# Dr. Linux KERNEL STORIES

MARIANNE WACHHOLZ

**Surgery is open again! The good doctor observes the patients in Linux newsgroups, issues prescriptions and cures Linux kernel ailments.**

## Make it!

I was just trying to compile a **kernel**, but after the entry

```
root# make menuconfig
```

I kept getting the message

```
make: *** No rule to make target `menuconfi?
g'. Stop.
```

The command *make xconfig* also gives me the corresponding error message. What's going wrong here?

***Kernel:*** *The operating system kernel consists of the components which comprise the actual operating system. Only this has direct access to the resources of the computer (disk space, memory and computing time, keyboard etc). If a command is sent or a program started, the requisite program code is loaded in the main memory and started. This program is now referred to as a task. Tasks have no access to the resources: they request these as required from the kernel. The Linux operating system kernel allocates the necessary computing time and the memory so lightning-fast that it gives the impression that programs can start in an instant.*

***Compiling:*** *If a source code file is transferred to a compiler (gcc), the original code (readable by humans) turns into a object file (readable by machines). The object file is in turn linked with the necessary libraries and then a program that is ready to run is created. Libraries contain standard functions, which are used by many programs, such as for screen output. The result of this is that all these functions do not have to be rewritten all the time for every single program.*

***Sources:*** *Often also referred to as source code or source text. This is the text a programmer has written (in a programming language such as C++). It is only through compilers known as conversion programs that the text which humans can read is turned into a binary program or a file that the machine can execute.*

■

**Dr. Linux:** Unlike many commands, which are simply entered on a command line for execution, regardless of the current working directory, to execute this command you have to be in the directory with the kernel **source** (usually */usr/src/linux*).

If you still get the error message after a

```
root# cd /usr/src/linux
```

the sources of the kernel (often also called kernel sources) are missing altogether and will have to be installed. Because these are very large, they are not usually automatically included in the system during a standard installation. For SuSE users, from SuSE Linux 6.2 the automatic installation of the kernel sources is also no longer done by default. The kernel sources for your actual system can be found in the most common distributions on the installation medium. From there, depending on the system; YaST, gnorpm or dselect these can be installed later. On the Internet you can find packed kernel archive as *tar.gz* or *tar.bz2* among others at *http://www.kernel.org/*

## Over and out

I have installed SuSE 7.0 on my computer. Now I want my computer to switch itself off after power down, as happens with Red Hat. Can this be done with the SuSE distribution?

**Dr. Linux:** If you want to use APM (advanced power management) SuSE provides you with a special kernel from Version 6.3. This is precompiled and can be installed with YaST. SuSE gives you information on this in the support database on your

system or at *http://sdb.suse.de/sdb/de/html/index.html* under the keywords APM, ATX. The prerequisite for this kernel to work perfectly is that the APM functions in the **BIOS** of your computer must be activated.

If you start YaST as root, have your installation CD to hand, because by the time you get to kernel selection you will be asked to insert this CD. Via the menus *Administration of the system/kernel* and *boot configuration/Select boot kernel* you will reach the appropriate selection window, where you will find among other things the Kernel with APM support (Figure 2).

If you select yes in the next dialog window (Figure 3), YaST creates a new */usr/src/linux/.config* with the description of the APM kernel.

To finish, you will receive a message confirming the kernel installation, and YaST reinstalls LILO (Linux Loader), so that the kernel which has just been placed on your hard disk is also ready when you boot up. Normally you confirm this in turn with 'yes'. Experienced users may wish to edit the LILO configuration file */etc/lilo.conf* thus created personally and inform the boot manager, by calling up the */sbin/lilo* program, about the new kernel. If you are using a boot manager other than LILO, please follow the appropriate instructions.

And now you've done it – the new kernel will become active when you next boot up. Before you do so, though, and install the APM kernel on your computer, one more last tip: On my computer, bought from a department store, the installation of this kernel with SuSE 7.0 did not in fact work very well, but it is still sensible to find out about the (later) installation of modules in your manual, so that you can cope with any error messages. Users of SuSE version 6.3 should (as described in the manual in chapter 3, 'YaST'), also reinstall the packet *kernmod* from series *a* after installing the kernel. If you want to build the kernel yourself, you can of course also include advanced power management. You can create the kernel configuration on a graphical user interface with the command

```
root# make xconfig
```

The APM functions can be found under the menu

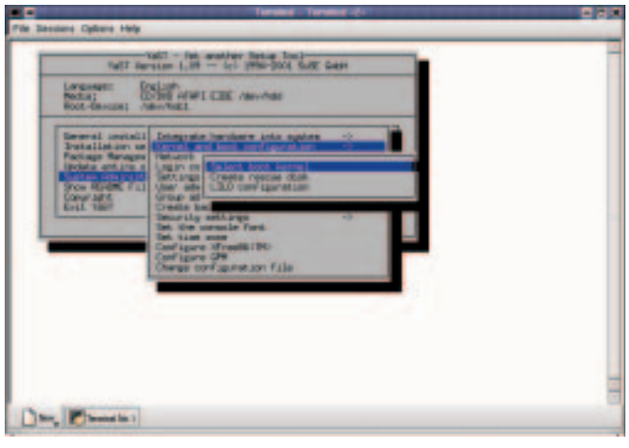
***BIOS:*** *(basic input/output system) The basic software of computers. When booting up the computer this program performs a self-test, which among other things, determines the graphics mode and checks the motherboard and the main memory. Users can make changes to the BIOS settings. You can find precise details in the manuals, which come with the computer and/or motherboard when you buy them. You can also find more detailed information on the website http://www.sysopt.com/bios.html*

