Yet Another Gentoo GNU/Linux Installation Guide

Gary Bhumbra

November 9, 2017

Installing Gentoo GNU/Linux amd64 is not difficult, just tedious; it's therefore just like installing any other x86_64 operating system. However while Gentoo GNU/Linux comes with benefit of being one of the most customisable operating systems in existence, it comes at the cost of making its installation much more tedious. And depending on precisely what packages you install, Gentoo GNU/Linux is also both difficult and tedious to maintain. And be prepared for a colossal amount of computer-assisted time-wasting if you are on the path trying out Gentoo as your very first exposure to GNU/Linux.

Although installation and maintenance can be equally tedious, installation can sometimes be easier. Therefore a cowardly but occasionally most expeditive solution to updating a troublesome Gentoo GNU/Linux installation is reinstallation. Therefore make a note of what you do in case it makes the next time round slightly less tedious. You will thank yourself. If you haven't installed Gentoo before, you may look forward to the buzz of an exciting new learning experience. After you've done it a few times however the novelty quickly wears off and it just becomes a time-consuming ceremony laden with traps and plenty of easily forgettable steps that can be unforgivingly punishing if omitted or done incorrectly. As a result I've written this guide as an aide-mémoire for myself in the hope of not making too critical errors during installation.

This guide cannot replace the up-to-date information provided in the excellent Gentoo's Handbook but a confident and reliable full installation tends to require going through dozens of webpages with extensive cross-referencing. This is also tedious but unfortunately usually necessary especially when you run into problems. It is hoped this guide obviates this particular source of tedium at least in part by putting all the information in one place that was up to date in one moment in time. By necessity it makes a few choices for you (and not coincidently the ones I usually make) and does not explain every step in intricate detail. Be warned, Gentoo is changing all the time, and as a result this guide is revised practically every time I refer to it. So there will almost certainly be something in this guide which is already out of date by the time you finish reading this sentence.

1 Before installation

Software is replaceable whereas data is not. Therefore before considering installation, backup your data then backup your backups if you have not done so already. Use external media. If you prefer the cloud, that's your choice but I wouldn't make it myself. If you haven't installed Gentoo GNU/Linux before, set aside three hours for installing the GNU/Linux base system and another three for Xorg alongside a desktop environment. You might also benefit from having a few stiff drinks to hand, but pace yourself so you don't make any critical errors.

2 Live media

In order to write the live media, you need access to an internet-connected GNU/Linux distribution. This can also be done from Windows or (probably) MacOS machines but be sure you perform the checksums using whatever software is required. If you tempted to forego checksum confirmations, bear in mind you are about to invest a lot of time installing an operating system from scratch that could very easily go awry only because you decided to skip a step that takes a mere ten seconds and probably therefore not the smartest move.

Navigate to `http://distfiles.gentoo.org/releases/amd64/autobuilds/current-install-amd64-minimal` and download the latest install-amd-64.minimal-20??????.iso file and install-amd-64.minimal-20??????.iso.DIGESTS.asc. Use the sha512sum to confirm the integrity of the iso file:

```
# sha512sum install-amd-64.minimal-20??????.iso
# sed '5!d' install-amd-64.minimal-20??????.iso.DIGESTS.asc
```

Either burn the iso image to your preferred optical media (in this case /dev/sr0) using wodim (I use '\' here as a continuation character so omit it and just continue the line):

```
# wodim dev=/dev/sr0 blank=disc
# wodim dev=/dev/sr0 rdriveropts=burnfree speed=1 -dao fs=16m  \
        install-amd-64.minimal-20??????.iso
```

... or decompress to a flash memory media with a GNU/Linux file system (in this case /dev/sdi1), making sure the output device is correct or you may lose data.

```
# dd if=install-amd64-minimal-20??????.iso of=/dev/sdi1
```

Aspiring UEFI booters will save themselves a lot of work at this stage by downloading and burning the latest SystemRescueCd, which is based on Gentoo, in order to be able boot to console in UEFI mode. It is possible to create UEFI-bootable media within Gentoo but it really is not worth the effort. If you want to be able to boot Gentoo through UEFI, creating a SystemRescueCd disk at this stage can be a real time saver.

3 Live session and network initialisation

Note now may be a good time to update the UEFI/BIOS firmware on your motherboard. Indeed it might be essential if you want to install to recently developed solid state drive technologies. Just be sure you do so safely and if necessary use an uninterruptable power supply.

Use BBS or if you prefer, make the relevant changes in your UEFI/BIOS, to boot from the live media and your computer should boot via the ISOLINUX bootloader so you can load the Gentoo GNU/Linux live media with your preferred keymap configuration. Note that at the time of writing the Gentoo live install media cannot boot into UEFI mode but only in BIOS legacy mode. Upon booting to console, first check the system time is correct:

```
# date
```

If you have a DHCP connection, it is possible that the network interface card will be configured automatically with the correct TCP/IP settings. For manual TCP/IP configuration, you need to specify the network settings yourself (obviously your numbers will be different):

```
# ifconfig -a
# ifconfig enp7s0 192.168.1.1 netmask 255.255.255.0 up
# route add default gw 192.168.1.255
# vi /etc/resolv.conf
```

As an alternative to vi, you can use 'nano -w' to edit /etc/resolv.conf to list your DNS addresses such as:

```
nameserver 192.168.1.245
nameserver 192.168.1.254
```

Whether the network was configured automatically or manually, you should test it:

```
# ifconfig
# ping -c 3 www.gentoo.org
```

4 Partitioning and formatting

By this stage, it is necessary to have already chosen the following:

1. Which hard drive to install Gentoo onto.
2. Which partition arrangement you want.
3. Which file system to use for each partition.

This is entirely up to you. Don't do what exactly what follows because I'm just showing examples of partition arrangements none of which I use myself. The optimal choices depend on the hard drive configuration and the intended use of Gentoo GNU/Linux. An overview of your SATA drives can be obtained using the following:

```
# for devsd in $(find /dev/sd* ! -name "*[0-9]"); do $devsd \
  unit GiB print; done
```

Note this will not include NVMe drives. Once you have chosen the hard drive (say /dev/sdX), begin the partitioning using 'parted':

```
# parted -a optimal /dev/sdX
```

Here I write /dev/sdX rather than use a real example (e.g. /dev/sdg) because any lapse in concentration at this particular point may have dire consequences if you get this wrong. In any case, check the existing partitions by listing them using the 'print' command:

```
(parted) print
```

Partitions can be deleted by reference to their number (in the case 1):

```
(parted) rm 1
```

MBR or GPT labels can be established on partition free drives, which will of course delete all data on that drive:

```
(parted) mklabel msdos
or..
(parted) mklabel gpt
```

