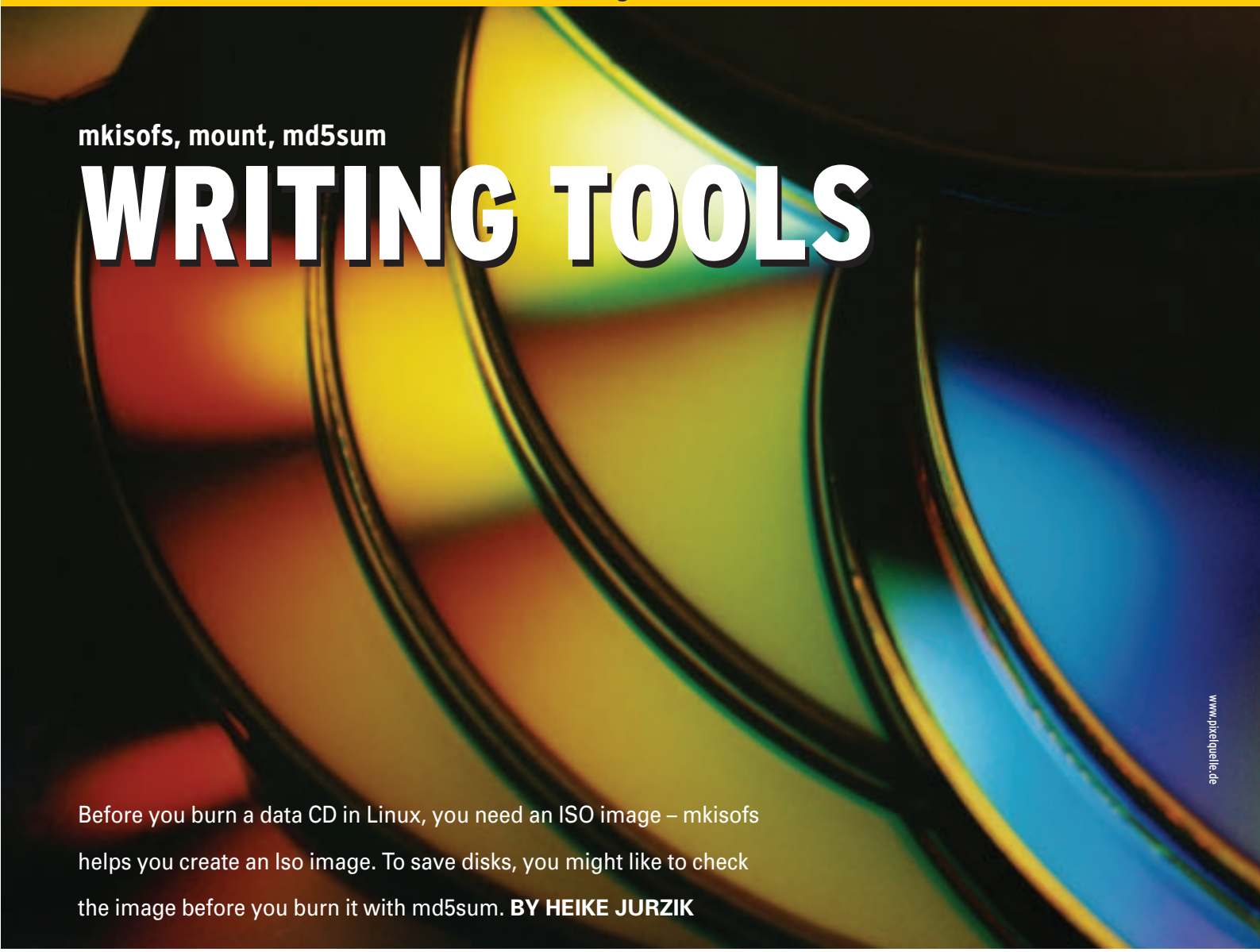

mkisofs, mount, md5sum

WRITING TOOLS

www.pixelquelle.de

Before you burn a data CD in Linux, you need an ISO image – mkisofs helps you create an Iso image. To save disks, you might like to check the image before you burn it with md5sum. **BY HEIKE JURZIK**

CDs and disk partitions use different filesystems. Whereas Linux may use the Ext3 or ReiserFS format for partitions, CDs use the ISO9660 filesystem. It doesn't make much sense to burn a partition block by block onto a CD. Instead, you will need to run the *mkisofs* tool first to create an ISO9660 filesystem where you can store the required files. Having done so, you can burn the "ISO image" onto disk using the *cdrecord* tool.

This issue of Command Line takes an in-depth look at command line tools for burning and verifying CD images.

