

Building a better ubuntu

Gutsy Gibbon is almost here – but how does the Ubuntu team make sure it's still the world's most popular distro? **Mike Saunders** investigates...

Distros come and go like the seasons. WinLinux, AlphaNet, Xpresso and so many others have appeared under our radar, boasting new features and innovations that promised to etch their vendors's name into the slate of Linux history. Yet where are they today? What happened? Why is all the talk now about Fedora, OpenSUSE, Ubuntu and co.?

The answer is simple: community. Bleating about so-called

“Popularity brings its own set of problems – most notably, a high level of expectation from users.”

revolutionary features may generate a few weeks of hype, but it's not going to establish a distro for the long run. So many of the ostensibly promising distros we've seen over the years have ultimately disappeared into nothingness, eliminated by their lack of developer support and community involvement.

But the Ubuntu team gets this. They took the well-established Debian base, added canisters of polish to make it user-friendly, and backed it up with a supremely helpful and lively community. Ubuntu's strength lies in the fact that it rarely attempts to pioneer a world-changing Linux feature – it uses the standard Gnome config utilities and has lagged behind when it comes to drool-inducing eye-candy such as *Compiz*. And yet, it's still the most popular distribution in the world.

Popularity brings its own set of problems, though, most notably a high expectation level from users. The Ubuntu team has to juggle a vast array of demands from users, working out which features deserve inclusion in a new release and fixing as many bugs as it humanly possible. Tough decisions need to be made, arguments break out – and the

occasional basket of dirty washing is aired due to the open development process.

So, how does the Ubuntu team – and father company Canonical – bring together all these elements and produce a new distro release? Over the next seven pages, we're going to delve into the Ubuntu development process, looking at the upcoming Gutsy Gibbon (7.10) release and seeing how it progresses from feature ideas into usable code. With the help of some key Ubuntu hackers, we'll see what it takes to make a successful Ubuntu release – even if you don't use the distro, you'll get an insight into the processes and people involved in building a new Linux flavour.

Note that on our cover DVD we have the latest development snapshot of Gutsy (a 'Tribe' release), so you can see for yourself how the new features are coming along – it boots straight from the disc in Live mode. If you fancy building your own customised version of Ubuntu, we have a guide later on in this feature. But first off, let's look at the embryonic stages of a new Ubuntu release...







The birth

Before the development-fest kicks off, Ubuntu hackers need to decide which features to include.

Imagine yourself as an Ubuntu developer: you've just got the latest, shiny, brand-new release out the door, and Linux users around the world are eagerly installing it. You've grafted away on writing code and fixing bugs, ready for a well-deserved rest – but no! It's already time to get cracking on the next release. Ubuntu follows a six-month release schedule (having deviated once in 6.06 for stability reasons), so the turnaround for getting new features implemented is very tight.

The Ubuntu development team comprises paid and voluntary coders, the former working full-time on the distro and employed by Canonical, while the latter chip in whenever they can. Despite this marriage of paid and unpaid work, there's virtually zero friction between the two groups – especially as Canonical has employed some Debian developers in the past. For many coders, part-time Ubuntu hacking provides a possible career-path towards salaried distro making.

Once a new Ubuntu release has been sent out to the mirrors, the process of choosing features for the next release gets underway. Matt Zimmerman, Ubuntu's CTO and Technical Board chairman, explains the process: "Ideas are suggested by the Ubuntu community and by the developers themselves. If a developer is interested in the idea, they flesh it out into a technical specification, which can then be the basis for a development project."

These feature ideas are assembled in Launchpad, Canonical's developer support site – for Gutsy, they're listed at <https://launchpad.net/ubuntu/gutsy/+specs>. Some of the feature ideas are discussed face-to-face amongst developers, as Zimmerman continues: "Much of the activity happens at the Ubuntu Developer Summit, near the beginning of each release cycle. These specifications are reviewed by a team of technical reviewers to assess their correctness and feasibility."

Priorities, priorities!

Each new feature idea is given a priority (high, medium and low) dependent on how much in demand it is, what effects it will have on the distro's overall stability and whether it can feasibly be accomplished in time. *Launchpad* also provides constant updates on the process of a new feature: developers can see if it's just being started, making good progress or ready for beta testing. They can also see who is working on each project.

