laptop users. */proc/acpi/battery/BATx/info* provides details on your machine's battery status. The charge state, capacity, and voltage are available in */proc/acpi/battery/BATx/state*. If the value drops below a specific threshold, which can be defined by typing *echo threshold /proc/acpi/battery/BATx/ alarm*, ACPI can use */proc/acpi/events* to alert the user.

The */proc/acpi/ac_adapter/ADPx/state* entry tells you if the mains adapter is attached. Plugging in and unplugging the adapter is logged in */proc/acpi/events*. The button driver (*CONFIG_ACPI_BUTTON* kernel option) and the (*CONFIG_ACPI_THERMAL*) thermal driver use the same approach. The button driver alerts when the power, sleep, or lid buttons are pressed. The thermal driver monitors the CPU temperature (the current value is available in */proc/acpi/thermal_zone/THRM/temperature*) and triggers when the threshold defined in */proc/acpi/thermal_zone/THRM/trip_points* is reached.

Laptop owners can use the ACPI daemon (acpid [9]) to respond to the messages in */proc/acpi/events* on their machines. When the lid is closed, acpid automatically tells the machine to suspend to disk.

Despite the variety of ACPI functions, most owners of modern laptops will soon become disillusioned when they notice that they cannot use keyboard shortcuts to change the display brightness on Linux, and that the function [Fn] keys do not provide the expected functionality. Good news for Asus and Medion notebook users: the *ASUS/Medion Laptop Extras* (*CONFIG_ACPI_ASUS*) driver option creates entries below */proc/acpi/asus* that allow you to modify the brightness of your display, and control your laptop's LEDs. The driver converts [Fn] keyboard combinations to ACPI events. A userspace daemon [10] is available for all other functions.

Listing 3: The *cpufreq* test

```
01 [root@sunshine:~]$ dmesg |
grep cpufreq
02 cpufreq: CPU0 - ACPI
performance management activated.
03 cpufreq: *P0: 750 MHz, 22000
mW, 250 uS
04 cpufreq: P1: 350 MHz, 9800 mW,
250 uS
```

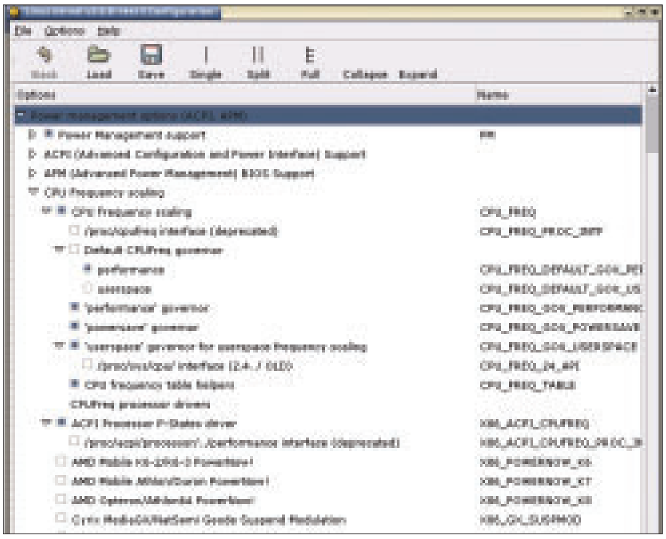

Figure 2: ***cpufreq*** **ACPI options for current mobile CPUs.**

Toshiba notebooks can also leverage a special ACPI driver known as the *Toshiba Laptop Extras* (*CONFIG_ACPI_TOSHIBA*). Unfortunately, the module simply uses */proc/acpi/toshiba/keys* in the proc filesystem to pass [Fn] keypresses, instead of using standard ACPI events. The other */proc/acpi/toshiba* entries manipulate the brightness and the fan. There are userspace daemons available for the Toshiba extensions [11].

Where's the Docs?

Users experiencing issues with Linux 2.6 and ACPI can try following their noses, and searching the Web for documentation. Unfortunately, there isn't any. Current special publications collectively steer clear of the topic "ACPI as a Replacement for the Plug&Play BIOS". The documentation on the kernel itself is the only exception, but it is fairly insubstantial apart from a few notes on ACPI-based power management. The best results that a spot of googling returned were the Acpi4linux mailing list at [12] and a few newsgroups, where at least you can join forces with other people facing similar problems.

Conclusion

The current spate of ACPI hardware will put the operating system developers under pressure to do something about it. Some of you may remember the ACPI problems that accompanied MS Windows 2000. The goals of the ACPI specification with systematic hardware configuration and unified power management, are urgent and important.

On the upside, the recent implementation of PCI interrupt IRQ routing via ACPI on Linux is approaching stability. However, Linux 2.6 rightly tags the new power management Suspend-to-RAM feature as *experimental* – it provides too little support to too few machines.

What's worse, but this is not Linux' fault, is that many systems on the market have faulty DSDTs – even though some of them are by manufacturers who are members of the ACPI consortium. Mere mortal users will probably be out of their depth if expected to provide patched DSDTs to set things straight. A lot of work still needs to be done before ACPI can really assume the role of a central interface that will configure any known hardware. ■

INFO

[1] ACPI: *http://www.acpi.info*

[2] ACPI4Linux: *http://acpi.sf.net*

[3] ACPI4Linux DSDT overview: *http://acpi.sf.net/dsdt/index.php*

[4] DSDT initrd Patch: *http://gaugusch.at/kernel.shtml*

[5] Intel IASL: *http://www.intel.com/technology/iapc/acpi/downloads.htm*

[6] Repairing faulty DSDTs: *http://www.cpqlinux.com/acpi-howto.html#fix_broken_dsdt*

[7] Software Suspend for Linux: *http://swsusp.sf.net*

[8] CPU Dynamic Frequency Control: *http://mnm.uib.es/~gallir/cpudyn/*

[9] ACPI Event Daemon: *http://acpid.sf.net, http://sf.net/project/showfiles.php?group_id=33140*

[10] ASUS/Medion Laptop Extras: *http://julien.lerouge.free.fr*

[11] Toshiba ACPI tools: *http://fnfx.sf.net, http://sourceforge.net/projects/tclkeymon/*

[12] ACPI4Linux mailing list: *http://acpi.sf.net/mailinglists.html*