Access to Zip and Jaz Media Made Easy

IOMEGA TOOLS

HANS-GEORG ESSER

Iomega's Zip drives have become a popular storage medium, not least because of the low cost of a 100MB diskette. We take a look at running this handy hardware under Linux.

After the Zip drive came the Jaz drive with 1GB of storage, later a 2GB variant and then a new version of the Zip drive, which can handle, alongside the old Zip data media, 250MB media too. Common to all Iomega media is the fact that the write-protection is done by the software, hence special tools are needed to set and to remove the write protection. There is also the option of setting a password, without which write access or any access to the medium can be blocked.

Lomega

One representative of the Iomega tools clones is Lomega. This program offers all the functions necessary and combines them under a single, easy interface. But be warned, *lomega* has the SetUID bits set, so is always executed with *root* rights, even when you start it as normal user. The advantage is that you always have access to the device files, but the drawback is that you have to trust the program.

As usual, Lomega has to be compiled before use. The program is not especially demanding, when it comes to existing libraries, so no major problems should arise here. Unpack the archive *lomega-0.2.tar.gz*. In */usr/local/src*, switch to the new sub-directory *Lomega-0.2* and there execute the classic three-step of *./configure*, *make* and *make install*. After that the program should be found and started under */usr/local/bin/lomega*.

On first starting, *lomega* scans the SCSI bus and displays the Zip and Jaz drives found. A sample output is

```
Welcome hgesser, starting scan...
IOMEGA  ZIP 100
```

The seven buttons have the following functions:

- *Info* outputs the current status of the drive and any medium that has been inserted
- *Mount* mounts the medium in the mount point which, depending on the type of device, is either */mnt/zip* or */mnt/jaz*. These mount points can, however, be altered in the configuration file */etc/lomega.conf*
- *Unmount* cancels this
- *Eject* ejects the medium
- *Lock* provides the medium with a write or read/write protection, whereby a password can be issued – this is needed to cancel this protection again later on (Figure 2)
- *Un-Lock* cancels this
- *Back-Up* backs up a marked directory on Zip. This directory must first also be selected in the tab, also named *Back-Up*, under which a file manager is hidden

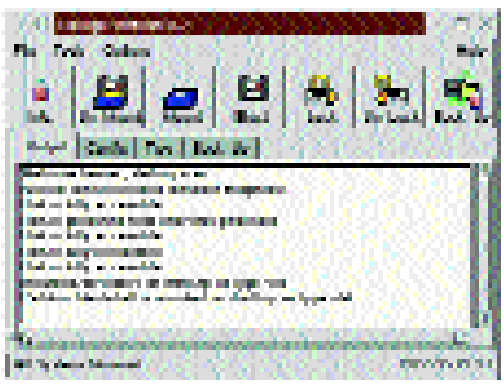

Figure 1: Lomega is the most powerful tool in the comparison

SCSI preferred

Many of the tools described here prefer to work with the SCSI drives. ATAPI and parallel port drives can, in some cases, be persuaded to co-operate – reference is made to this in each case.

As support for parallel port drives Lomega offers the two options *File/modprobe/ppa* and *File/modprobe/imm*. *imm* is a driver for the "Iomega MatchMaker parallel port SCSI host adapter", which is used in the Zip Plus drive; *ppa* supports the normal parallel Zip drive. ATAPI drives are only supported if the IDE-SCSI-Emulation in the kernel has been activated. The requisite kernel module is called *ide-scsi.o*.

gtkZip

One alternative to Lomega is gtkZip. The program has a modern design. Unfortunately it refused to work on our test machines.

jaZip

jaZip is not dissimilar to Lomega, and on closer inspection one notes in Lomega's credits the reference, "Portions of code by: Jarrod A. Smith". This is the author of jaZip. So it's no wonder that the functions of jaZip are similar. But jaZip cannot create a read protection, which is why we have devoted more time to Lomega.

Incidentally, the following behaviour of jaZip was somewhat problematic: A Zip medium was already mounted in */mnt/zip* when jaZip was started. After a click on the unmount button jaZip then wrote "Unmounting and removing /mnt/zip" - not very nice, since the directory later had to be made again by hand. Before starting jaZip, you should take a look in its configuration file */etc/jaziptab*, which has a similar structure to *fstab*.

