Introduction to the ACPI specification and its implementation in the Linux Operation System

Max Berger

October 10, 1999

Abstract

This document gives an overview of Advanced Configuration and Power Interface (ACPI), often referred to as successor of APM, and its implementation in Linux. The basic issues of power management are described. This includes the handling of sleeping, suspending, hibernation and shutdown. Thermal management, especially the difference between active cooling and passive cooling, performance mode and silence mode is explained. The Plug'n'Play model of ACPI is introduced. An overview of the core languages ACPI control method Machine Language (AML) and ACPI control method Source Language (ASL) is given. The ACPI implementations in Linux are described: The user-kernel interface (/proc/acpi, /dev/acpi, libacpi, acpid), the new `Int 15h` extension for memory holes, APM compatibility and the AML virtual machine. There is also a short report about ACPI4Linux's current status.

Contents

1 What is ACPI?

ACPI stands for "**A**dvanced **C**onfiguration and **P**ower **I**nterface". Most people refer to ACPI as the successor of APM. Sure, ACPI does specify some power management issues, but they are only a small part of it. Actually ACPI defines a whole new interface for interaction between hard- and software. For a computer system to be ACPI-compliant, the hardware, meaning the chipset and the BIOS, and the software, namingly the OS, must know and use ACPI.

ACPI is the first specification to define real OSPM (**O**perating **S**ystem defined **P**ower **M**anagement). Every decision on what the hardware should do is up to the OS.

2 The Promises of ACPI

2.1 Total control over your hardware

So what are the advantages of having an ACPI compliant system? Imagine, for example, Mike R. Soft, who is working for a large software company. When he arrives at his office in the morning, he presses the power button of his computer. It awakes from hibernation, properly restoring all the programs Mike had opened when he left work yesterday. But still, his computer has not consumed any power during the night.

Mike is a programmer. Therefore, his computer has a very fast CPU, which unfortunately needs a noisy fan to cool it. But this morning, he's answering the phone and needs his computer to take notes instead of compiling huge C++ programs. So he has set it to "silent mode". His CPU is throttled down and hardly generating any heat, and the fan would only be activated in case of an emergency.

Lunchtime. Mike presses the power button of his computer. The computer knows it's lunchtime and enters a sleeping mode, where it will take a maximum of one second to be ready to work again. When Mike comes back from lunch, the power LED of his computer is blinking in doubles. He moves the mouse, the computer wakes up. Mike starts his e-mail program and reads the two messages.

Later that day, it finally is time for him to go home. When he presses the power button, the computer automatically enters hibernation mode, but not for long. There's a backup scheduled tonight. Mikes computer powers up, does the backup, and returns to hibernation again.

2.2 Power management

To explain how ACPI does all this, I need to explain the new concept of ACPIs power management. On legacy[1] systems basically only two power states exist: The System is either **on** or **off**. ACPI defines a much greater quantity. There are:

Global states, ranging from G0 to G3. G0 is the equivalent to **on** in legacy systems where all devices are fully powered. G3 equivalents to **off**, and is (in ACPI Terms) also refered to as "mechanical off". G1 is a state mostly used in laptops today: It means the system is in one of the sleeping states, as described below. And, last but not least, G2 defines a state currently used on desktops with ATX power supply – the so-called "soft off".

Sleeping states, which range from S1 to S5. They differ in power consumption, wake-up latency, and context preservation. While the system is almost awake in S1 (it must wake up within 100 msec.) it is deeper asleep in S3. The states S1, S2 and S3 are part of the G1 global state, where the system is actually still running. S4 and S5 are part of G2, where the system is near to being powered off. S4 is hibernation, currently known from some Laptops, where the memory contents are saved to disk and restored when the system boots up. S5 is the soft-off state.

Device states, ranging from D0 to D3. Not only the system as a whole, but each device may be put individually to sleep. D0 means the device is fully powered and functional, D3 means the device is not powered at all. D1 and D2 define different states in between, which may and will vary with each device.

CPU states, defined as C0 to C3. While a processor in C0 state is fully working, a processor in C3 may loose its internal cache. These power states might come in handy when it comes to multi-processor systems, where each processor can be individually put to sleep.

One example use of these different power states is a voice box. Most people have smart modems which can be used as answering machines. A large problem with these is that in legacy systems the whole computer system must be turned on for the computer to answer the phone. With ACPI, the modem could actually detect a RING signal on it's own, and then wake up

[1]The ACPI specification refers to every non-ACPI system as "legacy"

the Computer. And still, devices not necessary for the voice box program such as printers, graphic cards, etc. could stay off.