But how is the development effort split up? Zimmerman enthuses: "Canonical itself determines which projects its developers engage in, while community developers make their

My favourite Gutsy feature: Mark Shuttleworth



"That would have to be a toss-up between *Compiz*, which gives us a wonderful multi-dimensional desktop experience, and the new Mobile edition of Ubuntu which is designed for internet tablets that let you carry the net in your pocket."

own decisions about how to spend their time. In practice, many of the projects that Canonical developers pursue are also their own ideas, but we also make some top-down decisions." So voluntary developers can cherry-pick their own favourite features to hack on, whereas Canonical adds more direction to its paid team.

Feature proposals come from all aspects of the Linux world: some are additions found in another distros, some are nascent technologies creeping out of a random SourceForge site, and some are simply cool gadgets that may be useful in the next Ubuntu release. Once the feature list is complete, the Development Manager (for Gutsy, Scott James Remnant) sends out a summary to the development mailing list (<http://tinyurl.com/2cxfev>), and the coding begins...



> <http://launchpad.net> helps developers to organise and prioritise Gutsy's new features.

Rising through the ranks

There are two types of Ubuntu developer: MOTU and Core. The first forms part of the 'Masters of the Universe' team, a large group of hackers who maintain software in the distro's Universe and Multiverse software repositories. Universe encompasses packages which aren't deemed important enough for the main base system, but which users may want nonetheless. New developers join the MOTU team, packaging programs in Ubuntu .debs and maintaining them for bug and security fixes. After a developer has proved

capable of producing good Universe packages, s/he can apply to the Ubuntu Technical Board and Community Council for promotion into the Core team. If the developer has done good work: reliable packages, strong communication skills and broad Linux knowledge, s/he may be able to join the Core team, working on critical packages in Main and Restricted repositories (ie many of those shipped on Ubuntu discs). Ubuntu has a code of conduct to ensure that developers help, respect and collaborate with one another.

Coding time

“Making a distro is 1 per cent inspiration, 99 per cent perspiration” – as Thomas Edison would have said if he lived in these Linux times...

Although Gutsy is in the midst of a huge development charge right now, the overall goals for the release have been set, as described previously. These are the key features and changes that the Ubuntu team hopes to include in 7.10 – but as always with a distro release, some may be dropped if they can't be made functional or stable enough by the release date!

» **Desktop Gutsy** will ship with Gnome 2.20, and its Kubuntu variant will have KDE 3.5.7. Those looking to get a glimpse of KDE 4 will be able to install a release candidate alongside the stable version. Gutsy will be the first release to include *Compiz Fusion*, providing swish 3D desktop effects on supported graphics chips. Users will be able to switch back to vanilla window managers with a few clicks. Also, a new *GTK*-based graphical config tool for screen settings (resolution, colours *etc*) is in the works.

» **X.org X**, the display system, will undergo a few changes: version 7.3 will be included, providing hot-plugging of display devices. One of the goals here is 'bullet-proof X' – that is, an X setup that safely falls back to standard video modes when there's a problem, instead of collapsing in a heap on the command line. For the sake of all Linux newbies in the world, we're cheering for this feature here at *LXF Towers!*

» **WinModems** Yes, those nasty software-driven modems, most commonly found on laptops or given away free with cheap broadband connection packages, will be supported. Well, some of them. Gutsy will provide access to some 'restricted' (ie not completely free or open source) drivers to help get people online.

» **Security** AppArmor, a security framework that lets admins lock-down programs and the resources they can use, will be available as an optional extra. This is the system used by Novell in SUSE; Red Hat opts for SELinux instead.

» **Free Flash** Some Ubuntu coders are investigating the possibility of a free Flash player being included. Currently, the most compatible player is the non-free one from Adobe, but the Ubuntu team is considering *Gnash*, a free software alternative. *Gnash* still lags behind the proprietary player, though.

» **Free as in freedom** A technical review is underway on the possibility of a non-restricted Gutsy installation – that is, the ability to install from the CD without copying any non-free components to your hard drive. Some believe this is a more sensible approach than having a completely separate distro for those who want pure open source/free software goodness. This will be called Gobuntu.

» **Flavours** A new variant, Ubuntu Mobile and Embedded Edition, will be available for developers working on internet tablets and other small-scale devices. Also, the Ubuntu Server project will have a new team beavering away on it at Canonical.