An MBR partition can be created and flagged for boot using the following commands, the first which creates a primary partition starting at the 1MiB position and finishing at the 1024MiB position of the hard drive:

```
(parted) mkpart primary 1MiB 1024MiB
(parted) set 1 boot on
```

For GPT partitioning with BIOS, a partition can be created, named, and flagged for boot using these commands:

```
(parted) mkpart primary 1MiB 1024MiB
(parted) set 1 boot_legacy on
(parted) set 1 bios_grub on
```

For GPT partitioning with UEFI, this first partition must be flagged as ESI with fat32 formatting:

```
(parted) mkpart ESI fat32 1MiB 1024MiB
(parted) set 1 boot on
```

For GPT partitioning, all partitions can be made primary. If many partitions are desired for MBR partitioning, it is very straightforward to do so by combining primary, extended, and logical partitions. In either case, it often convenient to default the size units beforehand:

```
(parted) unit MiB
(parted) mkpart primary 1024 98302
(parted) mkpart extended 98303 -1
(parted) mkpart logical 98304 131071
(parted) mkpart logical 131072 133119
(parted) mkpart logical 133120 163839
(parted) mkpart logical 163840 -1
(parted) unit GiB
(parted) print
(parted) quit
```

We can then use mkfs to format the partitions according to your preferred file system(s):

```
# mkfs.ext4 /dev/sdX1
# mkfs.xfs /dev/sdX2
# mkfs.ext4 /dev/sdX6
# mkfs.ext4 /dev/sdX7
# mkfs.btrfs /dev/sdX8
```

Note that the genkernel package does not work with reiser4 so avoid this file system. Finally, if you have an ESI partition, you must format it using fat32:

```
# mkfs.vfat -F 32 /dev/sdX1
```

5 Swap and mounting

In the above example, we are using /dev/sdX5 for swap, and can initialise and activate it by following commands:

```
# mkswap /dev/sdX5
# swapon /dev/sdX5
```

The first partition to mount is the intended / partition.

```
# mount /dev/sdX2 /mnt/gentoo/
```

Before mounting the remaining partitions, we must create the corresponding subdirectories, and after mounting in the the case of /tmp and /var/ we change the permissions setting.

```
# mkdir /mnt/gentoo/boot
# mkdir /mnt/gentoo/tmp
# mkdir /mnt/gentoo/var
# mkdir /mnt/gentoo/home
# mount /dev/sdX1 /mnt/gentoo/boot
# chmod 1777 /mnt/gentoo/tmp
# mount /dev/sdX6 /mnt/gentoo/tmp
# chmod 1777 /mnt/gentoo/var
# mount /dev/sdX7 /mnt/gentoo/var
# mount /dev/sdX8 /mnt/gentoo/home
# ls -l /mnt/gentoo/
```

Having established the mount points for the partitions, we are in a position to download the installation package. Notice we have not yet chrooted.

6 Stage 3 tarball downloading and unpacking

The stage 3 tarball is simply Gentoo GNU compiled without GRUB. So unlike Gentoo GNU/Linux, the stage 3 tarball has neither kernel nor bootloader. The latest Gentoo stage 3 tarball can be downloaded to /mnt/gentoo/ using a console browser:

```
# cd /mnt/gentoo
# links http://www.gentoo.org/main/en/mirrors.xml
```

Press the 'Page Down' and 'Down' cursor key until you find your preferred ftp mirror. Hit Return, then navigate to /releases/amd64/autobuilds/current-stage3-amd64/ and highlight then download, by pressing 'd' and Return, the stage3-amd64-20??????.tar.bz2 and stage3-amd64-20??????.tar.bz2.DIGESTS.asc. After downloading both files press 'q' to exit links. Confirm the sha512 sum is correct:

```
# sha512sum stage3-amd64-20??????.tar.bz2
# sed '5!d' stage3-amd64-20@?????.tar.bz2.DIGESTS.asc
```

We unpack the installer with tar:

```
# tar fxjp stage3-amd64-20??????.tar.bz2
# ls
```

Notice the standard GNU directories have been unpacked. The tarball files can now be archived in the root home directory.

```
# mkdir root/stage3
# mv stage-* root/stage3/
# ls
```

You can decide at some point in the future to delete the stage3 tarball and directory or keep them for posterity.

7 Flags and chrooting

Before updating the software package database and installing the Linux kernel within the unpacked tarball, it is a good idea to edit the configuration file /etc/portage/make.conf to optimise the installation for the hardware. Edits can be made using vi or nano:

```
# vi etc/portage/make.conf
```

The CFLAGS, USE, and MAKEOPTS fields can be edited to optimise each compile. An example might be:

```
CFLAGS="-O2 -march=native -ftree-vectorize -pipe"
USE="bindist openmp truetype"
CPU_FLAGS_X86="mmx mmxext sse sse2 ssse3 sse4_1 sse4_2 avx avx2"
MAKEOPTS="-j8"
```

While it may be tempting to change CFLAGS to "... -O3 ...", don't. For the base system, reliability is preferred over performance. If you want optimal flags for specific packages, this can always be set at a later stage. The MAKEOPTS variable refers to the number of parallel sessions for parallel jobs. The recommended number (which in the illustrated example is 8) corresponds to the total number of logical processors of the CPU(s).

The USE flags denote package library inclusions in way that will look opaque to Gentoo newcomers at this stage. For now, "bindist openmp truetype" is sufficient although over time, your USE flags might expand to something more like this:

```
USE="bindist openmp truetype dbus policykit udisks elogind
     -systemd -consolekit cups ppds imap smtp sasl python"
```

... but this can wait for now. More immediately, we have to configure the system to point to locations to download sources and synchronise the package repository, that contains the ebuilds and related files used to compile the sources. The source mirror can be selected using the handy 'mirrorselect' utility, remembering to use the Spacebar to select the desired mirror.

```
# mirrorselect -i -o >> etc/portage/make.conf
```

The 'GENTOO_MIRRORS' setting can be confirmed thus:

```
# cat etc/portage/make.conf | grep MIRRORS
```

The package repository synchronisation location is specified by a file which presently does not exist located in directory which also does not exist. Create the directory and then create the file by copying from a template:

```
# mkdir -p etc/portage/repos.conf
# cp usr/share/portage/config/repos.conf \
     etc/portage/repos.conf/gentoo.conf
```

It should not be necessary to modify the synchronisation location from the default, but it can be viewed from the file.

```
# cat etc/portage/repos.conf/gentoo.conf | grep sync
```

Before chrooting, there are a couple housekeeping jobs we have to perform. The first is making the DNS name server settings in our present environment available to our chrooted environment:

```
# cp -L /etc/resolv.conf etc/
```

The second is, as ever before any chroot, mounting proc, sys, and dev:

```
# mount -t proc /proc /mnt/gentoo/proc
# mount --rbind /sys /mnt/gentoo/sys
# mount --rbind /dev /mnt/gentoo/dev
```

After chrooting, we update the environment variables and load the source settings from /etc/profile to memory, and tweak the prompt to remind us we are in a chrooted shell:

```
# chroot /mnt/gentoo /bin/bash
# env-update && source /etc/profile
# export PS1="(chroot) $PS1"
```

If you are able to chroot, you can either congratulate yourself or feel relieved that you are rooted into your new Gentoo system. Otherwise, you must have made an error and must backtrack. But even if you can chroot, don't get carried away because we are still a long away from a functioning system with neither a configured package manager nor kernel nor bootloader.