Another example would be a server for a small business. It should always be able to answer queries, even at night and on weekends. But most of the time it is idle. Like a modem, an ACPI compliant network card could power up the computer, when it is needed.

2.3 Thermal management

Running a computer idle does not only mean unnecessary consuming a lot of power, but also generating a certain amount of heat. ACPI gives the OS the power to control how cooling is managed. Basically, there are two ways of cooling a system:

Active cooling is at the present time a widely used way to cope with the excess heat of a computer system. Active cooling means there is a device, which is working to reduce heat – almost always a fan. But fans have some disadvantages: They are noisy, which is not good in larger offices and also consume a share of power, which is undesirable in portable systems, where energy is very limited.

Passive cooling is a different approach towards reducing heat. Heat is simply reduced by generating less of it. A regular workstation only needs full performance for short periods of time, and mostly runs close to being idle. Therefore, ACPI gives the OS the power to throttle a CPU. This decreases its temperature, and, as a result, increases the CPU lifetime, which is surely desirable.

So who is to decide between a noisy fan and a slow system? Until now, it was up to the hardware to decide. ACPI puts this decision in the hands of the OS. This has two major advantages: The OS can automatically use an appropriate level of passive cooling, depending on the system load and the user can request high performance when he needs it.

One thing that needs to be mentioned here for the sake of completeness are thermal zones. With ACPI, a computer system can be split up in several thermal zones, each with it's own cooling devices and temperature ranges. Of course, thermal settings can be set individually for each zone.

An important note that should be made on thermal management: The ACPI specification defines an emergency power off on a software base only. In case the system should overheat, it is up to the OS to power it off. Therefore, any OS implementing ACPI should be absolutely reliable to avoid possible hardware damages. Some hardware vendors have implemented an emergency shutdown, but again, this is unfortunately not a must!

2.4 Plug'n'play

So much about thermal management of single devices and the computer system as a whole. But ACPI defines even more! Remember again what ACPI stands for: Advanced Configuration and Power Interface. Now some words on advanced configuration. There currently are many Plug'n'Play systems out there: ISAPNP, PCI, PC-Card, just to name a few. Instead of adding another PnP system for enumeration of ACPI-only devices, such as fans, ACPI defines a new PnP interface which can include the others. The APCI PnP defines the following common functions for each device:

CRS Returns the devices **C**urrent **R**esource **S**etting. This might be useful if a device is already configured by another PnP system or the BIOS. It could be used on module drivers to enable auto-sensing

DIS **DIS**ables the device. When disabled, a device will not use any resources.

PRS Returns the devices **P**ossible **R**esource **S**ettings. This is important if ACPI does the actual configuration. It gathers possible settings of all devices and tries to figure out a setting without resource conflicts.

SRS **S**ets a device's **R**esource **S**etting. After deciding which resources to set a device to, the OS calls this function to do the actual configuration.

This new PnP system gives OS and driver vendors the choice of using legacy or ACPI device configuration. Any existing module can be used unmodified, but new ACPI compliant drivers may use the new functions provided by ACPI. This will make a device driver independent from bus systems.

2.5 ACPIs abstract driver model

ACPI defines methods, thermal zones and sleeping state in the most general way such things can be defined – by creating a new language called "**A**CPI control method **M**achine **L**anguage", AML[2]. AML is basically just a set of methods and definitions, set up in a tree-like structure. AML is

[2]Actually ACPI defines two languages: AML and ASL (**A**CPI control method **S**ource **L**anguage). The can be somehow compared to Java and Java-Bytecode. ASL is the source language for AML

Object Oriented The tree structure in AML suggest a certain kind of object orientation. Some devices have common interfaces, for example for entering the sleeping states.

Dynamic Methods and definitions may be loaded or unloaded at any time. This is very useful when it comes to PnP: A device could bring it's own drivers which get activated once it is plugged in!

Platform independent AML is a well defined byte code language. It has to be executed in a virtual machine, which makes it independent from any OS or hardware platform.

AML gives ACPI quite a potential. If someone would define a common interface for, lets say sound cards, drivers could be written once and then be used on any OS. An OS could even support cards that don't exist yet.

3 ACPI and Linux

So much about ACPI and its promises. Now we are going to take a look at how ACPI and Linux work together. Please note that this is work in progress. Some of the things mentioned here are implemented, some are currently under development and others are merely planned.