So, these are the current feature goals for Gutsy, and by the current rate of development, it looks like most of them will be in place for the final release. At this



» Behold! 3D desktop effects are on the way, thanks to the integration of *Compiz Fusion*.

stage, a few nerves start twitching in the lead developers – after all, news websites and magazine articles (like the one you're reading!) start discussing the new features, so they want to ensure that nothing major is dropped and nobody is disappointed when the final release is cut. Zimmerman describes how this problem intensifies as development ramps up: "The challenge becomes keeping abreast of many parallel projects and their progress, so that we can deliver what we planned on time."

Code frenzy

And the coding begins. Packagers from the Core and MOTU teams start grabbing the latest source code of thousands of programs, adding the .deb build scripts and generating Ubuntu-flavoured packages. Here, the distro is like water, constantly flowing and changing underneath the developers, and as critical dependencies

Key Ubuntu people

Some names of repute in the Ubuntu hacking team:

» **Mark Shuttleworth** Uber-wealthy one-time-astronaut, made his fortune by founding Thawte, and now the self-appointed "Benevolent Dictator For Life". He sets the overall direction of the project and, as a former Debian developer, chips in with some technical decisions too.

» **Matt Zimmerman** Chief Technical Officer. Zimmerman looks after the coding side of the

distro, making sure the development and release processes aren't hindered by major bugs, developer arguments or whimsical feature requests.

» **Brian Murray** Chief bugmaster and manager of the Ubuntu QA team, responsible for squashing as many bugs in the distro as possible before a release.

» **Corey Burger** Head of the marketing team, in charge of contacting media groups with release information and making

sure that the distribution exudes positive vibes.

» **Troy Sobotka** Artwork group lead. His team works on themes and icons to keep Ubuntu looking fresh (well, as fresh as a brown theme can look!) and ensuring that the distribution is visually consistent throughout.

» **Matthew East** Oversees the documentation project, which helps to make the distro more accessible and provide translations for a wide range of languages.

Building a better Ubuntu



(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...

Creative Commons image, RAMONRAMON, <http://tinyurl.com/2z7v5>



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

The release process: a timeline



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}' | 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/sedscrip <<END
```

```
/casper/d
/libdebian-installer4/d
/os-prober/d
/ubiquity/d
/ubuntu-live/d
/user-setup/d
END
```

```
sed -f /tmp/sedscrip < 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
-O 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/



Release time!

The code has been written, the features have been added, and the release deadline creeps ever closer...

Towards the end of the development cycle, things get a little hectic as the number of bug reports starts to swell. Brian Murray heads up the Quality Assurance team for Ubuntu, and monitors the level of feedback from the community. "With a completely open bug reporting system, like Ubuntu has, it seems that we receive the greatest quantity of bug reports after the final version of the release has been made and it becomes more widely used. However, the Desktop CD with its Live environment provides a great way for people to test the development release and contribute to the process."

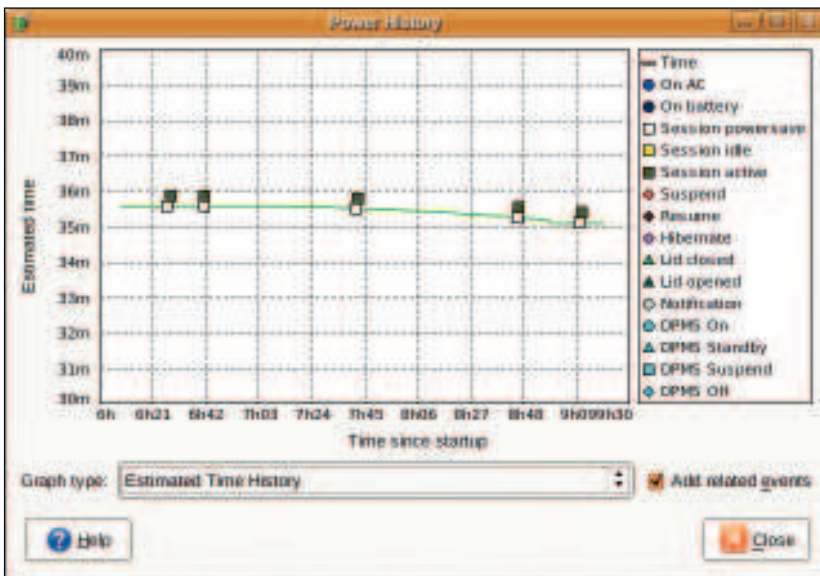
So, it's a catch-22 situation: the Ubuntu team want bug reports to make the final release super-polished, but they'll get the most bug reports when the distro has actually been released! That said, if a major stability problems occur, the release date for snapshots and the final version can be altered: "We would rather provide our users with a great experience than meet a target release date."