Creating ISO Images with mkisofs

Linux offers multiple solutions to the problem of creating **ISO** images. The command line gives you the *mkisofs* ("make ISO 9660 filesystem") tool and *dd* (see the box titled "Creating ISO Images with dd"). *mkisofs* expects you to specify an output file (*-o* parameter) and the source files from which to create the image:

```
mkisofs [parameter] -o file.iso ➥
  /directory/with/data
```

As an optional parameter, you can specify *-r* (Rock Ridge) and *-J* (Joliet extensions) to ensure that the CD will be readable on other operating systems. If you specify *-R* instead of *-r*, the tool will still use Rock Ridge, but file ownerships and permissions are kept. This setting helps if you are backing up your own files:

```
$ mkisofs -J -R -o backup.iso ➥
/home/linux-magazine
  8.56% done, estimate finish ➥
Wed Feb 23 13:53:32 2005
 17.10% done, estimate finish ➥
Wed Feb 23 13:53:32 2005
[...]
```

The progress indicator tells you how far along *mkisofs* is with the job. If you prefer not to watch the indicator, you should specify *-quiet* to tell *mkisofs* to keep quiet. The *-V* ("Volume ID") option allows you to specify a label for the CD:

```
mkisofs -V "LinuxMag Backup ➥
02/2005" -J -R -o backup.iso ➥
/home/linux-mag
```

Exclusions

To explicitly exclude files from a backup, you can specify the *-m* option, which

GLOSSARY

ISO: The International Organization for Standardization (ISO) defines technical standards for various fields. The ISO 9660 standard has established itself as the standard for managing files on CD-ROMs. Thanks to standardization, CDs can be read on many different operating systems. Originally, the standard only supported short filenames, but the Rock Ridge and Joliet extensions removed this restriction.

supports wildcards. You can string multiple *-m* parameters together, as in the following command

```
mkisofs -J -R -m '*.jpg' -m➥
 '*.png' -o backup.iso ➥
linux-mag/commandline/2005
```

which prevents files with the *.jpg* or *.png* extensions from ending up in the ISO image. There is a caveat: if you are creating an image of the files in the current directory, you can't use the * wildcard to refer to all files. The following command

```
mkisofs -J -R -m '*.jpg' -o ➥
backup.iso *
```

would actually include all JPEG files in the directory. You should use the dot "." as a shortcut for the current directory instead; *mkisofs* will work as expected in this case and not place any graphics files in the image.

The same thing applies to the *-no-bak* option, which is designed to exclude files with extensions typical of backup programs such as ~, # or *.bak*. Do not use the asterisk; instead use a path name to exclude any files you do not want to put in the image.

-x <path> allows you to exclude a whole directory. Path names for the directories can be absolute (such as */usr/local*) or relative (such as *../../usr/local*) – again, multiple selections are permitted:

```
mkisofs -J -R -x ➥
/usr/local -x ➥
../../usr/games -o ➥
backup.iso /
```

Creating ISO Images with dd

dd is the "convert and copy" program; if you recall, the more obvious abbreviation, *cc*, had already been assigned to the C compiler). *dd* transfers data between various storage media, converting the data if required. *dd* can copy single files and also access devices. This allows you to copy whole hard disks (partitions) and even CD-ROMs to image files.

When you call *dd*, you need to specify the "infile" (*if=*) and the "outfile" (*of=*). The tool then reads blockwise from the source and writes the data to the target. You can optionally specify the block size and the number of blocks. The following command

```
dd if=/dev/hda of=bootsector ➥
bs=512 count=1
```

tells *dd* to read the first block (*count=1*) from the first IDE drive (*/dev/hda*) with a size of 512 bytes (*bs=512*) and to write it to a file called *bootsector*.

To create an image of a data CD with *dd*, you need the CD-ROM device file as the input (e.g., */dev/hdc*) and the filename as the output:

```
dd if=/dev/hdc of=cd.iso
```

Verification

There are a number of options for verifying ISO images. First of all, you might like to mount the image on your filesystem, just to make sure.

Advertisment

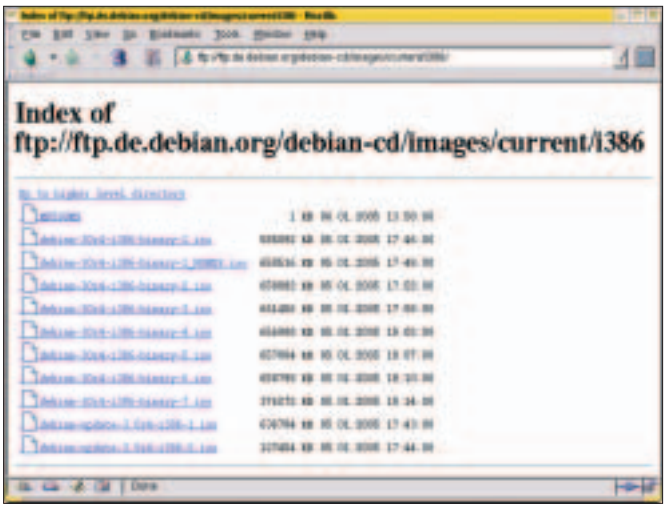

Figure 1: Verifying the MD5 checksum helps you ensure that the downloaded image file is clean.

