# ASK KLAUS!

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

## Boot Sequence

? Thanks for answering our "average user" questions so diligently and simply. I am not new to Linux. In fact, I have been using it since the mid-90s when I installed a mail relay and Internet gateway using Red Hat 6. I also jumped with joy when my ipchains-based firewall setup actually worked. At the time, I could find no support or training in my country, and I had to figure everything out myself.

I have a problem that has been annoying me for some time. Would you detail the steps involved in booting a Linux distribution such as Red Hat? Describe every step taken in the boot sequence – loading of bootsector, kernel, mounting of partitions, etc. I would also like to know which files are involved, and in what order they are read and used.

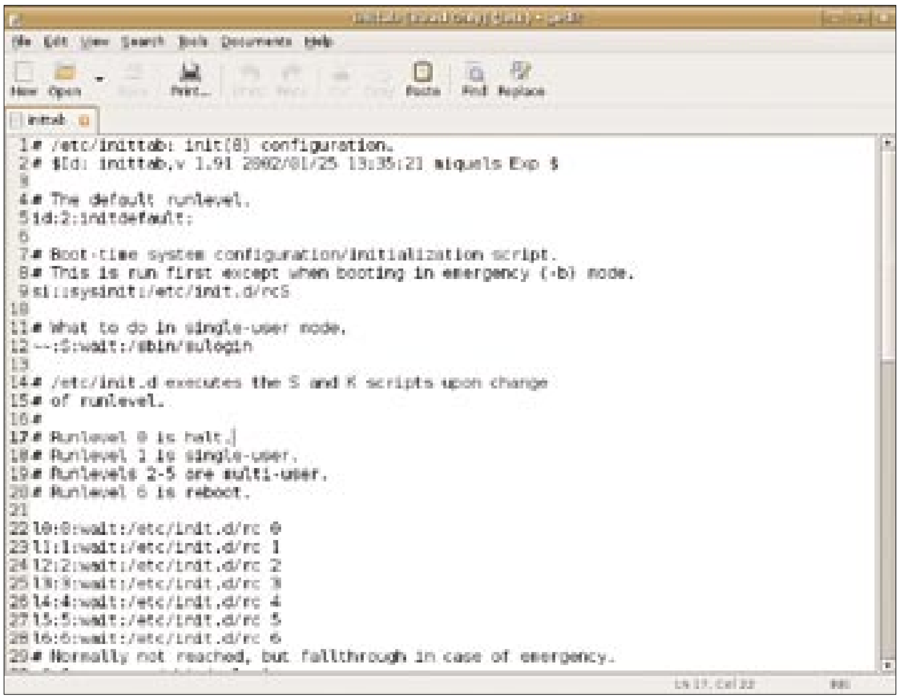
The full story would probably fill a book, but here are some things that I think are most important to know about the boot process. The following answer may be specific to what is know as the "PC Architecture," usually meaning Intel/AMD-based processors and corresponding boards. In other hardware architectures, the boot process is similar, yet the process varies depending on hardware design.

Usually, when you switch on your computer, the hardware runs a short "self-test" sequence that can partly be influenced by the settings stored in the BIOS. Parts of this self test are:

- A quick RAM check just to see whether all addresses of installed RAM seem to be accessible. (For a more thorough test, you can use a tool such as *memtest86+*).
- A quick auto-configuration of the PCI and (old) ISA bus. Interrupts and IO addresses, again influenced by BIOS settings, are assigned for the initial boot phase.
- A search for "bootable roms" as part of the hardware initialization. At this point, the graphics card or SCSI adapters can run small configuration subroutines that don't require a running operating system.
- A check of bootable devices (in the order set in the BIOS) for a bootloader. If a bootloader is found, the bootloader is started. Some bootloaders (frequently PCE) may return with a "no luck, continue" result, and then the next device is tried.
- The bootloader (LILO, GRUB or the Windows *boot.ini* bootloader) reads the operating system kernel from disk/CD-ROM/USB-flash or another bootable device. Optionally (Linux feature), a compressed initial ramdisk image is also loaded into memory as a "virtual disk." Kernel and initial ramdisk get decompressed into memory.
- Now comes a critical part. At this point, the Linux kernel, after optionally changing the VGA resolution, switches to "protected mode," which has a different address scheme. The computer "forgets" everything it once found when running the initial self tests and drivers. This also means that (apart from a few exceptions), Linux now has to provide its own drivers in order to access any storage media.

This can lead to the funny effect that loading Linux from a hard disk works well (because the BIOS routines knew how to read from a drive), but after the kernel is running, you get an error message saying that the media from which the kernel has just been loaded is "not accessible." This is, of course, just happening because of a missing module (not included in the kernel or initrd) for handling the device. I tend to build the kernel in a way that it can handle all common "bootable devices" on its own without needing additional modules or configuration. Therefore, the Knoppix kernel contains drivers for IDE/PATA and SATA, so devices attached to these controllers get recognized at an early stage.