This was the case with Ubuntu 6.04 – in order to fix some niggling stability issues, the team pushed the release back two months for some extra work. Consequently, it was numbered 6.06 on launch.

Around six weeks before the release is due, the "Upstream Version Freeze" kicks into effect: no new package versions, either from individual software developers or existing Debian packages, can be added to the distro. So if you're the *AbiWord* packager with 2.4.6 built for Gutsy, and then 2.5.0 comes out, you have to leave it for the next Ubuntu release. Developers can still issue new packages with bug and security fixes, but the goal at this point is to have a known platform that the team can begin to stabilise.

About a month after this, "Feature Freeze" comes into play – from this point onwards, no new features which alter the workings of the distro may be added. As with the version number freeze, this is essential in keeping the distro together: you don't want some developer completely rewriting the boot scripts a couple of weeks before the release. If a planned feature isn't completed by now, it has to be delayed for the following distro release. In rare cases, a developer can ask the Ubuntu managers for an exception

› **Gutsy will feature a new power meter which shows a graph of juice consumption over time.**



My favourite Gutsy feature: Corey Burger



"I'm looking forward to all the server work that is being done. There's a lot to do to make Ubuntu Server as polished and easy to use as the desktop. Work is being done on LDAP libraries and packaging of *Ebox*, a web-based network services control tool."

to this rule, providing that the benefits it brings outweigh the possibility of severe stability impacts. The developer has to clearly demonstrate that his/her new package won't break any other packages that depend on it.

End in sight

Two more freezes follow: artwork and "strings". The latter stops developers from changing text in their programs, and coupled with the former it means that documenters can prepare screenshots and help text without the distro suddenly changing its appearance. One week after the string freeze, and a month before the final version is due, a beta release is produced which contains working – albeit potentially unstable – implementations of the distro's new goodies. Ubuntuers term it "feature complete", meaning that it has everything in place for the final version, but needs testers.

At this point the bugfixing team gets into overdrive, as Murray explains: "As the final release nears more people take it for a test drive and therefore the quantity of bug reports can increase. Subsequently the challenge becomes ensuring that those new bug reports are triaged quickly and then that the high priority ones are identified and resolved."

Aside from the technical issues, parts of the community start working on the distro's image. Press releases need to be prepared, release notes drawn together and screenshot-laden new feature guides need making. Corey Burger is in charge of marketing at Ubuntu: "The marketing team and Canonical usually chat just before release to see what each is doing. That usually happens via myself calling one of the Canonical marketing people and figuring out what they are doing. For the Feisty release, this meant Canonical handled the press release and any incoming press inquiries (at least in the English press) and the community created a lot of the online collateral, such as the 7.04 Features tour."

One week before the final release date, the team uploads an RC (release candidate) build of the distro: potentially the final version unless any last-minute "show-stopper" bugs are found. Development is halted; only fixes for extremely notable bugs will be allowed in at this stage. Most of the community waits anxiously, hoping that no disk-frying nightmares surface on some obscure combination of hardware. If all's well, the team produces a final build of the packages and disc ISO images, sends them to mirror servers around the world, and announces the new release on the Ubuntu website.

Meanwhile, some developers start piecing together plans for the next Ubuntu release. The cycle starts all over again...

Creative Commons image: eikbuntu, <http://tinyurl.com/Zs9sch>

Getting involved

Now that you've got the lowdown on Ubuntu's development process, why not chip in some help with the distro – or indeed any other distro?

As Ubuntu continues to flourish, the project is constantly looking out for new talent – and it's easy to get involved. You don't have to be a long-time coder to help out with the distro, as there are many ways to dive in and add your improvements. A new distro release is the product of many different talents, all working together with a single goal: to make a usable operating system. So even if one aspect of the development process seems alien or intimidating, never fear – you can work on your own projects without having to dip your toes into everything else.

