So, you have mastered Linux and learned all its quirks and foibles? Whether you’re a developer or an end user and have Linux singing to your own tune, you may be wondering why we are suggesting you should break an unwritten rule of technology: “If it ain’t broke, don’t fix it.”

Well, Linux is not the only free operating system out in the open source wilderness, and it has some very worthy Unix competitors. The closest of these alternatives is a Unix system known as BSD, which has been with us since the 70s.

However, if you think that anything that’s been around since the 70s must be slow, bloated and frail in its old age then you couldn’t be more wrong. Originally distributed for free around the University of California within the Berkeley campus up until 1995, it has evolved into a highly secure and stable system that gives Linux (a relative newcomer by comparison) a run for its money. BSD has also been used on standard workstations where Unix licensing costs were prohibitively high.

The attention to detail within all the BSD development teams is absolutely meticulous. Hacks and workarounds are almost unheard of as developers tend to hold off for the most elegant solution. In short, BSD is the slimmer and more powerful ancestor that is still far from past its prime.

If all this has piqued your interest (or, like George Mallory, you just want to try it because it’s there!) then you will be glad to hear that not only will we be covering FreeBSD this issue but we will also be doing something similar with OpenBSD next month, with further tips on how to tailor your system to your own requirements.

Before we start however we must warn that these are hardcore tutorials, so we are assuming that you already have a high degree of familiarity with the Linux command line and are confident with using text-mode installers. We will therefore cover only the key stumbling blocks to look out for, with the initial installation and desktop setup and how to manage software packages rather than giving an in-depth step-by-step guide to these processes.

Wagons roll
The installer may take some by surprise. It’s a text-mode process, but you won’t be left typing Y or N the whole way through. You should see a multi-coloured text-mode menu that presents you with options. Much of the setup itself is self-explanatory, with most of the answers being ‘No’.

One part of the installation process that we suggest you take the time to configure is your network. It will be one of the first stages that appear, and we recommend you don’t worry about IPv6 for the time being and then simply set up the DHCP settings with the default options. Ignore the gateway setup and the options that follow within the network configuration menu, then proceed to the next stage. Another part would be your time zone. Simply select a continent, then...
the difference

your country. When BSD tries to set things up using BST rather than GMT this is completely optional, but ensure that the difference is correct first before proceeding.

Once you have moved past the initial configuration you are now ready to set up and configure packages. If you opted to download the CD ISO image you will only be able to install additional language packs, while those of you who downloaded the DVD will find additional packages here. This tutorial will assume that you are using the former and will therefore cover how to set up your system using network downloaded packages, but DVD users can read on to the packages they are likely to need, as this could speed up their setup (though you will still need to go through the same configuration steps later).

You can simply select ‘All’ here and it will install the packages they need. DVD users can select or deselect packages using the space bar in the same way that we will later when we configure the GUI.

Next, we recommend creating a standard user to accompany your root user account. If you leave the group space blank while creating your user, an associated group will be created automatically. After setting your root password it is safe to say ‘No’ to the following options. You will need to save and exit then restart your machine for changes to take effect.

The BSD philosophy

BSD and Linux are similar in many ways, but the philosophies behind both are very different. For instance. Linux is developed in a modular fashion, with everything apart from the kernel being an add-on to the core system. This means that you get bleeding-edge features and the system as a whole develops very quickly, but occasionally you’ll see messy hacks emerge to squash bugs or add a sexy new feature.

This is not necessarily a bad thing, as Linux leads the charge of the open source world and innovates quicker than almost any other large software project, but this can come as a sacrifice to stability as a change to one module could in theory have an adverse affect on another. This tends to be resolved by heavy maintenance from the kernel maintainers, distro makers and end users, but all it requires is one misplaced update or lack of communication between two component project teams and the system can become unstable or insecure. We saw this with the OpenSSL flaw with Debian systems listed here http://bit.ly/32hnok. This wasn’t spotted for nearly two years, but has since been fixed.

Keep it clean

BSD takes a very different approach. There is one base package, which contains the kernel and all the essential add-ons that the maintainers feel are absolutely essential to every system. All of these are maintained together at the same time, which means that you can be absolutely sure that the core of your system will ‘just work’ all of the time. This does mean that you lose a lot of scope to pull things apart and it can reduce the speed of innovation, but you get a huge pay-off in stability.

