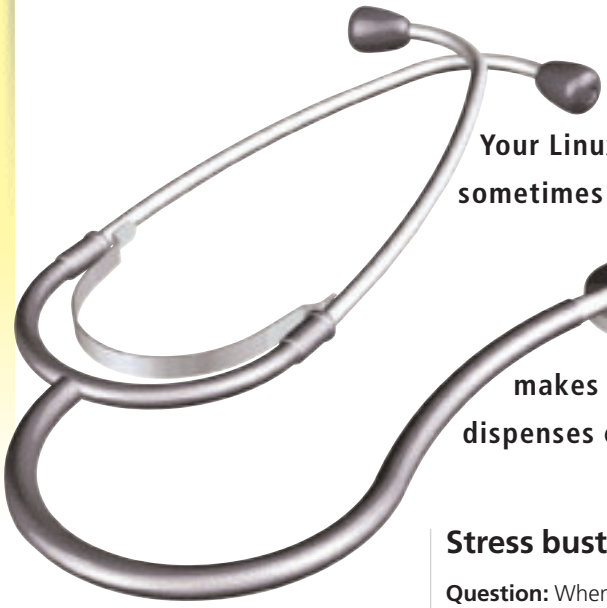
Dr. Linux
THE DOCTOR PRESCRIBES

MARIANNE WACHHOLZ



Your Linux operating system may sometimes be in less than perfect health. Dr. Linux monitors the case load, makes out prescriptions and dispenses expert advice.

Stress busting

Question: When I power up my computer the following messages appear repeatedly:

```
/dev/Drive has reached maximal mount count, check forced.
```

Can I influence the frequency with which my **file system** is checked?

Partitions and file systems: *Before data can be stored on a hard disk, one or more logical areas or partitions must be defined. Their locations are recorded in a partition table. Within each partition a file system suitable for the operating system used must be set up. This enables the operating system to manage files and folders. The most widespread Linux file system is the "Second Extended File System" ext2fs. A device name is specially allocated to each partition, thus enabling the operating system to access the data in the file system. Linux uses letters and numbers to describe hard disks and partitions. The letters indicate which hard disk is used. For IDE hard disks the following applies:*

- *hda: the master hard disk on the primary controller,*
- *hdb: the slave hard disk on the primary controller.*

In the case of SCSI devices the device names are as follows:

- *sda: first SCSI disk,*
- *sdb: second SCSI disk.*

Each hard disk can have a maximum of four partitions, which would be named, for example, hdax, where x is the partition number. One partition can be used as an extended partition which may itself be divided into several logical partitions. Values of x of 1 to 4 are only used for primary partitions. Logical partitions are always numbered 5 and higher.

Dr. Linux: The checking of a Linux ext2 file system is normally performed every twentieth boot-up. The interval can be changed using */sbin/tune2fs*.

Before you can use this command line tool you must know which **partition** contains the file system in question. If the partition is one that is automatically mounted by Linux at start-up you can find this out by typing the command *mount* without further options. If you only have one Linux partition, the only "genuine" file system is that attached to the root directory "/" in the list:

```
user$ mount
/dev/hda6 on  /  type ext2 (rw)
proc on /proc type proc (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=0620)
```

In the file */etc/fstab* ("*Filesystemtab*le") is a list of which partition is automatically mounted where. Here the root directory "/" is listed with its file system:

```
/dev/hda5   swap    swap     defaults   0  0
/dev/hda6   /       ext2     defaults   1  1
/dev/hdc    /cdrom  iso9660  ro,noauto,user,exec 0  0
```

In this example the root file system is quite clearly located at */dev/hda6*. The command to change the check intervals could therefore look as follows:

```
root# tune2fs -c 5 /dev/hda6
```

The option -c sets the maximum number of mounts that may occur between two file system checks. The value 5 sets the number of boot processes that may occur until the next test. In this example the system would in future be checked on every fifth boot-up.

When you run this command *tune2fs* confirms its execution like this:

```
tune2fs 1.17, 26-Oct-1999 for EXT2 FS 0.5b, 95/08/09
Setting maximal mount count to 5
```

If your system was not properly shut down, an automatic check of the file system occurs as if you have

set the system check interval to *0*. In addition to the maximum number of mounts which are set using *tune2fs*, the **valid bit** is stored in the **superblock** of an ext2 file system.

When the system is booted up the file system is checked by running the program *e2fsck*. If the valid bit it is set to 0, *e2fsck* tries to save what data can be saved by performing an automatic repair of the file system. Before you completely suspend the checks of your system using *tune2fs*, you should be aware of what *fsck* and *e2fsck* do.

