



**Klaus Knopper is the creator of Knoppix and co-founder of the LinuxTag expo. He currently works as a teacher, programmer, and consultant. If you have a configuration problem, or if you just want to learn more about how Linux works, send your questions to:**  
[klaus@linux-magazine.com](mailto:klaus@linux-magazine.com).

# ASK KLAUS!



consumption, battery issues, suspend-to-ram, and even suspend-to-disk, and the operating system could blissfully ignore these functions.

Then, things started to break with the introduction of the now-standard ACPI, which is supposed to not only let the operating system control power features, but also manage part of the hardware (IRQ) configuration and system state management. Some may say ACPI was introduced in order to make more money by selling cheaper chipsets, because all the logic now has to be put into the software instead of self-controlling hardware. But the board manufacturers praise ACPI for its “fully controllable access” to certain features (like power management and hardware configuration) from within the operating system, and today there are even boards that really won’t work without an ACPI-enabled kernel.

If your board still supports APM, and you would rather let the hardware automatically control power management than experiment with ACPI, switch to APM instead of ACPI. In my opinion, APM, if it is available, gives better results (both in terms of performance and battery lifetime for laptops) and it is extremely easy to configure. You don’t have to configure ANYTHING; just compile APM support into the kernel, and disable ACPI on boot by giving the `acpi=off` kernel option. For some boards, this is the only reliable way to get suspend-to-ram or suspend-to-disk working perfectly.

Now, if you still want to use ACPI (or if you have to, because you have a board that requires it), and you configure ev-

erything by yourself, you are in for a lot of trial-and-error sessions until you find out which parts of ACPI will work, and which won’t. I’ve had go through this procedure on many boards, too. If vendors would just offer working configuration examples for Linux (as they do for Windows by providing “board drivers” that also fix errors in the chipset), everything would be easy. But somehow they seem to assume that every Linux user is a hardware expert who will manage to circumvent broken chipsets.

Now for configuring and experimenting with ACPI, you should first install an ACPI-enabled and REALLY UP-TO-DATE kernel. Boards with different ACPI implementations appear all the time, and Linux developers have to fix bugs and find out how to handle ACPI for these newer boards in each scheduled kernel release. So please try a kernel that is at least newer than your computer’s board.

Parts of ACPI are in the static part of the kernel, but virtually each and every kernel module must also implement the necessary ACPI hooks in order to shut

## Power Management

**?** No matter which Linux distribution I’m using, I never can get the power management features to work on my Intel-based laptop. The system either doesn’t sleep, or it breaks something on its way down. How do I identify what the problem is, and what can I do to get power management working?

**💡** A couple of years ago, there used to be a thing called “Automatic Power Management” or APM (which still works well even on some newer computers, even if the board description claims it’s no longer supported). With APM, the hardware took care of CPU fan speed, CPU power



down a device or change power status on request. For some computers, you have to use the `acpi=force` boot option to REALLY enable ACPI, even if the kernel doesn't think it's a good idea.

For things like suspend-to-disk, the tricky part is to find out which hardware components support it, as well as which kernel modules have the necessary ACPI hooks to put a component into a sleep state and wake it up later without confusing the hardware.

An sample procedure for enabling suspend-to-disk on your laptop would be something like:

0. You remembered the bit about backing up all your valuable data, right?

1. Make sure your swap partition can hold the complete size of your computer's RAM, plus all swapped-out components of the running software. (You only need this for suspend-to-disk, not for suspend-to-ram.) The kernel should be compiled with "software suspend-to-disk" enabled. Caution: this option only appears on SMP kernels if you have marked "hot-pluggable CPU support" as enabled as well. Also, make sure the suspend partition (your largest swap partition) has been configured in the kernel setup. You can, however, use the `resume=/dev/hd*` boot option to declare this on boot up.

2. Unload ALL kernel modules before attempting to suspend-to-disk. Onboard sound cards, PCMCIA and USB controllers, and devices are especially critical. Most of these components don't really support power-off and wake-up anyway, so other operating systems do the same thing: they disable/unload the drivers on suspend and restart them on wake-up. You should probably also try this technique without `framebuffer` mode on the console. Until recently (kernel 2.6.15), the `framebuffer` did not seem to wake up well, especially when direct rendering was enabled.

