(*eg* Gnome and KDE libraries) are rebuilt, the packagers need to keep track and make sure their pet projects are constantly in sync. For instance, if the person who's responsible for packaging GTK releases a new .deb of that GUI toolkit, those packaging GTK-based apps need to ensure that their software is compatible and runs smoothly.

Although Ubuntu is built from packages, a handful of them aren't just standalone programs, though, and make invasive changes to the system. Take the startup (*init*) scripts, for example – they exist in .deb form, but their contents and operation can make drastic changes to the way Ubuntu works. The same goes for kernel packages, which can enable or disable critical parts of the system (eg hardware drivers). As Zimmerman explains, such changes are thoroughly reviewed: "Invasive or contentious changes are generally discussed informally in the community, and issues resolved through consensus. In cases where a clear consensus is not reached, the Ubuntu Technical Board can make a final decision on technical issues."

Developers keep uploading their packages with new software versions or fixes, and all of this work is in the open, ready for anyone to try. Many keen Ubuntu fans "chase the development branch" – geektalk for constantly updating to the very latest versions of packages. It's risky business, but eager users can find bugs and help to resolve them You may have heard the term "triage" before in a bugfixing scenario, but maybe aren't sure what it means. When users submit bug reports about Ubuntu, they have to be triaged: a system that determines how severe they are and what needs doing.

My favourite Gutsy feature: Daniel Holbach

"*Compiz*: I've used it for weeks now and it's amazing the way how slick animations and window effects give you a completely new look and feel and way of working. Also expect lots of improvements and good stuff on the Ubuntu Server (*AppArmor, EBox, etc*) and a just awesome Ubuntu Mobile edition."

But there's a but. Most Ubuntu developers spend their time packaging up other hackers' software, be it a tiny text-mode email client or a mammoth suite such as *OpenOffice.org*. So, the Ubuntu team has to balance work from the 'upstream' (original developer) source with the distro's goals. Zimmerman says: "[it's] challenging where a project is dependent on activity happening outside of Ubuntu itself; we are often dependent on certain enabling work being done in upstream open source projects." If a program maker messes up his/her software and adds truckloads of bugs, the Ubuntu packager has to decide whether to accept the new version or stick with the older (and more stable) version, risking outcry from bleeding-edge-demanding users. Conversely, sometimes the Ubuntu packagers are waiting desperately for new features, with the every-six-months release clock ticking in their ears.

Packaging and testing

This process of packaging new software, fixing bugs and adding changes to the base system continues for one to two months, before the team has something concrete to show the world. Every Ubuntu user can download the latest packages and test them at any point, but to make things easier for the wider Linux-using world, the Ubuntu team puts together snapshot releases of the work-in-progress. Like a final release, these are standalone, installable CDs, but they're targeted at developers only – Ubuntu hackers dissuade casual users from trying the snapshots.

Each distro has a codename for its snapshot distro releases: Feisty Fawn (7.04) had "Herd" snapshots, while the upcoming Gutsy uses the codename "Tribe". As development progresses, the team churns out successive "Tribe" snapshots (Tribe 1, Tribe 2...) to demonstrate new features and gather bug reports. These are vital in quantifying feedback on the distro: what's going well, what needs fixing, and what needs to be cut to make the release date.

Then, around six weeks before the final release is due, the rampant development halts: new features can't be added, drastic changes can't be made. It's time for a rush of bugfixing and release candidates...

Ubuntu hackers dream up ideas and hack code at the Developer Summit in Seville.

The release process: a timeline

MONTH 0

- New ideas drafted
- Feature goals set
- Development underway

Snapshot releases

Snapshot releases make up the bulk of the development schedule, typically arriving around once every three weeks. The events here are planned out at the beginning of a release, but the exact dates can change as bugs appear or feature goals are modified.

- Release candidates
- Final sent out to mirrors
- Release announced

MONTH 6

Building your own Ubuntu

Here's how to make your own customised version of Ubuntu, with only the packages you want – the same way we do it for the Linux Format DVD.

It's not difficult to remaster Ubuntu, providing you understand the distro's workings and aren't afraid of the command line. Creating a customised version is also a great way to get a feel for distro building – you can see the complexities that lie underneath the Gnomey surface of an Ubuntu release. The following guide shows you how to extract an Ubuntu CD, add and remove the packages you want, and then assemble it back together again. This is one of the techniques we use to make the super-mega-enhanced versions of Ubuntu on the LXF DVD (*eg* issues LXF94 and LXF88)! Please note that you should have at least 3GB of free hard drive space to follow these steps.

1 Get the ISO

To rebuild Ubuntu, we need an image file of the Desktop CD (*ie* the one that runs in Live mode, rather than the text-based Alternate CD). You can download the latest stable version from **http://releases.ubuntu.com/feisty/**. Grab **ubuntu-7.04-desktop-i386.iso**, save it in your home directory and rename it **feisty.iso**. (It's about 700MB to download, so a broadband internet connection is pretty much essential here, unless you have the patience of a saint!)