8 Configuring Portage and regionalisation

The first step is to retrieve a Portage snapshot of available software, update the Portage tree, and update the Portage installation. Welcome to 'emerge', the spell that invokes Portage's sourcery (the 'q' option stops Portage from spouting all over the slow console framebuffer, 'a' asks for confirmation, and 'u' updates).

```
# emerge-webrsync
# emerge -q --sync
# emerge -qau portage
```

During the snapshot retrieve, emerge-webrsync may complain of a nonexistent directory '/usr/portage'; this is expected at this stage and no cause for alarm. The latest Gentoo news can be listed or read using eselect:

```
# eselect news list
# eselect news read
```

Read the news and if there is anything that conflicts with what follows in this guide, do what the news article(s) tells you. The default settings for the package manager is specified by the choice of profile. Available profiles can listed and set using ‘eselect’:

```
# eselect profile list
# eselect profile set 1
```

Optionally, configure the timezone setting. The available timezones can be listed easily.

```
# ls /usr/share/zoneinfo
```

The preferred timezone can be selected (in this case UTC) using the echo command.

```
# echo "UTC" > /etc/timezone
```

If the timezone is set, it must be invoked using through the package manager.

```
# emerge --config sys-libs/timezone-data
```

Before selecting a regional setting, it must be configured by editing locale-gen. Since after chrooting vi is no longer available, you must use nano:

```
# nano -w /etc/locale.gen
```

Here, the regional ISO and UTF-8 locale settings can be uncommented or entered as required; at least one UTF-8 locale is recommended. Then the regionalisation settings can be generated:

```
# locale-gen
```

Finally the regionalisation setting can be selected and applied to present environment:

```
# eselect locale list
# eselect locale set 3
# env-update && source /etc/profile
# export PS1="(chroot) $PS1"
```

At this stage we have updated and configured the package manager, and therefore now in a position to configure and compile the kernel.

9 The Linux kernel sources and compiling

At this point is essential to know the difference between GNU and Linux. GNU is almost the entire operating system. It includes essential elements such as the shell (i.e. Bash) and the compiler collection (i.e. gcc), as well as important code libraries (e.g. glibc), tools (e.g. libtool), and utilities (e.g. binutils). It does not (normally) include any kernel, which is the part of the operating system that interfaces with hardware. This function is (normally) performed by the Linux kernel. Linux is therefore not the operating system, despite casual references to the contrary being the norm, but only the kernel of the operating system. In Gentoo the distinction between GNU and Linux is very important and that is why this guide refers to the full operating system as Gentoo GNU/Linux.

The Linux kernel sources, kernel generator, and USB utilities application should first be installed:

```
# emerge -qan gentoo-sources genkernel usbutils
```

Notice how these packages are added to the 'world' set. The world set is a list of packages you want to install and have updated on a regular basis including their dependencies. This set should include the Linux kernel contained within the gentoo-sources package.

Before compiling the kernel, features can be flagged to be omitted, flagged to be compiled intrinsically, or flagged to be compiled as modules. The ceremony of tweaking those features is one you are likely to repeat many times if you end up using Gentoo extensively. While in theory it is possible to make tweaks by editing a text file (/usr/src/linux/.config), it's only really practical to make changes using Gentoo's menu-based kernel configuration program:

```
# cd /usr/src/linux
# make menuconfig
```

In a strange sence, it almost doesn't matter what you do at this point because it is almost inevitable that you will be revisiting the kernel configuration program many times to make changes. There are however a few things that might be worth changing at this point. Most systems have multiple cores, whose support have be enabled for all of them to be used.

```
Processor type and features -->
[*] Symmetric multi-processor support
```

If you intend to use the Nvidia proprietary driver, consider enabling full MTRR and PAT support for 2D graphics acceleration:

```
Processor type and features -->
[*] MTRR (Memory Type Range Register) support
[*]   MTRR cleanup support
(1)     MTRR cleanup enable value (0-1)
(1)     MTRR cleanup spare reg num (0-7)
[*]   x86 PAT support
```

For UEFI boots, enable support for EFI runtime.

```
Processor type and features -->
[*] EFI runtime services support
[*]   EFI stub support
[*]     EFI mixed-mode support
```

At this point you need to change to a different submenu, 'File systems', where you can add the EFI filesystem support.

```
File systems -->
[*] Pseudo Filesystems -->
[*]   EFI Variable filesystem
[*] Miscellaneous Filesystems -->
[*]   EFI file system support (read only)
```

If you dual-booting with Windows or require access to NTFS partitions, then you should consider:

```
File systems -->
[*] FUSE (Filesystem in Usersupace) support
<*> Overlay filesystem support -->
-->DOS/FAT/NT Filesystems -->
<*>   NTFS file system support -->
[*]  NTFS write support
```

If you are using optical media, you might consider:

```
File systems -->
<*> Overlay filesystem support -->
--> CD-ROM/DVD Filesystems -->
<*>   UDF file system support
```

For the tmpfs pseudo file system, it needs to be enabled.

```
File systems -->
--> Pseudo Filesystems -->
-*-Tmpfs virtual memory system file support
```

For CUPS/SAMBA/CIFS services, you should activate the following kernel options:

```
File systems -->
[*] Network File Systems -->
<*>   CIFS system support (advanced network filesystem, SMBFS successor)-->
[*]     CIFS extended attributes
[*]     DFS feature support
[*]     SMB2 and SMB3 network file system sypport
```

Under ‘File systems’, you will also find support for XFS and Btrfs.

```
File systems -->
<*> XFS filesystem support -->
<*> Btrfs filesystem support -->
```

Under Device Drivers (why the second ‘D’ is capital I don’t know), you want to check you have support for the ethernet card.

```
Device Drivers -->
[*] Network device support -->
[*]   Ethernet driver support -->
[*]     Your network interface device
```

... so you can be sure that your network interface device is included. If you are not sure which one to install, you might benefit from opening another virtual console (Ctrl-Alt-F2) and running:

```
# lspci | grep -i ethernet
```

Then you can return to the configuration program (Ctrl-Alt-F1) to include the driver in the kernel compile.