The same problem also applies to X-Servers, although you should not have to shut down X in order to suspend-to-disk (and wake up) properly. Sometimes it helps to switch to a text console before suspend in order to force a clean refresh of the X-Server later. You can even do this inside a script with `chvt 1`.

3. Initiate suspend-to-disk with

```
sync ; sync ; sync # Trust me...
echo 4 > /proc/acpi/sleep
```

(You should have switched to the system console in order to see what's happening.)

Your kernel should now try to free memory by copying out everything that's in RAM, plus the chipset state, to the swap partition, remove the swap signature bits from the partition (so a `swapon` will fail if someone accidentally boots without the `resume` option), and then switch off power.

### Listing 1: ACPI Example

```
01 # cat /proc/acpi/thermal_zone/THRM/cooling_mode \
02     /proc/acpi/processor/CPU0/performance
03 Result:
04 cooling mode:   active
05 state count:   2
06 active state:   P0
07 states:
08   *P0:          1800 MHz, 20000 mW, 250 uS
09   P1:          1200 MHz, 10000 mW, 250 uS
10 # echo 1 >/proc/acpi/processor/CPU0/performance
11 # echo 1 >/proc/acpi/thermal_zone/THRM/cooling_mode
12 # cat /proc/acpi/thermal_zone/THRM/cooling_mode \
13     /proc/acpi/processor/CPU0/performance
14 Result:
15 cooling mode:   passive
16 state count:   2
17 active state:   P1
18 states:
19   P0:          1800 MHz, 20000 mW, 250 uS
20   *P1:          1200 MHz, 10000 mW, 250 uS
```

### More on Heat

Most boards SHOULD have sane ACPI implementations that automatically turn the fan on if the CPU starts to overheat. However, my recommendation is not to rely on this and run some checks to be sure. You can have a look at the current CPU temperature every now and then by typing:

```
# acpi -V
```

```
Result:
Battery 1: discharging, 99%, 2
02:27:58 remaining
Thermal 1: ok, 75.0 degrees C
AC Adapter 1: off-line
```

(The "acpi" command is part of the "acpi" tools package).

If you see the CPU temperature climbing up over 90 degrees C and your board fails to auto-control "emergency" cases,

you should switch "active" cooling back on quickly.

```
# echo 0 >/proc/acpi/2
```

```
thermal_zone/THRM/cooling_mode
```

Some programs that run as daemons are supposed to automatically reduce CPU speed and power when the system is not doing anything too strenuous, and these programs give the CPU some extra power in case fast calculations are needed or load is high. Most of these programs don't work that well, though, and the schemes under which they decide WHEN more CPU power is needed are kind of arbitrary. If you want to try, just `load acpi-cpufreq`, or `speedstep-ich`, and see what you can change in `/proc/acpi`. Programs like `cpufreqd`, `powernowd`, and `powersaved` are supposed to provide some automatic power and frequency scaling for CPUs that support them.

This part usually has a good chance of working flawlessly. (The *really exciting* part is when your computer wakes up later.)

If you experience kernel panic at this early stage, recheck if everything that can be built as a module (and is not needed at an early boot stage for accessing the data on the root file system) is really built as a module and has been successfully unloaded by `rmmmod` before the suspend. Don't forget to run the file system check after a failed suspend-to-disk if your installation does not do this for you. Maybe you need a newer kernel...

- For booting into the saved/suspended session, just boot up your linux kernel, but (see item 1, above) you should have previously specified the "resume" partition in your kernel configuration or use the `resume=/dev/hd*` boot option. Otherwise, the kernel will just boot up normally and ignore the fact that there is a saved session on your swap partition. This can lead to a very bad situation. All file systems will be checked and probably modified, but when, at a later boot up, the saved session is detected correctly, it still assumes the old state of the file system, which will almost certainly lead to a very corrupt file system.