- The kernel tries to initialize the hardware, so as to make it usable for applications. For part of this, Linux uses its own settings for interrupt/IO and chipset configuration, but the options set in the BIOS are used, too. You can influence the Linux kernel's behavior at the initial boot command line or via the *append=* boot options set forth in *lilo.conf* (LILO) or *menu.lst* (GRUB).
- If an initial ramdisk was loaded, this is now used as the root file system, and a script or program called */linuxrc* is executed. If the newer method of initramfs was used, the ramdisk can even be included in the compressed kernel binary. The initial ramdisk usually contains add-on modules (or module loaders with some kind of initial hardware detection); or, if you need a very quick startup without using "init" (the next step) as the control program, the GUI or program that is supposed to run as the main application is immediately started. In most distributions, hardware support and network boot is also handled in the initial ramdisk.
- After access to initrd has finished and the last process ends, the kernel will try to start the master control program init, which is located in */sbin* or */etc*. If it is no longer needed, the memory used for the initial ramdisk is freed. init reads its configuration from a file called */etc/inittab* (man 5 inittab).
- init then starts all programs, mostly shell scripts that start other programs or daemons, which are given in the */etc/inittab* configuration lines. For many GNU/Linux distributions, the scripts under */etc/rcS.d/* are run first. The scripts do further system configuration: remount of the root filesystem (a different one); filesystem check and mounting of hard disk partitions given in */etc/fstab*; network setup; and the permanent login processes on the text consoles, as well as (in most distributions) the display manager *kdm*, *gdm*, or *xdm*, depending on your preferences. The runlevel selected at the boot command line or given as default in */etc/inittab*, determines which scripts are run afterwards. Look for *ls -l /etc/runlevelnumber.d/**. You can get your current runlevel number by just typing *runlevel*.


File Edit View Search Tools Documents Help
New Open Print... Paste Find Replace
inittab

```
# /etc/inittab: init(8) configuration.
# $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $

# The default runlevel.
id:2:initdefault:

# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS

# What to do in single-user mode.
~~:S:wait:/sbin/sulogin

# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
# Normally not reached, but fallthrough in case of emergency.
```

**Figure 1: init reads its configuration from the inittab file.**

After this, the system is up and running and you can log in (if you have not automatically been logged in already). Most errors happen (if not due to kernel misconfiguration) in the init scripts, which are not always written with easy recovery in mind.

For example, if a corruption in the root filesystem is detected during the file system check, you may be unable to log in and "repair errors manually" in the way that is suggested by the error messages at this step.

In this case, you can often use a trick that allows you to skip most of the boot process and just start a single program right away after kernel and initrd are loaded. This can, referring back to your question, be extremely helpful if you accidentally misconfigured iptables or low-level settings so that the system does not reach a stage that would allow you to log into it.

To get such an "emergency shell" at the lilo or grub boot prompt, type:

```
linux init=/bin/sh
```

You will then be prompted to enter commands immediately after the kernel's hardware detection, a least if your root filesystem is still in a somewhat usable condition.

In most setups, the mounted root file system is still read-only, so if you plan, for example, to reset the root password without knowing the old one, you should remount your root file system read-write first, using:

```
mount -o remount,rw /
```

Then you can manually try to repair everything from the shell.

Be very careful not to hit the **ctrl-alt-del** key combination during your system repair attempts, since, without using init, that key combination is not rerouted to a clean shutdown and would reset your computer without any further notice or question, probably without giving the kernel a chance to save any buffered changes.

After you are done with your emergency rescue shell session, just remount filesystems as read-only:

```
mount -o remount,ro /
```

This will then allow you to just hit the reset button or use the previously mentioned immediate reset by keyboard.

## Ruined Everything!

I'm not sure what's wrong. My computer recently gave the message "operating system error" when I tried to start it. Now a message says "searching for a boot record," and it never stops searching. And now my DVD/CD drive is missing.