If you intend to install a VPN client, then you need to enable TUN/TAP device driver support in the kernel.

```
Device Drivers -->
[*] Network device support -->
[*]   Network core driver support -->
<*>     Universal TUN/TAP device drivers support
```

If you intend to install Xorg, then you need to include input device event interface support:

```
Device Drivers -->
--> Input device support -->
<*>   Event interface
```

Resolution changes in virtual consoles are facilitated, though by no means guaranteed, by enabling support for frame buffer devices (and for EFI-based support if required):

```
Device Drivers -->
--> Graphics support -->
-->    Support for frame buffer Devices --->
[*]      Enable Video Mode Handling Helpers
<*>      Userspace VESA VGA graphics support
[*]      VESA VGA graphics support
[*]      EFI-based Framebuffer Support
```

Xorg also requires KMS framebuffer support for consoles:

```
Device Drivers -->
--> Graphics support -->
-->  Console display driver support --->
<*>    Framebuffer Console Support
```

If you intend to install the open source drivers for Intel, Nvidia, or AMD, you must include DRI support within the kernel. On the other hand if you intend to install proprietary drivers, make sure these are disabled.

```
Device Drivers -->
--> Graphics support -->
<*>Direct Rendering Manager --->
<*> ATI Radeon
<*> Nouveau (NVIDIA) cards.
<*> Intel Graphics.
```

Sound card support is also specifiable under Device Drivers.

```
Device Drivers -->
<*> Sound card support -->
<*>   Advanced Linux Sound Architecture --->
[*]     PCI sound devices ->
<*>       Your sound card
```

Once again, another virtual console may be helpful in identifying the sound card.

```
# lspci | grep -i audio
```

Multimedia support for audio/video USB devices (such as webcams) is enabled within the following:

```
Device Drivers -->
<*> Multimedia support -->
[*]   Cameras/video grabber support
[*]   Media USB Adapters -->
<*>     USB Video Class (UVC)
[*]       UCV input events device support
[*]     Your USB webcam (if listed)
```

Identification of these devices is facilitated by lsusb installed from the usbutils package:

```
# lsusb
```

Finally, if you intend to use KDE Plasma with elogind, then other kernel options are recommended:

```
General setup  --->
[*] Control Group support  --->
[*] Configure standard kernel features (expert users)  --->
[*]   Enable eventpoll support
[*]   Enable signalfd() system call
[*]   Enable timerfd() system call
File systems  --->
[*] Inotify support for userspace
Security options  --->
[*] Simplified Mandatory Access Control Kernel Support
```

Once you've saved your configuration (using the left/right cursor keys and selecting 'Save') and exited the program, you can backup for configuration file, for example:

```
# cp -i .config .config.copy
```

The main part of the ceremony is compiling the kernel, its modules, and installing the kernel image into the boot partition:

```
# make && make modules_install
# make install
```

Depending on the speed of your computer and hard drive, this may take some time. You might want to have a book to read in the meantime. Very rarely there are errors, but when there are there is generally a good description of the what the problem is which make should make it straightforward to correct. After the first kernel compile, you should install initramfs support if it is needed, such as if you have decided on separate partitions for /var/ or /tmp/:

```
# genkernel --install initramfs
```

The presence of the initramfs file can be confirmed on the boot partition:

```
# ls /boot/initramfs*
```

And so far we have downloaded the stage 3 tarball to unpack Gentoo GNU, configured Portage, compiled and installed the Linux kernel, but we remain without a fully operational GNU/Linux system which by the way still does not have a bootloader. At this stage it is normal to feel impatient and it can be highly therapeutic to take a break from the computer even if briefly. The next step is to establish the default partition mounting and set up the definitive network.

10 Initialising /etc/fstab and the network

Install parted and if you use ntfs, btrfs, xfs, consider installing their associated programs:

```
# emerge -qan parted ntfs3g btrfs-progs xfsprogs
```

Note for UEFI boots, you will need to create (if you have not already so) an efi directory in /boot:

```
# mkdir /boot/efi
```

Default partition mounts are mandatory in /etc/fstab entries:

```
# nano -w /etc/fstab
```

While classically, hard drive partitions are referred by their device identity directly (as we do here), it is best practice for reliability to refer them by their corresponding (UUID=???) in fstab which can be listed using (replacing sdX with you chosen drive):

```
# blkid /dev/sdX
```

Use of an additional virtual console and gpm (console mouse) can be very helpful here. Note for UEFI boots, you will need to include a label-based entry:

```
/dev/sdX1 /boot/efi vfat defaults,noatime 0 2
```

The default template is helpful to set up the correct fstab entries. A second tty terminal can provide helpful reminders at this point:

```
# df -h
```

You might end up with something that slightly resembles this:

```
UUID=??? swap swap defaults 0 0
UUID=??? / xfs defaults 1 1
UUID=??? /boot ext4 defaults,noatime 1 2
UUID=??? /boot/efi vfat default,noatime 0 2
UUID=??? /tmp ext4 defaults 1 1
UUID=??? /var ext4 defaults 1 1
UUID=??? /home btrfs defaults 0 0
```

Now it's probably not the best time to test your /etc/fstab (e.g. using 'mount -a') since you are still chrooted with a number of essential things to do before rebooting. The first is setting up the definitive network. Whether networking via DHCP or manually, the hostname and domain is set by editing their corresponding configuration files:

```
# nano -w /etc/conf.d/hostname
-> hostname="tux"
# nano -w /etc/conf.d/net
-> dns_domain_lo="dns_domain"
```

Configuring DHCP-based connections is very straightforward and requires only installing the DHCP client:

```
# emerge -qan dhcpcd
```

Otherwise, the manually-set network is most easily configured by first installing netifrc:

```
# emerge -qan netifrc
```

For manual configuartion, enter the domain name and IP addresses by editing the file /etc/conf.d/net to set the variables corresponding to the ethernet device (here we use eth0).

```
dns_domain_lo="localdomain.com"
config_eth0="192.168.1.128 netmask 255.255.255.0"
routes_eth0="default via 192.168.1.254"
```

In order to activate the manually-configured network on boot, it must be added to the default runlevel configuration:

```
# ifconfig
# cd /etc/init.d/
# ln -s net.lo net.eth0
# rc-update add net.eth0 default
```

The host configuration should be set by editing the /etc/hosts file to specify the host and domain names:

```
# nano -w /etc/hosts
--> 127.0.0.1 localhost.localdomain.com localhost
```

If you ever change the contents of /etc/conf.d/net (e.g. to change eth0 to enp7s0), you can remove the previous configuration by deleting the previous symbolic link and its runlevel inclusion:

```
# cd /etc/init.d/
# ln -s net.lo net.enp7s0
# rm net.eth0
# rc-update add net.enp7s0 default
# rc-update del net.eth0 default
```

Now the mount points have been configured and definitive network has been established, the root account must be set and basic services initialised.