The *mount* command not only mounts media such as hard disks and CDs on the Linux file system, but it also mounts ISO images, using the *-o loop* flag. As an administrator, you can use the following command to mount an image

```
mount -o loop file.iso /mnt/tmp/
```

Of course, the */mnt/tmp/* directory needs to exist (as usual); if not,w *mount* will complain:

```
mount: mountpoint /mnt/tmp/ ➥
does not exist
```

If *mount* complains about an unknown filesystem instead, you may need to specify the *-t iso9660* parameter. The */mnt/tmp* directory should give you the full content of the ISO image – exactly as it will appear on the CD. To unmount the image – again working as the root user – issue the following *umount* command:

```
umount /mnt/tmp
```

Validated

If you download an ISO file off the Internet, you might like to verify the file's **MD5 checksum**. Linux gives you the *md5sum* tool, which creates a 128-bit hash of the file. A checksum allows users to ensure that messages are transmitted correctly – the sender and the recipient both calculate the checksum, and, assuming the file remains undamaged in transit, the results should be identical. MD5 is a checksum algorithm that was developed in 1991 by Ron Rivest [1].

Most download servers give you MD5 checksums for ISO images (Figure 1). Text files, typically called *MD5SUM(s)* or *xyz.iso.md5*, store the hashes for the image files on the servers. As the checksum changes if a single bit in the file is modified, it is extremely difficult to generate two different files with the same checksum. In other words, it is extremely unlikely that a file would generate the same checksum if it had been manipulated or damaged in transit.

You can run the *md5sum* command line tool to verify the MD5 fingerprint. If you simply want to output the hash value, type *md5sum file*, and the checksum will be displayed on screen:

```
$ md5sum FC3-i386-DVD.iso
ca49964739f84848ca78fc036➥
62272fb FC3-i386-DVD.iso
```

If you are burning ISO images with K3b, you can save yourself the trouble of manually outputting the checksum at the command line. After opting to burn a CD or DVD image in the *Tools* menu, K3b will automatically calculate the checksum. The progress indicator gives you the latest status. When it has finished burning the CD or DVD, K3b pops up a dialog allowing you to validate the MD5 sum.

Of course, you could open the checksum files in a text editor, or use *cat* to display them on screen, and compare the digits individually. But as this process is time-consuming and prone to error, you might prefer to run *md5sum* with *-c* instead. To do so, load both the ISO image and the checksum file on your computer and enter the following command:

```
$ md5sum -c MD5SUM
FC3-i386-DVD.iso: Ok
```

The output shows that the Fedora Core 3 image in our example is complete and unblemished – the checksums match. If the check fails, *md5sum* will say:

```
FC3-i386-DVD.iso: Error
```

Well Combined

Command line programs offer you the advantage of being able to use multiple tools in combination. For example, the clever combination of a call to *mkisofs* integrated with an ensuing *cdrecord* command allows you to burn CDs "on the fly" – this removes the need for hard disk space for the ISO image. Leave out the *-o outputfile* option in *mkisofs*, and simply pipe the output directly to *cdrecord*. The only problem with this approach is that you need a buffer to avoid a buffer **underrun** and thus to avoid wasting a disk. Although Linux is well equipped for multitasking, you might wish to add a buffer of 6Mbytes to the *cdrecord* command, just to be on the safe side:

```
mkisofs -V "LinuxMag Backup ➥
02/2005" -J -R /home/linux-mag ➥
| cdrecord -v fs=6m speed=4 ➥
dev=0,0,0 -
```

Don't forget the minus sign at the end of this line. This tells *cdrecord* not to read from a file, but from standard input – this should have your data on the disk in next to no time. ■

GLOSSARY

Buffer underrun: While actually burning a CD, data are stored in buffer memory to ensure a constant flow of data. If the buffer runs empty, and new data is not forthcoming, older devices might stop burning and output a "buffer underrun" error – at the same time, this destroys the new CD. More modern devices with "BurnProof" technology actively prevent buffer underruns. If the data flow is interrupted, the drive simply waits for new data, and then launches back into the operation where it left off.

INFO

[1] RFC 1321 "The MD5 Message-Digest Algorithm": *http://www.ietf.org/rfc/rfc1321.txt*