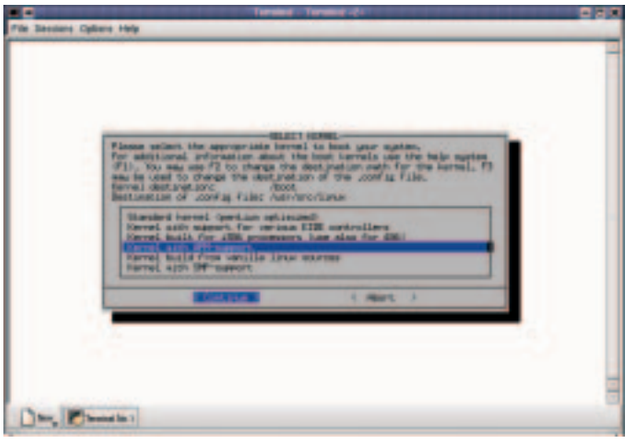
**[top left]**
**Figure 1: The path to kernel selection in YaST**

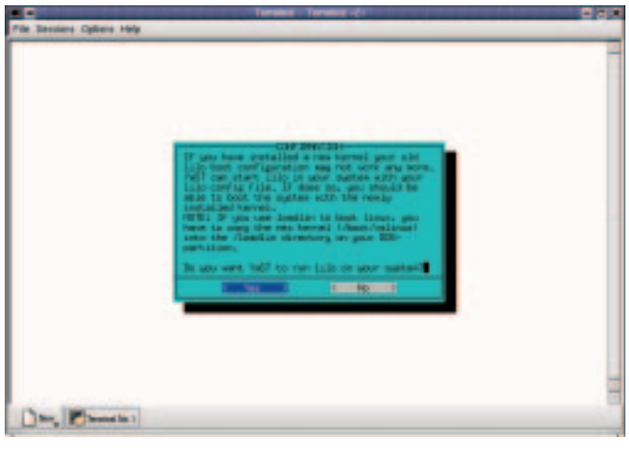
**[top right]**
**Figure 2: Kernel selection of SuSE 7.0 in YaST**

**[bottom left]**
**Figure 4: The boot manager has to be reinstalled**

**[bottom right]**
**Figure 4a: Finally you will see the entries of your LILO boot configuration**







***Real mode**: When a computer is switched on a code starts the BIOS, which after its (successful) checks reports back with 'OK'. Only then does the search begin for the operating system. Depending on the sequence set in the BIOS the first sectors of disk, CD-ROM or hard drives are searched for a boot loader (e.g. LILO). If a loader is found and activated, the kernel begins to load. This starts the operating system, and the CPU can switch from Real Mode into Protected Mode. It is only in the second of these that a computer can contact its whole memory.*

*Modules: Drivers which are only loaded as needed into a (modularised) kernel. The advantage of such a kernel over the monolithic kernel, which has all drivers integrated permanently, is that only drivers which are needed during running time are loaded and take up no memory space when not in use.*

*Basic settings* (Figure 5). This also applies, of course, if you configure your kernel with the command

```
root# make menuconfig
```

(Figure 6).

## In at the deep end

I would love to build a kernel, but I'm afraid that with my level of Linux skill, something could go wrong. Is there a safe method?

**Dr. Linux:** Get hold of plenty of information on the subject of kernel conversion, so that you are not taken by surprise by problems which suddenly crop up. One source of information is the kernel HOWTO. If you have not installed this on your system, then you can find it at *http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html*. Your system also usually offers you comprehensive texts for self study in the directory */usr/src/linux/Documentation*. In order to avoid errors due to excitement or volatility, I put all of my kernels onto a boot disk first to test that they were working. This allows you to take a fairly nonchalant view of any error messages and if necessary, start again at the beginning. You should also have a boot disk that has been proven to work. Using

```
root# make zdisk
```

or

```
root# make bzdisk
```

You can place a new kernel directly onto disk, where the kernel is first created, then compressed and finally copied onto the disk.

## System too big?

It's not possible to compile a kernel on my system. After the command

```
root# make zImage
```

I get the message

```
System is too big. Try using bzImage or modules.
```

**Dr. Linux:** This message or, again, the message 'kernel too big' means that your kernel image is over the permissible upper limit of 512K. This maximum size is due to the fact that the kernel loader is started in **Real mode**, where there is no more memory available.

But with the latest versions of Linux it is no longer necessary to create a kernel image which keeps within these limits: The latest versions of LILO or Loadlin also cope with a bigger kernel – although only if it has been specially created. The corresponding command for this reads

```
root# make bzImage
```

The b stands for 'big'. Incidentally, you also get a big kernel like this if you use the commands

```
root# make bzlilo
```

or the aforementioned

```
root# make bzdisk
```

*make bzlilo* not only builds your new kernel, but also takes over the entry in the boot manager LILO. The prerequisite for this is that you do not use any alternative Linux boot loaders (grub for example) and that the program *lilo* is located in */sbin*. It may sound simple, but *make bzlilo* is not the best choice for a newbie kernel. You should first read up on how to edit */etc/lilo.conf* in the event of problems and enter your old, functioning kernel as a precautionary boot alternative. If you want to stay with *make zImage* when the error message appears all you can do is swap several drivers as modules to trim down your kernel image. ■

**Figure 5: Setting APM support with make xconfig**

**Figure 3: Kernel installation on the fly with YaST**

**Figure 6: ... and with make menuconfig**