- Initially, a comparison of the information in the superblock occurs with the current state of the system.
- A check is made as to whether every **Inode** entry is valid and can be allocated to a folder entry.
- Then a check is made on whether the pertinent data blocks exist for all files and are unambiguous.
- The link number in all folders is compared with the internal link counter in the inode.
- Finally, the check takes place for whether the whole number of all blocks is equal to the number of occupied plus free blocks.

You may find all of this terribly theoretical and boring: if so, the best thing would be to just allow Linux to continue to check the file system regularly.

Another reason for leaving things as they are is that you can not just decide to check the file system whenever you like. If the partition is writably mounted, no file system check may take place. Your system checks the root directory when it only has read entitlement, which you can easily observe on system start-up:

```
[...]
Partition check:
hda: hda1 hda3 <\<> hda5 hda6 >
VFS: Mounted root (ext2 file system) read only.
Freeing unused kernel memory: 60k freed
INIT: version 2.76 booting
Running /sbin/init.d/boot
Mounting /proc device                    done
Activating swap-devices in /etc/fstab...
Adding Swap: 128484k swap-space (priority -1)
                         done
Checking file systems...
Parallelizing fsck version 1.17 (26-Oct-1999)
/dev/hda6: clean, 139696/1038336 files, 222532/4152771 blocks
       done
Mounting local file systems...
[...]
```

Only after this has been done are the partitions fully (i.e. both readably and writably) mounted in the system. Note how much more sensible this is than the situation under Microsoft Windows 9x, which cannot prevent writes to disk from occurring whilst a file system check (ScanDisk) is run, and which must, therefore, restart the entire check each time a disk write occurs.

Dr. Linux recommends that you should only try to check a file system "manually" after the man pages for *tune2fs* and *e2fsck* have been read and fully understood.

Lost Property Bureau

On my system I have discovered the *lost+found* folder. But there is nothing in it. What is it for?

Dr. Linux: A folder called *lost+found* exists on all ext2 file systems. Files are moved here which have "failed" in a file system check. This is to give you the opportunity to try to save something of corrupted files.

For example, the folder entry of a file may be missing but its content may still be present. Such faults in the file system are caused when the system has not been shut down correctly, as in the case of a power failure or accidentally pressing the reset button.

Because information may be cached and only written back to disk when the system is shut down your Linux system should *never* be shut down or restarted except by means of the *shutdown* command.

Nocturnal activities

I'm finding that at regular intervals my hard disk suddenly bursts into activity. Using *top* I have ascertained which processes are running. In particular I have found that *find* and *mandb* run. What irritates me is that *root* is obviously carrying out automatic actions whilst I am not logged in at all as *root*.

It also bothers me that in the middle of the night the hard disk whirrs on account of a *find* process (which is logical, as something is being searched for). My question is: How can I change or cancel these automatic processes? Do they serve a useful purpose?

Dr. Linux: Such automatic processes are executed by the program *crond* (often also called just *cron*). The *Cron D*aemon is a program which is started when the system is booted by the *cron*(*d*) – sometimes also *cron.init* – initialisation script in one of the folders */etc/rc.d/init.d*, */etc/init.d* or */sbin/init.d*. If you would like to know the precise origin of your daemon enter the command:

```
root# locate cron
```

The program cron(d) runs – like all daemons – in the background and checks certain files in the system to see if tasks are stored there which are to be executed at set times. Look in your folder /etc. There you will find the file /etc/crontab (also called System-crontab) and usually the following folders:

- cron.daily: Here the executable scripts for daily tasks are stored.
- cron.hourly: In this folder lie scripts for tasks which are to be processed hourly.
- cron.weekly: Scripts of tasks to be executed weekly are stored here.
- cron.monthly: Store for scripts which contain tasks to be executed monthly.

The tasks or cron-jobs here will be carried out at the times indicated in the /etc/crontab. The /etc/crontab of

Superblock: *When the computer is switched on, the first block of a partition or diskette is read and evaluated. This block may contain a program for loading an operating system, a so-called boot loader. For this reason this block is termed the boot block. The real file system begins with the second block; this is the so-called superblock.*

Valid bit: *In the superblock of an ext2 partition, the valid bit is deleted or set at 0 before the partition is mounted. When shutting down the system, all files in the main memory are saved and the partition is unmounted from the system. The valid bit is stored again or set at 1. The next time the system is booted up, its value is checked again. If it is set at 0, this means that the system has not been properly shut down. A check and repair is then initiated by running the program e2fsck.*

Inode: *An inode is a data structure in which information such as the size, properties, group and access rights are kept for all files, directories and links in the system. In addition, each inode contains seven pointers (or "references") to data which belong to the file, folder or link.*

■

an SuSE Linux system differs quite considerably even at first glance from that of a Red Hat Linux distribution.