3.1 Different ACPI implementations

One problem all ACPI compliant operating systems are facing is the very different implementation of the ACPI specifications by the hardware vendors. Many BIOSes contain broken AML code, wrong entries and other bugs. Windows faces this problem by having a "bad BIOS list". Every BIOS which is on the list has ACPI totally disabled. With ACPI4Linux we try a different approach: Instead of disabling ACPI completely, we split its support up into smaller parts, which can be turned on or off individually. This is the reason for all the different config options you get when configuring your kernel for ACPI.

3.2 /proc/acpi

The philosophy of Linux is to inform the user about what is actually going on in his system. The prefered way to do so is the `/proc` interface. Since we have chosen to give the user a lot of information, we created our own subtree: `/proc/acpi`. Some of the information you will find there is taken straight from from the BIOS, like all the ACPI tables. Other information is generated by the ACPI kernel driver.

3.3 The new memory management

ACPI defines a new memory management for intel-x86-based systems. On legacy systems, hardware-software interaction takes place in the memory space between 640k and 1MB. There is no way to define memory blocks above 1 MB. But ACPI defines an extension to `INT 15h`, allowing the hardware to define additional memory ranges.

3.4 APM compatibility

A very important thing to have is APM compatibility. After all, ACPI is the successor of APM. There are currently a lot of programs using the `/proc/apm` and `/dev/apm_bios` interface. What we are working on is totally faking this interface and thus allowing APM programs to continue to work. The values given in `/proc/apm` are currently all fake, and not based on actual values. `/dev/apm_bios` knows currently about standby and suspend, where standby is mapped to S3 (normal sleep), and suspend to S4 (hibernation).

3.5 Kernel interface

APM compatibility is one way for the user and regular programs to interact with the ACPI drivers, but there are, of course, new ACPI ones. We have decided to give a user three ways for talking to the ACPI driver:

`/dev/acpi` is the main interface for user programs to talk to the kernel modules. The definitions for this interface are under development as this document is being written.

libacpi Due to this and to keep consistency with future version, one should never use the `/dev/acpi` interface directly, but instead use the provided libacpi. Functions here will not change, but instead be marked as deprecated in future versions. This ensures backward compatibility.

acpid There are a lot of decisions that don't need to be made in kernel-space. But still there should be on central module for managing them. This module is the **ACPI d**aemon. It contains all code that is neither essential for startup or shutdown, nor time-critical, such as basic thermal management, nor needs to reside in kernel space. Without the daemon running the ACPI driver is limited to these critical functions.

3.6 AML virtual machine

Unfortunately, work on the AML virtual machine is not finished yet. The VM is a central part of an ACPI implementation. AML consists of method- and other definitions. Therefore the VM is actually two parts: A parser at boottime which creates the AML tree structure and stores the definitions and functions there, and the actual VM which executes functions when they get called.

3.7 You and ACPI4Linux

So what do you, as a user, have to do to use ACPI? First of all, get the kernel patch and install it. ACPI will hopefully be part of Linux 2.3.x, but we are developing a patch for 2.2.x right now. Then, if you want to, submit information to our BIOS web-page. This will help us to find out what different ACPI hardware implementations actually exist. There will hopefully soon be some nice ACPI applets, that will show your CPU's temperature, the battery status, or other ACPI information.

If you are a programmer, take a look at the libacpi. Use some of the functions, especially if you are writing daemons. And, of course, help in the ACPI4Linux project is very welcome.

3.8 Current status

The APM compatibility is is still not finished. We're currently working on the userspace-kernel interface and on the AML VM. ACPI support in Linux is far from being completed. For updates on this issue please take a look at our web site.

4 The future of ACPI

I've explained what ACPI is, what ACPI promises, how ACPI and Linux are working together. The current ACPI specification is ACPI 1.0b. But version 1 is only the beginning. Actually, ACPI 2 should be out by now, but there have been some delays. We hope the new specification will be more specific in certain points, provide us more options, and most important is compatibile to 1.0b.

ACPI is surely not a must-have feature. If you don't like it – don't use it. After all, in Linux it is your choice! But I won't stop using ACPI. The new things that can be done with the hardware are just too tempting!

Thanks

I want to thank everyone involved in the ACPI4Linux project. First of all Simon Richter, who's the other project leader and co-author of this document. Then the people whose contributions are in the patch so far: Stephen Early, David Fries and Oliver Neukum. Of course, everyone who has submitted his ACPI tables for our ACPI Bios list. And, last but not least the people at our student representation for enabling us to put up our web-server.

References

This document describes a current status as on the date listed on its first page. The main reference was the ACPI specification 1.0b by Toshiba, Intel and Microsoft.

Max Berger
max.berger@phobos.fs.tum.de
http://phobos.fs.tum.de/acpi/