Also, BSD users know nothing of messy hacks. This is because of the BSD developers’ instance on clean code. By that they mean that you do not simply patch things as soon as possible without any thought to the overall security and stability to the system, and that you have not bypassed the most elegant solution to the issue. A Linux developer could place a patch as a temporary measure and never get around to patching it, but not in BSD. Again, this slows innovation and the rate at which patches arrive but the end result is a much smoother running system and a very elegant code base. This is also the root of why BSD tends to be much more secure and stable than other systems, because there is no room for compromise. To gain some perspective on this, where a Linux machine will typically give you a few months’ uptime in most cases, variants of BSD can happily give you a year or two. This may explain why the likes of NetBSD and FreeBSD are gaining quite a following in the web server markets.

Licensing

The Linux kernel and the software that runs on top of it tend to be licensed under the GNU General Public Licence (version 2 or 3). This means that you can share and redistribute the software (and even sell it) but if you make any changes you have to resubmit them to the author to improve future works. You also have to provide source code for any binaries you create.

The BSD licence has no such caveats. As long as you acknowledge the authors of the original code you can do whatever you like with it. This doesn’t encourage community code participation quite as much as the GPL, which is partly why BSD development is sometimes thought of as an elite band of kernel hackers. But it does mean from an end user point of view that you have more freedom at the expense of the rights the developers have given up.

The licences fit the cultures that surround the two kernels’ development, so it’s a personal decision after weighing up the pros and cons of each as to which you prefer and support.

Quick tip

This tutorial assumes you are using the CD install ISO for FreeBSD, but there’s also a DVD ISO containing additional packages from www.freebsd.org/where.html.

“Don’t say we didn’t warn you about text-mode installers!”

If you missed last issue Call 0870 837 4773 or +44 1858 438795.

www.tuxradar.com
Now that you have a hard-disk install complete, we can start to take a look at some quirks in the BSD command line when compared with Linux. For example, if you run the following command:

```bash
pwd
```

you will notice that your home folder is located in a very different place to the Linux standard directory structure. You can also see all the subfolders as you would on a Linux box. We recommend you navigate around the BSD directory structure to familiarise yourself, as this is one of the major differences between the two systems.

This system is ready to use but the next step is to set up CVSup (pronounced siv-sup), the tool we will need to keep the ports tree up to date (see the Green Fingers box, right, for an explanation of what it is and how it works). You will need to start by logging in as root and running `sysinstall` using the following two lines:

```bash
login root
sysinstall
```

You will need to supply your root password after the first command as you would with any other root terminal. The text-mode menu that appears will look very familiar indeed, and that would be because it is almost exactly the same as your CD installer. To install CVSup you will need to head to Configure > Packages > FTP Server, then pick the nearest server to your geographic location while ensuring you select Yes from the menu that follows.

It’s in the trees!

Now head to Net > cvs-without-gui, hit the Space bar and then select ‘OK’ by tabbing to it and pressing Enter twice (as you may have done with the DVD installer). Once completed, exit the installer and reboot to save your changes. Next, to install and populate the ports tree you need to run the following command in a root terminal session:

```bash
cvsup -h cvsup.freebsd.org /usr/share/examples/cvsup/ports-supfile
```

where `cvsup.freebsd.org` is the server you selected in the previous stage. The address I have quoted in this line is the central FreeBSD server, but if you can’t see the address in the text-mode installer then you will be able to find a list of available servers via `http://tinyurl.com/bsdmirrors`.

If all is going well at this stage you should see huge volumes of files being downloaded, and many of the package names you see will look very familiar indeed for most Linux users. Though most BSDs do have Linux binary emulation, a vast library of open source software has been ported natively to the ports tree. Reboot your machine to apply changes.

Automate and Update

As any Linux user will know, rerunning a terminal command every time you want to update the ports tree is not a very user-friendly thing to have to do. Thankfully we can automate things in BSD in a very similar way to Linux.