2 Loopback mount it

Now we need to attach this CD image to the filesystem. Open up a terminal and switch to root (**sudo bash**), then enter:

```
mkdir /mnt/loop
mount -o loop feisty.iso /mnt/loop
```

Now the contents of feisty.iso are accessible in **/mnt/loop**. We want to copy these to our filesystem, so create a directory and copy the entire contents as follows:

```
mkdir ubuntu-rebuild
rsync -ax /mnt/loop/. ubuntu-rebuild
```

Once this is complete, the **ubuntu-rebuild** directory will contain the disc contents. You can now unmount the loopback ISO image (**umount /mnt/loop**).

3 Preparation

Next we need to extract the compressed filesystem that's included on the Ubuntu CD; this uses *SquashFS*, available in Ubuntu via the *squashfs* package; you should also install *squashfs-tools*. Loopback mount the compressed filesystem as follows:

```
mount ubuntu-rebuild/casper/filesystem.squashfs
/mnt/loop -t squashfs -o loop
```

Now **/mnt/loop** holds the contents of the compressed Ubuntu filesystem – the one used when you boot up the CD in Live mode. Let's copy this into our home directory, in a new folder:

```
mkdir ubuntu-source
rsync -av /mnt/loop/. ubuntu-source
umount /mnt/loop
```

4 Changing packages

Hurrah: we have everything in place! You can now perform some magic to switch into the Ubuntu distro files as if you were running it natively. This is thanks to a little tool called *chroot*, which changes the perceived root filesystem and therefore lets you 'pretend' that you're in another distro. Enter these commands, the first of which sets up networking inside the Ubuntu filesystem:

```
cp /etc/resolv.conf ubuntu-source/etc/
chroot ubuntu-source
```

Now you're inside the **ubuntu-source** folder, as if it was the root (/) directory. You're essentially running the same distro as supplied on the Live CD, but this time you can modify it! I recommend against deleting packages unless you're absolutely sure what you're doing; some of them are dependencies for critical system packages. But you can start adding programs via **apt-get**, for instance, to add *AbiWord* to your system:

```
apt-get install abiword
```

Programs you add at this stage will appear on the final CD/DVD when we rebuild it. So you could add *Xfce* if that's your favourite desktop environment, or 'build-essential' if you want *GCC* and its cohorts ready to run.

5 Update

Once you're done, enter **exit** to leave the Ubuntu filesystem and return to your normal distro. We now need to generate a list of files that are on the updated Ubuntu image, so enter the following monster command:

```
chroot ubuntu-source dpkg-query -W --
showformat='${Package} ${Version}\n' | grep -v
deinstall > ubuntu-rebuild/casper/filesystem.
manifest
```

We also need to tell the Ubuntu installer, when it is run, to avoid certain packages. For instance, after installation, you don't want the 'Install Ubuntu' icon on your desktop! So we *sed* through the list of installable files and remove various Live CD-only components:

```
cat > /tmp/sedscript <<END
/casper/d
/libdebian-installer4/d
/os-prober/d
/ubiquity/d
/ubuntu-live/d
/user-setup/d
END
sed -f /tmp/sedscript < ubuntu-rebuild/casper/
filesystem.manifest > ubuntu-rebuild/casper/
filesystem.manifest-desktop
```

6 Remaster

Now we need to rebuild the *SquashFS* filesystem image; this can take up to half an hour...

```
mksquashfs ubuntu-source/ ubuntu-rebuild/
casper/filesystem.squashfs -noappend
```

Then we have to update the MD5 checksums to confirm the disc integrity:

```
(cd ubuntu-rebuild && find . -type f -print0 | xargs
-0 md5sum > md5sum.txt)
```

And finally, the **mkisofs** command which builds a shiny new ISO image:

```
mkisofs -r -V "My Modified Ubuntu"  -cache-
inodes -J -l -b isolinux/isolinux.bin -c isolinux/
boot.cat -no-emul-boot -boot-load-size 4 -boot-
info-table -o newbuntu.iso ubuntu-rebuild
```

Et voila: if all has gone well, you'll now have a file called **newbuntu.iso** which you can burn to a CD-R (or, if it's larger than 700MB, to a DVD). Note that disc images larger than 2GB can have problems running. But if it's smaller than that, burn it, boot it and enjoy!

Help other distros

Not a 24x7 Ubuntu fan, but spurred on to help out your favourite distro? Typically, most projects seek coders, packages, testers and documenters as described for Ubuntu overpage – so the same requirements apply. Get a feel for your distros's community and key players, watch the mailing lists to gauge the level of activity and nature of posts, then start to put forward your own features or ideas.

- **OpenSUSE**: http://en.opensuse.org/How_to_Participate
- **Fedora**: http://fedoraproject.org/wiki/Join
- **Debian**: www.debian.org/intro/help
- **Mandriva**: www.mandriva.com/en/community/contribute
- **Gentoo**: www.gentoo.org/proj/en/devrel/staffing-needs/