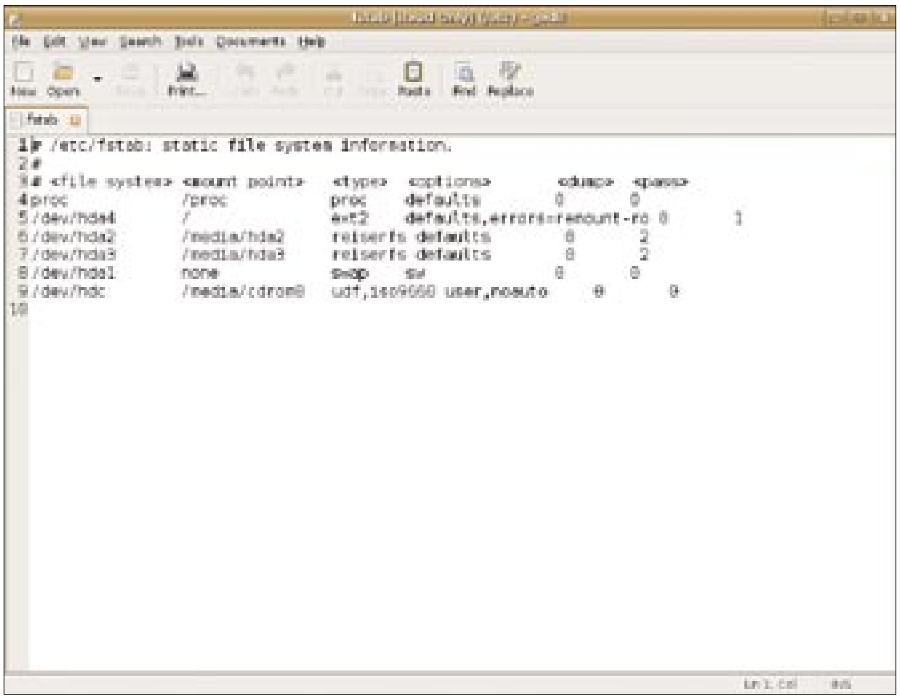
It may be a hardware error, which cannot be repaired by software. If a ruined BIOS setup is likely, try setting all BIOS options to the "slow but safe" variant. (Sometimes this is *not* the safe variant, but the "normal" setting is.)

Also, major failure to work can also arise from a changed hardware configuration, even if the BIOS only *thinks* that you may have changed something, so it automatically changes the chipset settings, interrupt numbers or IO addresses on its own. I've seen it happen. Hardware is evil.

I've seen computers that would do hard lockups without even attempting to boot into a usable system, working again by just changing the sequence of network cards *or* by removing a soundcard that was installed in addition to the onboard card.

Also, connectors do age. Despite what hardware vendors want you to believe, there is such a thing as oxidation of contacts and changes in the physical geometry that makes signal cables lose their contacts. So, if you make sure that all hard disk and CD-ROM connectors are carefully fixed in their sockets, and no plastic attachments are loose or apparently broken, there is a good chance that your computer will magically work again even though it refused to recognize any drive attached to the IDE or SATA sockets earlier.

You can also try new cables. I don't believe in contact/cleaning sprays, though, because these can easily short-circuit modern connectors, disintegrate isolation, and – in the worst case – ruin your board. Also, ram may need to be replaced because it's failing after a few billion read/write cycles.


File Edit View Search Tools Documents Help
New Open Print... Paste Find Replace
fstab

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/hda4       /               ext2    defaults,errors=remount-ro 0       1
/dev/hda2       /media/hda2     reiserfs defaults        0       2
/dev/hda3       /media/hda3     reiserfs defaults        0       2
/dev/hda1       none            swap    sw              0       0
/dev/hdc        /media/cdrom0   udf,iso9660 user,noauto     0       0
```

**Figure 2: The fstab file contains information on drives and mount points.**

So much for the hardware rescue attempt. For the software side, sometimes a BIOS upgrade can help eliminate problems that only showed up after installing a new operating system version.

Sometimes hardware features won't work well, and instead of patching hardware with a soldering iron, it can be wise not to use some features that are not urgently needed, such as IRQ balancing or ACPI.

Knoppix has a good – yet long – boot command line to see whether problems caused by an incomplete onboard implementation of features that have found their way into the Linux kernel vanish:

```
knoppix acpi=off noapic ⮐
nolapic pnpbios=off pci=bios
```

*nolapic* is not a spelling error. I recently discovered some notebooks that would not boot without the "nolapic" option.

Using nolapic, everything – including ACPI, CPU frequency scaling, and all onboard controllers – worked flawlessly, while without nolapic, the kernel stopped dead at recognition of the hard-disk controllers (where the fault was apparently neither the disks, nor the controllers).

## Root Password

I have a live CD of Knoppix 3.7. I would like to use it to do recovery and disk partitioning, but when I boot, it comes up in user mode, and I need root to run administrative programs. How do I set the root password for a Live CD?

Most setup scripts in Knoppix (located in the Knoppix menu, beneath the KDE menu) will use "sudo" to run setup tasks under root privileges. When you start a program that is designed to ask for a root password, instead of being run by sudo, you probably do need a root password. Passwords in Knoppix are locked/invalid. There are no backdoors or "default passwords." Set a new root password with:

```
sudo passwd
```

or visit the root console by *control-alt-F1* and type:

```
passwd
```

Then just switch back to X with *control-alt-F5*. ■

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