11 Root account and basic services

This section is even more tedious than the last so we'll be brief here since it doesn't have to take a long time. Set the root password:

```
# passwd
```

If you want to configure OpenRC-controlled services yourself, this can be done by editing the following:

```
# nano -w /etc/rc.conf
```

Set the key map and review the clock options:

```
# nano -w /etc/conf.d/keymaps
# nano -w /etc/conf.d/hwclock
```

Now we are in a position to install and initialise some basic services. Install the cron and file index services, and add cronie to the runlevel default:

```
# emerge -qan cronie mlocate
# rc-update add cronie default
```

For UTC hardware clock time syncronisation with NTP, install the ntp-client, start it and add it to the default runlevel.

```
# emerge -qan net-misc/ntp
# /etc/conf.d/ntp-client start
# rc-update add ntp-client default
```

In order set the regionalised time-zone setting, backup /etc/localtime and overwrite it with a symbolic link to your desired location (here I use the London time zone, but all available zones are listed in /usr/share/zoneinfo/).

```
# cp -i /etc/localtime /etc/localtime.default
# ln -sf /usr/share/zoneinfo/Europe/London /etc/localtime
# date
```

If you wish sshd to be available by default, start it and add it to the runlevel configuration:

```
# rc-update add sshd default
```

With sshd started, it is a good idea to install and start a system logger (alongside a log rotation daemon), then adding it to the runlevel configuration.

```
# emerge -qan syslog-ng logrotate
# /etc/conf.d/syslog-ng start
# rc-update add syslog-ng default
```

Finally we are now in the position of installing the bootloader and rebooting.

12 Initialising bootloader and rebooting

If you intend to use GRUB, ensure you have included 'truetype' fonts under the use flag in the /etc/portage/make.conf file:

```
USE="... truetype ..."
```

Installing GRUB for BIOS legacy boots is very straightford. Install GRUB and the OS prober, set the bootloader onto the correct hard drive, then 'make' the GRUB configuration file, exit the chroot, then reboot:

```
# emerge -qan sys-boot/grub sys-boot/os-prober
# grub-install /dev/sdX
# grub-mkconfig -o /boot/grub/grub.cfg
# exit
# reboot
```

Remember that the grub-mkconfig line must repeated after every time you 'make install' the kernel.

For UEFI boots, GRUB installation is more long-winded. It is a chicken-and-egg situation where you have to boot into UEFI mode before you can install the bootloader. Before installing GRUB, the make.conf file needs to be modified to specify a GRUB-EFI installation.

```
# echo ‘GRUB_PLATFORMS="efi-64"‘ >> /etc/portage/make.conf
# emerge -qan sys=boot/grub:2 os-prober
# mkdir /boot/grub
# mount /dev/sdX1 /boot/efi
# grub-install --target=x86_64-efi --efi-directory=/boot/efi \
  --boot-directory=/boot/efi/EFI --removable --no-floppy --recheck
# grub-mkconfig -o /boot/grub/grub.cfg
# cp -i /boot/grub/grub.cfg /boot/efi/EFI/grub/
# exit
# reboot
```

If your computer booted from the Gentoo installation media, it is highly probably that GRUB will fail to install properly because of booting in BIOS legacy mode. For this reason, it is useful to have already a chicken (or egg) to hand in the form of the SystemRescueCd media. After booting SystemRescueCd to console in UEFI mode, it will be necessary to undergo the familiar chroot ceremony and remounting before installing the GRUB-EFI bootloader:

```
# mount /dev/sdX3 /mnt/gentoo
# mount -t proc /proc /mnt/gentoo/proc
# mount --rbind /sys /mnt/gentoo/sys
# mount --rbind /dev /mnt/gentoo/dev
# chroot /mnt/gentoo/ /bin/bash
# env-update && source /etc/profile
# mount -a
# grub-install --target=x86_64-efi --efi-directory=/boot/efi \
  --boot-directory=/boot/efi/EFI --removable --no-floppy --recheck
# grub-mkconfig -o /boot/grub/grub.cfg
# cp -i /boot/grub/grub.cfg /boot/efi/EFI/grub/
# exit
# reboot
```

When rebooting, select UEFI boot mode into GRUB. If you boot into a GRUB's minimal BASH, the most likely reason is that GRUB's configuration file cannot be found. The expected location of the configuration file can be viewed using the 'set' function:

```
(grub) set
```

If you look within 'prefix', you will see the expected location. If the location is not in /boot/efi/EFI, you will need to copy the configuration file into the correct directory. If needed, boot again using the SystemRescueCd, make any necessary mounts as before, chroot, then copy the configuration file to the target location:

```
# cp -i /boot/efi/EFI/grub/grub.cfg /boot/efi/grub/
# reboot
```

Now you should now be able to boot into Gentoo. Remember whenever you 'make install' the kernel, you have to rerun GRUB's mkconfig, and if necessary copy the image to the target location. If now you are able to boot, you can either congratulate yourself or feel relieved that can boot into your Gentoo system. Log-in as root, and then create the first non-root user account:

```
# useradd -m -G users,wheel,audio,video -s /bin/bash user
# passwd user
```

Now check you have a functioning network device.

```
# ifconfig
# ping -c 3 www.gentoo.org
```

If not, either your network settings are wrong or you need to recompile the kernel with the correct driver for the network interface card (the latter case evident from the result of ifconfig). So long as there is a functioning network connection, non-nano users would be delighted to install their preferred console-based editor:

```
# emerge -qan vim
```

If you encounter required USE changes, then when you are invited to make modifications. Note that if you answer 'Y', Portage rather misleadingly won't make the changes. Instead it will create ._cfg* files inside /etc/portage/ that will list the necessary changes. You need to decide which config changes are made. A convenient package for listing, comparing, and implementing the config changes is etc-update:

```
# etc-update
```

Once you fulfil the required USE changes, you should then be able to install or update the relevant packages.

At this stage, the direction to take with your installation will depend on how you want Gentoo to work for you. What follows is a guide on how to install an OpenRC-based Xorg-server run KDE-Plasma desktop. If you are looking for something different, such as using systemD with Wayland running a GNOME desktop, then you have to look elsewhere but remember there are plenty of online resources to help.