As we've seen, some high-level decisions regarding Ubuntu's direction are made by Canonical, but the vast majority of development activity is open for all to see. Helping out with a distro is rewarding and gives you great experience for the future, so here are some areas worth looking into (and they're equally applicable to other distros such as Fedora or OpenSUSE):

- » **Coding** Creating packages, fixing bugs and issuing security patches is the bulk effort of any distro; it doesn't require extensive programming knowledge though. Basic knowledge of C/C++ and Bash scripting is useful. If you're familiar with the command-line, editing text files and building software from source, you're equipped to produce Ubuntu .deb packages and see if you can work them into the Universe repository. If you're a dab-hand with programming, you can try one of Ubuntu's paid bounty projects.
- » **Testing** Download mid-development snapshots and new packages, and report bugs if they don't work at <http://bugs.launchpad.net>. Keep your bug reports snappy but informative; saying "Doesn't work, please fix" will only annoy developers, whereas listing exact error messages and your system spec will help them to narrow down the problem.
- » **Documentation** One problem with Linux – regardless of distro – is the piecemeal documentation. Coders love to write code, but don't relish the prospect of writing user guides. You can help out by writing a manual for an as-yet undocumented program, or improving the user guides for well-known apps that only have sketchy docs. Useful knowledge here is XML and DocBook.
- » **Artwork** Like user guides, this is something that hackers sometimes forget, leading to ugly or weakly designed interfaces.



» **Coders, testers, artists, documentation writers – many skills are essential in making a well-received Ubuntu release.**

My favourite Gutsy feature: Matt Zimmerman



"I'm personally looking forward to the launch of our mobile product, Ubuntu Mobile and Embedded, which will come as part of the 7.10 release. More about that at <http://wiki.ubuntu.com/MobileAndEmbedded>."

Create icons, images, desktop wallpapers and interface mock-ups to make your distro look smoother and easier to understand. If you can make these graphics using open source software such as *Gimp* and *Inkscape*, even better!

» **Support** Ubuntu's community is up there with Gentoo's as one of the most welcoming and friendly; try helping out at www.ubuntuforums.org by suggesting answers to common user problems and making sure that nobody feels alienated when switching to Linux. Your fuse may be shortened by annoying "ZOMG teh linux doent w0rk Y not"-type posts, but stay friendly and approachable, and you'll earn much kudos in the community.

» **Localisation** As highlighted by Ladislav in this month's Distrowatch on page 38, in many parts of the world, due to the requirement of knowledge of a second language, the benefits of computer use are denied to much of the populace. If you're bi- or multi-lingual, particularly with knowledge of a language that isn't supported by proprietary OSes, your help can be invaluable in every area, especially support, translating existing documentation and advising GUI developers on interface updates.

If you want to dive headlong into Ubuntu development, hoping to work your way into the MOTU and Core teams as described earlier, you should get involved in the mailing lists (<https://lists.ubuntu.com>). Suggesting ideas is good, but as Linus Torvalds once said, "Talk is cheap. Show me the code." If you can package up a piece of software that's not yet in the Ubuntu repositories, and produce stable, clean .deb files, you'll be able to develop contacts and get yourself involved in the mailing lists. Similarly, if you have an idea for the base Ubuntu distro and can implement it in a clear, safe and easily-testable way, code it up and get posting on the lists..

Write, right?

If programming isn't your thing, but you'd like to try your hand at improving the distro's documentation, visit <http://doc.ubuntu.com>. Or if you've got more of an artistic and graphic design leaning, try contributing icons, logos and desktop backgrounds at <https://wiki.ubuntu.com/Artwork>. Whatever you choose to do, the Ubuntu joining-in page at www.ubuntu.com/community/participate provides links to all manner of resources – mailing lists, wiki pages and *Launchpad* sections. If you're friendly and describe your work or suggestions in detail, you'll get useful feedback and more stepping-stones to further development.

Or why not lend a hand with your own favourite distribution? See the box at the bottom of page 45 for more. Good luck! **LXF**

Creative Commons image: sfilaw, <http://tinyurl.com/3489hh>