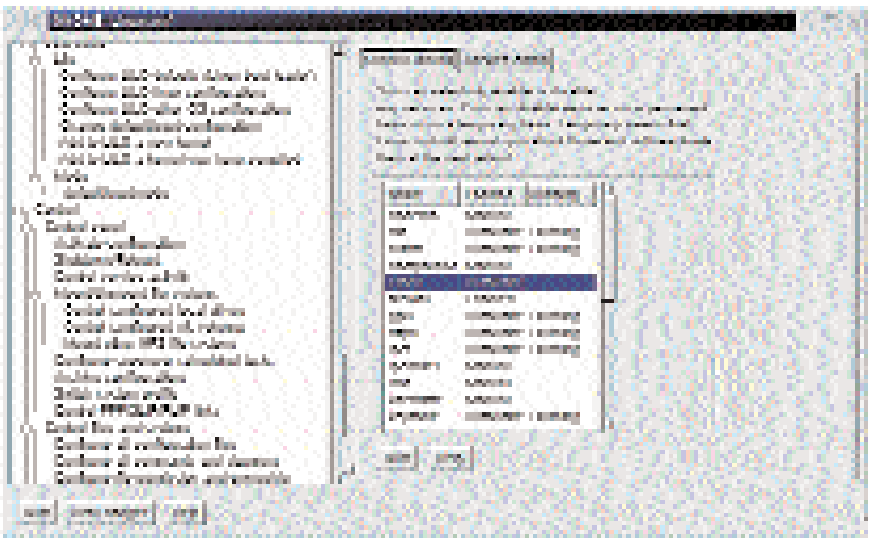
For example, the /etc/crontab in the case of SuSE 6.3 in extract looks as follows:

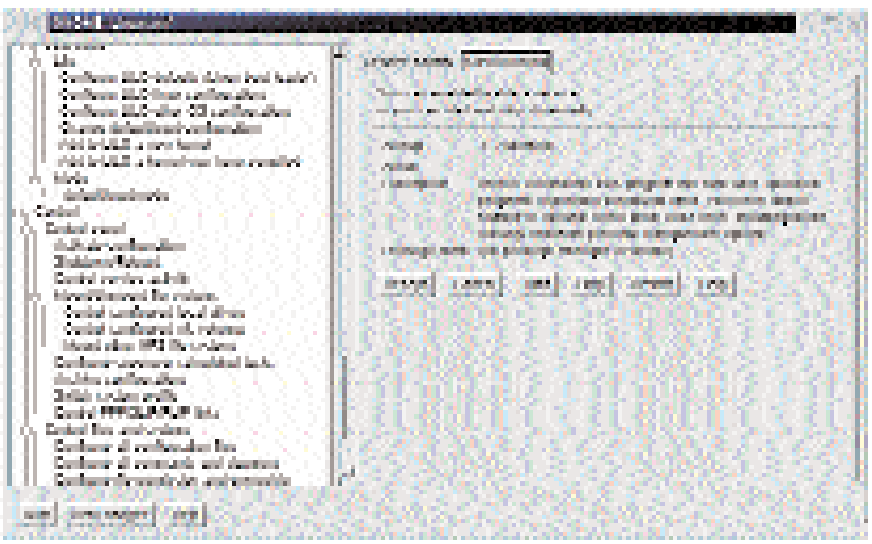
```
# check scripts in cron.hourly, cron.daily, cron.weekly and cron.monthly
#
-*/15 * * * *   root  test -x /usr/lib/cron/run-crons && /usr/lib/cron/run-crons
0 0  * * *    root  rm -f /var/cron/lastrun/cron.daily
0 0  * * 6    root  rm -f /var/cron/lastrun/cron.weekly
0 0  1 * *    root  rm -f /var/cron/lastrun/cron.monthly
```

… whereas the */etc/crontab* of Red Hat has the following entries:

```
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Let's look at at the format of an */etc/crontab* with all the puzzling asterisks and figures.





Figs. 2a and b: The out for cron?

Fig. 1: Set up cron jobs under root using *linuxconf*

Each line is divided into three areas: when, who, and what. In the "when" area the time indications are set separately in each case by *blank* in the sequence minute(s), hour(s), day, month, weekday. An * stands for every possible value in this field. The following are permitted:

- the minutes *0–59* as well as *. An entry like *0-59/15 * * * *** causes the subsequently set command to be executed every 15 minutes.
- the hours *0–23* and *. Here too you can indicate areas: *0 13-18 * * * *** results in the following command being executed each full hour between 13 and 18 hours. *0 10,12,16 * * * **, on the other hand, executes the appropriate cron job at the full hour at 10, 12 and 16 hours.
- the days of the months *0–31* and *. *15 12 6,20 * ** stand for the 6th and 20th of each month at a quarter past 12.
- the months *1–12* and * as well as
- the days of the week *0-7* and *, it being possible to identify Sunday either by 0 or by 7.

If you have been paying attention you will have noticed that the "-" at the start of the example SuSE *crontab* was not explained. If you use a different distribution perhaps you have already searched unsuccessfully for an explanation of it in the man pages. No wonder, for on most Unix and Linux systems this is an invalid entry. Information on its purpose can, however, be found in the SuSE man page for *crontab(5)*. The "who" area of the file is separated by blanks from the "when" area, and contains the name of the user as whom the cron job will be carried out. After more blanks the "what" area begins. If, for example, you wish to download your news to a local news server using the program *fetchnews* you would enter here the full path to the command, for example: */usr/sbin/fetchnews*. A completed entry in */etc/crontab* which causes the user *news* to fetch news articles Mondays to Fridays at 20.00 hours would look something like:

```
0 20 * * 1-5   news  /usr/sbin/fetchnews -vv
```

Note that the user specified must have the rights to carry out the command assigned to it.

If no network connection can be established at the scheduled time, *cron* sends you a mail:

Fig. 3: Finding the settings for cron is not quite so simple.

Fig. 4: You configure the scripts in cron daily with Yast like this

```
Date: Sun, 2 Jul 2000 11:00:01 +0200
From: Cron Daemon  root@max>
To: root@max
Subject: Cron  news@max>  /usr/sbin/➥
fetchnews -vv

1.9.4: verbosity level is 2
Trying to connect to news.myisp.net ... fai➥
led.
```

But a successful message is also delivered:

```
Date: Sun, 2 Jul 2000 13:00:03 +0200
From: Cron Daemon  root@max>
To: root@max
Subject: Cron  news@max  /usr/sbin/➥
fetchnews -vv

1.9.4: verbosity level is 2
Trying to connect to news.myisp.net ... co➥
nnected.
Getting new newsgroups from news.myisp.net
Read server info from /var/spool/news/leaf.n➥
ode/news.myisp.net
comp.os.unix.linux.newusers: considering ar➥
ticles 128807-128875
comp.os.unix.linux.newusers: 69 articles fet➥
ched, 0 killed
[...]
Disconnected from news.myisp.net
```

In each case you can read in the executable scripts in the folder */etc/cron.daily*. These are the automatic processes that make your hard disk whirr daily. On most systems only *root* has the right to read these files.

By entering:

```
user$ top
```

you can follow what processes are started and finished on a command line while the cron jobs are running. Depending on your system and the way it has been configured you may observe some of the following actions:

- *mandb* renews the manpage database.
- The folder */tmp* is emptied.
- *updatedb* updates the database which the search command *locate* accesses. *locate* works similarly to *find*, only more quickly because it accesses a database and does not examine the system directly. In order to obtain a correct result from search queries, this database must be up to date.
- The *log*(book) files are monitored so that they do not grow endlessly.

Depending on the system and its security settings, the cron jobs (pre)configured for you are executed by the user *root* or *nobody*.

Time savers

Is there no simpler way of editing the system crontab?

Dr. Linux: If you use *linuxconf* while running as *root* you can select *System->Cron jobs* and use this to enter new tasks. Under *System->Services* you can also select *crond* from the list and completely deactivate the cron daemon. (If you run the SuSE distribution you can also configure *cron* using *Yast*.)

Deactivating the daemon is advisable if you are running Linux on a laptop on battery power mode or when burning CDs.

Figure 4 shows where you can change the settings under SuSE Linux using *Yast*. After *Administration of the System* choose the entry *Change configuration file* (Figure 3). There you determine what tasks the scripts in the directory */etc/cron.daily* should process:

- *CRON*: Here you specify whether the cron daemon is activated or not.
- *MAX_DAYS_FOR_LOG_FILES*: How long do you want to archive the automatically compressed *log* files for before they are deleted?
- *MAX_DAYS_IN_TMP*: Insert here how long unused files may remain in */tmp*.
- *MAX_DAYS_FOR_CORE*: How long may **Core files** occupy space on the hard disk before they are deleted?
- *RPMDB_BACKUP_DIR*: Inform the system here where the backups of the **RPM database** should be stored, or whether there should be backups at all. With *MAX_RPMDB_BACKUP* you specify the maximum number thereof.
- *RUN_UPDATEDB*: If you would like to renew the database for *locate* daily, enter *yes* here.
- *REINIT_MANDB*: A *yes* at this point causes the manpage database to be updated daily. ■

Core files: *If a process terminates unexpectedly a core dump is produced. This is intended to help programmers reproduce the fatal moment when faults are experienced. For non-programmers these files only use up storage space unnecessarily.*

RPM database: *When installing, updating and uninstalling packages distributions like SuSE, Red Hat, Caldera or Mandrake update a database that contains detailed information about the software installed on the system.*

■