The sources would not compile on a current SuSE system; but we were able to install an RPM package, which was found at *ftp://ftp.rpmfind.net/linux/contrib/libc6/i386/jaZip-0.22-4.i386.rpm*, without any problem.

ziptool

Back to the command line. Everything you can do by a mouse click can also be performed in the console and thereby be built into its own shell scripts. The useful tool *ziptool* supports all special functions of the Zip and Jaz drives.

Firstly, *ziptool* has to be compiled, if you are not installing any binary packet – in this case this goes very quickly. Simply unpack, as *root*, the *ziptool-1.3.tar.gz* package, change to the new directory and there call up *make*. Another *make install* copies binaries and man pages to */usr/local*. A look at the man page of *ziptool* (which can also be addressed by symbolic link as *jaztool*) gives away the general syntax *ziptool -Option Device*. At this point, you should always specify as device the device file belonging to the Zip drive, not that of a data partition, thus */dev/sdd* and not */dev/sdd4*.

There now follows a list of the possible *ziptool* commands. For greater ease of reading the device */dev/sdd* is always specified, which is to be replaced by the correct name.

- *ziptool -e /dev/sdd*: eject; eject medium. This only works if the Zip is not mounted – otherwise the error message "Device is mounted" appears
- *ziptool -s /dev/sdd*: status; states the protection status of the Zip medium (thus read or write-protected, with or without password)
- *ziptool -ro /dev/sdd*: read-only; the medium is write-protected. It must not be mounted for this. This protection can be cancelled by any user, since no password has been issued
- *ziptool -rp /dev/sdd*: read-only, password; as with "-ro", but a password is requested, without which the protection cannot be cancelled:

```
[esser@dual ~]$ ziptool -rp /dev/sda
Password: test
ziptool: medium is password write-protected.
```

If you forget your password, you will have to format the medium in order to be able to write on it again.

- *ziptool -rw /dev/sdd*: read-write. This command makes the medium generally accessible again. Any write protection is cancelled. If a password has been used for protection, it must be entered here, otherwise the barrier will not be raised.
- *ziptool -ud /dev/sdd*: unlock door; mounting a Zip medium normally causes it to be locked into the drive. Pressing the Eject button will not work until the file system is released with *umount*. This allows the lock to be released without unmounting first – if that's what you want
- *ziptool -ld /dev/sdd*: lock door – cancels the unlock, locking the drive again
- *ziptool -m /dev/sdd mountpoint*: mount – mounts the medium. In principle this is nothing but *mount /dev/sda4 mountpoint*, but can be executed by any user, as *ziptool* always runs with *root* rights
- *ziptool -u /dev/sdd*: unmount – removes the file system from the file tree again

As already mentioned in the listing, *ziptool* is executed with administrator rights, for which it has set the SetUID bit:

```
[esser@kira iomega]$ ls -al `which ziptool`
-rwsr-xr-x  1 root  root  9708 Jun  1 16:10 /
usr/local/bin/ziptool
```

This means that with the aid of *ziptool*, any user can directly access all device files, even if these would not allow the user themselves access.

GUI or bash#

As usual, it is purely a matter of taste as to whether tasks are processed on the command line or in the graphical window – in the case of the Iomega tools, though, it is interesting that *Lomega* is the only tool which can also set up a read protection. This makes it preferable to *ziptool*. ■

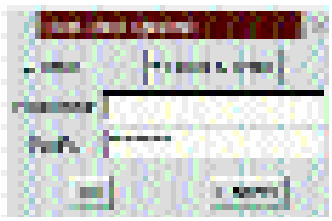

Figure 2: Lomega sets a read/write protection with password for the Zip medium

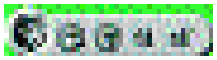

Figure 3: gtkZip's chic design means it is compact and utilises desktop space well

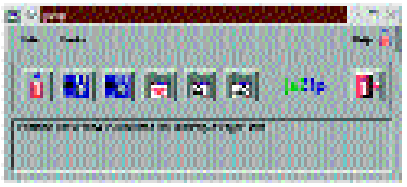

Figure 4: jaZip is similar to Lomega