13 Installing interface services and finalising /etc/fstab

With the inevitable USE changes you are likely to require, you might find preferable to use gpm rather than etc-update to make the updates. The gpm package installs a service that can needs to be started and added to the default runlevel.

```
# emerge -qan gpm
# /etc/init.d/gpm start
# rc-update add gpm default
```

It is perfectly possible that beyond this you have no desire to interface with foreign filing systems or devices (including printers) and can therefore merrily skip the remainder of this section. Otherwise, it may be convenient for you to install a useful quartet of packages that facilitate communication using protocols not native to GNU/Linux, namely NTFS, CIFS, SAMBA, and CUPS, even if you don't intend to use all of them for the moment. Note that the samba and cups services need to started and added to the runlevel after defaulting the smb configuration file.

```
# emerge -qan ntfs3g cifs-utils net-fs/samba cups
# cp -i /etc/samba/smb.conf.default /etc/samba/smb.conf
# /etc/init.d/samba start
# rc-update add samba default
# /etc/init.d/cupsd start
# rc-update add cupsd default
```

The use of CUPS (navigating to 127.0.0.1:631 on a web-browser) can make network printer installation very straightforward, for example via an Appsocket (port 9100) TCP/IP address (although it may require having to download a PPD file or 'emerge -qan hplip' for Hewlett-Packard printers). If you wish to access NTFS partitions, you can use for your preferred console text editor to modify /etc/fstab in order to mount NTFS partitions at startup, according to your locale-settings such as (here I use locale=en_GB.UTF-8):

```
/dev/sdb1 /mnt/ntfs/d ntfs-3g locale=en_GB.UTF-8 0 0
```

Again it is best practice for reliability to refer to the UUID identities of the relevant partitions which can be listed using:

```
# blkid
```

At this point, you can create the required mount point directories and test the fstab entries mount correctly.

```
# mkdir -p /mnt/ntfs/d
# mount -a
# ls /mnt/ntfs/d
```

If you are mounting network shares, they should be included in the /etc/fstab e.g.:

```
//192.168.1.128/foo /mnt/share cifs \
  credentials=/root/.sharecreds,uid=0,_netdev,file_mode=0777,\
  dir_mode=0777 0 0
```

In this case, we need to create /root/.sharecreds as a text file comprising of (hopefully not to the letter)...

```
username=foo
password=bar
```

... and we also need to create the share directory before mounting.

```
# mkdir /mnt/share
# mount -a
# ls /mnt/share
```

When you next reboot, check the network shares are properly mounted.

Now we are ready to install Xorg and the graphics driver.

14 Xorg and graphics driver

The two are installed together. It is important that you have configured the kernel to be compatible with Xorg (as described above), and if desired to include the open source graphics drivers. The /etc/portage/make.conf file must be expanded to include libinput and specify your graphics drivers:

```
INPUT_DEVICES="libinput"
VIDEO_CARDS="intel"
```

You should include ”synaptics” with the ”INPUT_DEVICES” if you intend to use a touchpad (separated from ‘libinput’ by only a space). The VIDEO_CARDS specifies the driver whether open-source ”radeon” or ”nouveau” or their propriety counterparts ”fglrx” (which Xorg no longer supports) or ”nvidia”. Note that modern AMD graphic cards should use the open-source driver ”amdgpu” in the state Gentoo's equivalent proprietary driver seems to be in a state of flux.

Before installing Xorg, it is very good idea to emerge a verbose ‘pretend’ run to look for any USE changes. At this point, you will need to almost certainly need to modify the /etc/portage/package.use/iputils file to include whichever changes are required. This can done using ‘echo >>’, an editor (with gpm), or ‘etc-update’. The following line will pretend-install the packages necessary for Xorg but will most likely fail:

```
# emerge -pv xorg-server
```

Use gpm or etc-update to make the necessary package use changes. Finally install Xorg and graphics driver, then reinitialise the environment variables and source path:

```
# emerge -qan xorg-server
# env-update && source /etc/profile
```

At this stage, you might want to test the Xorg is installed and configured correctly. Purely for this purpose it is a good idea transiently to install an X Terminal and the Tab Window Manager.

```
# emerge -qan x11-terms/xterm x11-wm/twm
```

If changes were made to the kernel, you need reboot into it before testing Xorg. Also, if you installed the Nvidia propietary driver, it needs to be initialised then rebooted before it can be tested:

```
# nvidia-xconfig
# reboot
```

The Xorg server can be started and tested from console.

```
# startx
```

If all is well, you should see a Xorg screen with some X terminals. By the way, if this is the first time you have seen Xorg working on Gentoo then wipe that grin of your face because we still have no proper desktop environment. Once satisfied, return to the virtual console tty and uninstall the test packages and their dependences.

```
# emerge --unmerge xterm twm
# emerge -a --depclean
```

Note the last line might uninstall nano, but this can be rescued now or later by reinstalling it.

```
# emerge -qan nano
```

15 Installing Plasma and finalising bootloader

First we select a profile for KDE Plasma. Available profiles can be listed:

```
# eselect profile list
```

And, for example, if we select the 6th profile for Plasma, we set it:

```
# eselect profile set 6
```

Before installing Plasma, a few services need to be installed, activated, and added to the default runlevel. Actually there are several combinations you can use, but the following represents a Gentoo-friendly OpenRC-compatible combination:

eudev : this service is the manager for various devices interfacing as files with the Linux kernel.

dbus : this service provides an interprocess communication system for software applications.

polkit : this service provides an authorisation interface for privileged services, such as daemons, to provide services for unprivileged services or processes.

udisks : this service is a dbus daemon offering a range of services to storage devices.

elogind : this services facilitates multiple-user services.

The eudev package is installed as a one-shot procedure so that it is not included in the world set before restarting it (as udev not eudev):

```
# emerge -qa --oneshot sys-fs/eudev
# /etc/init.d/udev --nodeps restart
```

Establish udev and udev-trigger are added in the sysinit runlevel:

```
# rc-update add udev sysinit
# rc-update add udev-trigger sysinit
```

The remaining packages are included within the USE list of /etc/portage/make.conf USE list, and while there, set the L10N regionalisation for Plasma (here I use 'en_GB' from the list provided in /usr/portage/profiles/desc/l10n.desc):

```
USE="... dbus polykit udisks elogind -systemd -consolekit ..."
L10N="en_GB"
```

Note to avoid conflicts, the recommended exclusions of systemd and consolekit. Invoke the use changes using emerge:

```
# emerge -DaquN @world
```

As a warning, this might might take some time and an unreliable connection to your chosen download mirror may necessitate repeating this command. After the services has been installed, start and add the first dbus then elogind to the initialisation configuration as necessary.

```
# /etc/init.d/dbus start
# rc-update add dbus default
# /etc/init.d/elogind start
# rc-update add elogind boot
```

At the time of writing, elogind was in a state flux and not marked as stable. Therefore stable users need it unmasked.

```
# mkdir /etc/portage.profile/
# echo "-elogind" >> /etc/portage/profile/use.stable.mask
```

Certain Plasma packages may also require including elogind as accepted package keyword to install. You need to check the Gentoo website to check the latest requirements. For example, something resembling the following might be required inside /etc/portage/package.accept_keywords which might have to created:

```
sys-auth/elogind
>=net-misc/networkmanager-1.8.0
```