Better never boot without resuming a saved session, even if you only plan to mount your file systems read-only. `ext3`, for example, at least the versions I have seen, modifies an `ext3` partition even when it's mounted read-only!

The kernel should display the progress of loading the saved memory from the swap partition. After this, the swap signature is restored, so the swap partition is recognized as swap again.


In the unlikely case that resume-from-disk really does work at the first attempt, you will still have to (re-)load the kernel modules that have been previously unloaded in order to get your sound card and everything else working.


If they are working correctly, save-to-disk and resume-from-disk really help speed up the boot process, because they keep the programs that were there be-

fore up and running (except for the ones that rely on kernel modules that have to be unloaded). You will have to find out which modules you can keep during suspend/wake-up and which ones are CPI-aware. The most problematic ones turn out to be everything related to sound cards, pcmcia (controller + devices), and USB (controller + devices). You should always build these as modules so they can be removed and reloaded easily and won't disturb the suspend/wakeup process.

Despite the problems, ACPI does provide some interesting options. Some "easier" things you can do with ACPI include tricks like switching the CPU power/frequency and fan noise (Listing 1). At least on my laptop, I can reduce the CPU frequency (and cut down power consumption), then switch off the CPU cooling fan for a quieter system that is still not too slow. Of course, if you tinker with the cooling fan, you'd better make sure your can turn it on again. See the box titled "More on Heat."

### Window Bet

 Please help me settle a bet. Will my system run significantly faster if I close open windows that are minimized in the taskbar?

 This bet is not easy to settle, because it depends on the programs that opened the windows.

If they are heavy on CPU, and memory hogs, of course, your system will be much faster if you just quit them.

By the way, just in terms of CPU usage, it sometimes helps if you just `SUSPEND` them instead of terminating them. Send

```
kill -STOP process_id
```

to the `process_id` of the program in question. The window will become non-re-

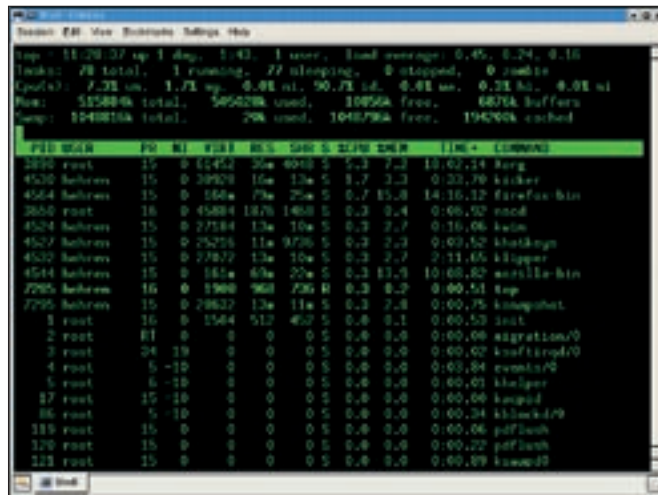


Figure 1: Checking CPU and memory usage with the "top" command.

sponsive, but you can still wake up the program later by sending

```
kill -CONT process_id
```

which lets the program grab some CPU time. This trick will not temporarily free the memory the program allocated, but it CAN help if you don't want to quit a running program but just want to suspend it.

You can also use `nice` on program start, or `renice` later, to give a lower priority to the process. But you are right in thinking a program that's NOT running can't eat up system resources.

Unfortunately, KDE may think that a suspended program has crashed, and it may pop up annoying dialogs that suggest you really kill the program.

I would agree that OpenOffice in the background consumes a lot of memory and can slow down other processes (unless the inactive parts of OpenOffice are swapped out). A more common case of programs hogging system resources is a web browser like Konqueror or Firefox running animated graphics, flash, or Java applets that consume CPU and memory. You can run "top" to check this. If you see a significant amount of system resources claimed by programs of this kind, it would make the system faster not to run them. Or, just exit the browser and relaunch.

It is probably a better idea to edit the browser's preferences, and disable automatic plugin launching, then just start them on-demand (if the browser supports this; I know that Konqueror does).

So did you win the bet? ■

Send your Linux questions to klaus@linux-magazine.com.