Those of you familiar with Vi can just go straight in and start editing files. For those of you who have not met Vi before you will be glad to know that it is easy enough to set Nano as the default text editor.

We’ll use the source method to install Nano, to show that you are not restricted to apps that have BSD packages. To install from source, use the following two commands:

```bash
cd /usr/ports/editors/nano
make install
```

You will notice that we don’t need to run the `configure` and `make` commands, because all the headers and the port-work are already done for you. Simply run `make install` and these stages (and even Autoconf) can and will be run for you.
a BSD install

as necessary. Ensure you hit ‘OK’ when the /etc/rc.conf window appears. Once again you will need to reboot to apply changes.

Now that Nano is installed you will need to set it as the default text editor before we can start scheduling things. Vi users can skip this part, but if you have elected to use Nano you will need to use the following commands:

```
cd /root
nano .cshrc
```

In this file you will need to scroll down until you find the following line:
```
setenv EDITOR vi
```
and then swap vi for nano. Press Ctrl+O to write your changes directly to disk (thus avoiding another reboot) then Ctrl+X to exit.

Now we need to edit the crontab entry using the following few lines:
```
cd /var/cron/tabs
ln -s /usr/share/examples/cvsup/ports-supfile /updatedir
```

This will shift you to the crontab folder and then log you in to edit the file as root. The reason for the symbolic link is because Nano secretly fakes word wrap by inserting a newline character. As this would appear within the command we are about to use (and crontab uses newlines to interpret commands) you could otherwise get all kinds of strange errors. If you are using Vi you should be unaffected, though the option is there should you need it.

Now that you’re editing the crontab file, add the following line to it:
```
15 3 * * * /usr/local/bin/cvsup -h cvsup.freebsd.org /updatedir
```

This will run an update at the same server you used before with root permissions at 3:15am every morning (if you need further direction on using crontab, subscribers will find that it’s covered in our shell scripting tutorial in LXF126). Save, exit and reboot to apply your changes.

Bells and whistles

Now you have a core system up and running, the next step (if you plan to use FreeBSD on a workstation) is to install a desktop. To install the X server run the following two lines.

```
pkg_add -r xorg
```

If using startx displays some strange-looking windows then all is fine with your installation. If you’re having issues with this step, subscribers will find some solutions to common pitfalls in the distro-making bonanza of LXF125.

Once you have rebooted the system it’s time to install a desktop. We will use Gnome with this tutorial and KDE in next month’s tutorial on OpenBSD to maintain our neutral stance on desktop choice. Use the following lines in the terminal:

```
pkg_add -r gnome
```

Then open /etc/rc.conf and add the following to the bottom of the file:

```
gnome_enable=YES
```

Save and exit as before, then run xorgconfig in the terminal. You can now finally launch GDM without running startx, and by using the line we used in /etc/rc.conf we ensure that the basic Gnome services like GDM, HAL, D-BUS and Avahi all run at boot time.

You should now have a functioning FreeBSD installation with a Gnome desktop. Look out for next issue’s hardcore tutorial when we’ll take things a step further using OpenBSD as our example BSD installation.

Green fingers

Almost any Linux distribution that springs to mind has its own package manager and at least one associated repository. In this respect FreeBSD is no different, but the way it does things may seem a little alien at first.

To start with, FreeBSD uses a CVS repository called the ‘ports tree’. The purpose of this is to save BSD users having to change header files around and port existing source packages to BSD, because it is already done for you.

In the setup tutorial you deal with a tool called CVSup, whose main purpose is to check out source packages. However, it is also used to upgrade existing package versions when you run a system update. This way you can keep up to date with the latest version of the software which has been ported by the BSD project.

It’s easy to install a precompiled binary package from the ports tree. Simply use the following command in a root session:

```
 pkg_add -r packagename
```

where packagename is the name of the package and version. If you are unsure of the version, just use grep as you would normally with the pkg_info command:

```
 pkg_info | grep package
```

where package is the full name of the package and version. If you are viewing any files or references to the package you should be unaffected, though the option is there should you need it.

Who doesn’t want screensavers installed for their BSD desktop?

Next month We’ll look at another cousin of Linux, OpenBSD.

www.tuxradar.com

October 2010 LXF136 | 99