Before installing Plasma, a few legacy support package options would optimally be implemented here. Legacy system tray compatibility can be supported with the following option:

```
# echo "kde-plasma/plasma-desktop legacy-systray gtk2 gtk3 qt4" \
  >> /etc/portage/package.use/plasma-systray
```

And support for 32-bit binaries, such as Skype, running from the system tray require another use specification:

```
# echo "dev-libs/sni-qt abi_x86_32" \
  >> /etc/portage/package.use/plasma-systray
```

Check for, and action any further package use requirements for Plasma, before installing the full Plasma suite.

```
# emerge -pv kde-plasma/plasma-meta
# emerge -qan kde-plasma/plasma-meta
```

This may take some time so you might want to have book ready. Something however might go wrong; if this is the case, the good news is that it is almost certainly your own fault and you have to back-track. If you make changes to your USE flags, then you can invoke them generally:

```
# emerge -DaquN --with-bdeps=y @world
```

There is always however a very remote chance it is not your fault. In this case you have little choice but to seek help at the Gentoo forum and/or submit a bug report and/or persevere. Otherwise if this goes through to the end without any hitch, you might want to remove redundant packages that you might have installed inadvertantly.

```
# emerge -a --depclean
```

Edit /etc/conf.d/xdm to change the entry under the 'DISPLAYMANAGER" variable to 'sddm':

```
DISPLAYMANAGER="sddm"
```

In order for the default runlevel to start Plasma automatically, add xdm to the configuration.

```
# rc-update add xdm default
```

Then reboot to log into a Plasma session as your non-root user. Almost certainly, you will want to change the default desktop wallpaper to something less garish; this is trivially done from the context menu right-clicking the desktop. Press Alt-F2 to enter System settings. Under 'Input Devices' click 'Hardware' and under the 'Layouts tab', click the 'Configure layouts' to choose your preferred keyboard layout, then finally click 'Apply'.

A number of Plasma's packages you might expect to be present may well not be installed. For example. why the plasma-meta installs VLC but neither Konsole nor Dolphin is entirely a mystery but there we go. They can be installed by logging in as root from a virtual terminal and using emerge.

```
# emerge -qan konsole kde-apps/dolphin
```

With a working desktop, it is now with some confidence you can finalise the bootloader in case of console blind booting. We will begin with setting the bootloader to the default ‘starfield’ theme, first copying the share GRUB directory contents to the boot grub directory, and backup our configuration.

```
# cp -r /usr/share/grub/* /boot/grub/
# cp /etc/default/grub /etc/default/grub.default
```

Now edit /etc/default/grub with the following changes (adjusting the GRUB_GFXPAYLOAD to your preferred resolution):

```
GRUB_TERMINAL=gfxterm
GRUB_GFXMODE=auto
GRUB_GFXPAYLOAD_LINUX=1920X1080X16
GRUB_THEME="/boot/grub/themes/starfield/theme.txt"
GRUB_BACKGROUND="/boot/grub/themes/starfield/starfield.png"
GRUB_FONT="/boot/grub/unicode.pf2"
```

The GRUB bootloader resolution however is not guaranteed, especially if you have to rely on a EFI GOP driver. If this cannot be fixed, you can at least set the virtual console resolution manually in the kernel parameters, so long as you have set the kernel configuration to support console framebuffer drivers correctly (see above). The kernel video parameters in /etc/default/grub allows manual setting of the virtual console resolution:

```
GRUB_CMDLINE_LINUX_DEFAULT=" video=1920x1080 "
```

Remember to run the grub-mkconfig to invoke the changes (and copy the the configuration file to the correct path if necessary), then reboot. If case of console blind booting, then at least you can use Plasma’s Konsole in Xorg to change the settings and if necessary copying the configuration file.

```
# grub-mkconfig -o /boot/grub/grub.cfg
# cp -i /boot/grub/grub.cfg /boot/efi/EFI/grub/
# reboot
```

This may still fail if the necessary initramfs source file is not available. What follows is workaround that may no longer work in Gentoo. This workaround uses the iniramfs source file of uvesafb framebuffer driver. It is installed by emerging the package.

```
# emerge -qan v86d
```

After installation, the kernel should be rebuilt with the following options:

```
General setup --->
[*] Initial RAM filesystem and RAM disk (initramds/initrd) support
(/usr/share/v86/initramfs Initramfs source file(s))
```

Recompile the kernel, update grub, then reboot, keep your fingers crossed, but just don’t hold your breath.

16 Installing software

Now the choices are very much up to you on what to install. Note it is not always necessary to have software installed from source. For example Libreoffice, Firefox, and Thunderbird, can be installed from binaries if you prefer.

```
# emerge -qan libreoffice-bin firefox-bin thunderbird-bin
```

Note that execution of firefox and thunderbird installed in this way must be done from their binary equivalents i.e. firefox-bin and thunderbird-bin respectively. Now before you go stampeding into installing your favourite desktop programs, consider for a moment the utility of console-based programs. For convenience it is a good idea to install nano and links in case you need them for when your Xorg is broken.

```
# emerge -qan nano links
```

There are a host of other console-based programs that you are likely to find indispensible if you spend any time using them but you need to read their associated documentation to make the most of them. A couple of very useful packages particularly in virtual consoles is GNU screen and tmux.

```
# emerge -qan app-misc/screen tmux
```

There are also a number of console-based programs that provide powerful console interfaces to access newsgroups, Internet relay chat, and HTML webpages.

```
# emerge -qan slrn irssi lynx
```

Before installing the mail client 'mutt', bear in mind certain USE flags may be necessary for supporting specific security protocols.

```
USE="... imap smtp sasl ..."
```

When installing mutt, you might also want to consider urlview for viewing HTML links included within mail bodies.

```
# emerge -qan mutt urlview
```

You might consider installing the console spreadsheet calculator sc:

```
# emerge -qan sc
```

If your machine is sufficiently powerful, you might also consider installing the text editor emacs.

```
# emerge -qan emacs
```

Notice if you already have Vim installed, but want to add optional support/bindings for Python it needs to be re-compiled with the new USE flag. First add Python to the USE list.

```
USE="... python ..."
```

Then update it using emerge.

```
# emerge -DaquN vim
```

If you are looking for an extremely capable typesetting engine, then TEXLive is second to none.

```
# emerge -qan texlive \
  texlive-science texlive-humanities texlive-latexextra
```

There is a vector-graphics code-based engine, called Asymptote, that works very harmoniously with LATEX. Asymptote renders graphics through OpenGL and therefore needs to run within a Xorg server. You might also want to install gv for viewing EPS files.

```
# emerge -qan asymptote gv
```

Note that Asymptote installation will almost certainly come with USE changes required to include a few Python-based packages.

While I might reluctantly admit that video playback may be a bit tricky on virtual consoles, there are a number of console-based audio packages that you might find is more than everything you require for your basic sound needs.

```
# emerge -qan cdparanoia lame mpd ncmpcpp pulseaudio
```

Many of these console-based applications have to be configured individually (often by editing an /.*rc file) but there is plenty of information online on how to do this. The bad news is that you have to read it. But the good news is that once you've created your configuration files, you can keep them for good and tinker with them to your heart's content. At this point the software choices are down to your preferences, and if you don't like my console-based program suggestions that's absolutely fine because it's entirely up to you. You can search for specific packages using emerge before installing e.g.

```
# emerge --search pavucontrol
# emerge -qan media-sound/pavucontrol
# emerge --search kate
# emerge -qan kde-apps/kate
# emerge --search okular
# emerge -qan kde-apps/okular
```

And beyond that, the @world is your oyster.

17 Updating Gentoo

Whereas installation is not difficult just tedious, maintenance is both difficult and tedious although the level of difficulty will depend on the combination of packages you have installed. However sooner or later you will need to update Gentoo. I recommend doing so weekly-fortnightly. Be prepared to lose possibly an entire day, or even a few, updating Gentoo if you have not done so for a long time, such as over a month. Unfortunately there is no 'right way' to update Gentoo that is guaranteed to work every time because every installation is different. The jury even is still out on recommending any 'pragmatic solution' that usually 'should just work'. Any such recommendation can only be credible coming from someone in an asbestos suit and should therefore be taken with a pinch of salt. All that is listed here is what I generally do, which may or may not work for you.

The first step is to retrieve a Portage snapshot of available software, updating the Portage tree, and updating the Portage program.

```
# emerge-webrsync
# emerge -q --sync
# emerge -qau portage
```

The above commands should always work. However, the next command probably won't work without configuration changes, but run it anyway:

```
# emerge -DaquN --with-bdeps=y @world
```

Look at the list and be prepared to groan for when it turns out to be a very long one. There will probably be a number of configuration changes to be made. If there are not, consider yourself very lucky and apply the updates before Portage changes its mind; then cross your fingers during the hopefully error-free update before jumping to the stage described below commencing with the 'emerge -a –depclean' command.

If there are changes required, this is normal and you might want to boil the kettle to make a cup of tea. Do not worry about implementing all the changes just for now. First check the package updates and see if there are any version changes to the Linux headers or GNU packages listed immediately below. If there are, then update them first.

```
# emerge -qau linux-headers gcc glibc binutils libtool gcc-config
```

If there are changes to the compiler package, select the new one and once again reinstall the GNU packages.

```
# gcc-config -l
# gcc-config 1
# env-update && source /etc/profile
# emerge -qau gcc glibc binutils libtool gcc-config
```

In order to update the base system attempt the following update command.

```
# emerge -DaquN --with-bdeps=y @system
```

If there are no required configuration changes then you are in luck. Otherwise you will prompted to make configuration changes. Note that if you answer 'Y', Portage rather misleadingly won't make the changes. Instead it will create ._cfg* files inside /etc/portage/ that will list the necessary changes. You need to decide which config changes are made. A convenient package for listing, comparing, and implementing the config changes is etc-update:

```
# etc-update
```

In general, the best advice is to keep the original version of all configuration files you customised yourself (as long as you remember), and update or replace other configuration files (notably the package files). Then once again, try updating the system packages.

```
# emerge -DaquN --with-bdeps=y @system
```

On some occasions you may need to unmask packages listed in /usr/portage/package.mask. There are three other common pitfalls here. The first is called 'circular dependencies' where two packages may appear to mutually depend on one another and seem to cause Portage to sulk irrevocably. Often, the best solution is a 'oneshot' installation of one of the affected packages e.g.:

```
# emerge -qau --oneshot ncurses
```

A oneshot update does not affect @world. The second pitfall is the 'slot conflict'. These conflicts can be horrible and require many hours trying to resolve the conflict. In such cases, it's sometimes easier to unmerge troublesome packages before the update and reinstall them afterwards. The voodoo fun that is Gentoo is the necessity of wielding just the right black magic during updates to maintain a balanced karma between installed packages.

The final and worst pitfall is a compiler error. You may have no idea why it fails. It is highly improbable to be a result of a typo in the source code, but may be an issue associated with compiler flags, dependencies, or a packaging bug. A clue as to the cause may be evident from the build log. You will almost certainly need to submit this build log to the Gentoo forum if you want online help. If you update a library and there turn's out to be problem from a subset of packages that depend on it, the old version is often recoverable and it might be possible to update packages that depend on the old version. After a troublesome library update, this subset might be reparably updated and cleaned by emerge:

```
# emerge -qa @preserved-rebuild
```

The system set update needs to run perfectly before attempting the world set. Before attempting the same to the @world set as you have done for @system, you might want to edit /var/lib/portage/world and clean the list of unnecessary packages. Then once again attempt to update world.

```
# emerge -DaquN --with-bdeps=y @world
```

Any issues regarding configuration changes, dependencies, or conflicts should be handled as described previously for @system. Even if everything proceeds smoothly, this can take some time. Once done, unused packages can be cleaned out with a single command, then any broken libraries can be rebuilt:

```
# emerge -a --depclean
# revdep-rebuild -i -- -aq
```

The cost of updating and cleaning is that at some point, old kernel sources will be removed. When this happens, attempts of configuring the kernel compile will be thwarted with this sort of error message:

```
# cd /usr/src/linux
# make menuconfig
make: *** No rule to make target 'menuconfig'.  Stop.
```

The only way round this issue is to instruct Gentoo to use the new kernel. The first step is to backup the previous kernel configuration and copy it to the new kernel directory.

```
# cd /usr/src/
# mkdir oldconfig
# cp -i linux/.config* oldconfig
# cp -i linux/.config* /usr/src/linux-NN-N/
```

The next step is to direct Portage to the latest kernel.

```
# eselect kernel list
# eselect kernel set 1
# ls -lrt /usr/src/
```

The last command simply confirms that the symbolic link /usr/src/linux/ is directed to the new kernel sources. As long as you have copied the old configuration file to the new kernel source directory, you can use it for the basis of configuring the new kernel. The configuration needs to updated to accommodate for changes resulting from any additional features.

```
# cd /usr/src/linux
# make silentoldconfig
```

Typically it will invite you include new drivers, and you should have a low threshold to select ‘no’ if your system is already working perfectly well. If you are not sure, then the default option is usually a safe strategy.

After completing the extended interview, re-make externel modules and rebuild dependent modules.

```
# make modules_prepare
# emerge -a @module-rebuild
```

Finally copy the confguration, rebuild the kernel, update the bootloader configuration, and reboot as normal.

```
# cp -i .config .config.copy
# make && make modules_install
# make install
# genkernel --install initramfs
# grub_mkconfig -o /boot/grub/grub.cfg
# cp -i /boot/grub/grub.cfg /boot/efi/EFI/grub/
# reboot
```