Gentoo Linux amd64 Handbook: Installing Gentoo

From Gentoo Wiki

Handbook:AMD64 (/wiki/Special:MyLanguage/Handbook:AMD64) | Full (/wiki/Special:MyLanguage/Handbook:AMD64/Full)

Contents

Introduction

Welcome

First of all, welcome to Gentoo. You are about to enter the world of choices and performance. Gentoo is all about choices. When installing Gentoo, this is made clear several times -- users can choose how much they want to compile themselves, how to install Gentoo, what system logger to use, etc.

Gentoo is a fast, modern meta-distribution with a clean and flexible design. Gentoo is built around free software and doesn't hide from its users what is beneath the hood. Portage, the package maintenance system which Gentoo uses, is written in Python, meaning the user can easily view and modify the source code. Gentoo's packaging system uses source code (although support for precompiled packages is included too) and configuring Gentoo happens through regular text files. In other words, openness everywhere.

It is very important that everybody understands that choices are what makes Gentoo run. We try not to force users onto anything they don't like. If anyone believes otherwise, please bugreport (https://bugs.gentoo.org) it.

How is the installation structured

The Gentoo Installation can be seen as a 10-step procedure, corresponding to the next set of chapters. Every step results in a certain state:

1. After step 1, the user is in a working environment ready to install Gentoo
2. After step 2, the Internet connection is ready to install Gentoo
3. After step 3, the hard disks are initialized to host the Gentoo installation
4. After step 4, the installation environment is prepared and the user is ready to chroot into the new environment
5. After step 5, core packages, which are the same on all Gentoo installations, are installed
6. After step 6, the Linux kernel is installed
7. After step 7, the user will have configured most of the Gentoo system configuration files
8. After step 8, the necessary system tools are installed
9. After step 9, the proper boot loader has been installed and configured
10. After step 10, the freshly installed Gentoo Linux environment is ready to be explored

Whenever a certain choice is presented, the handbook will try to explain what the pros and cons are. Although the text then continues with a default choice (identified by "Default: " in the title), the other possibilities will be documented as well (marked by "Alternative: " in the title). Do not think that the default is what Gentoo recommends. It is however what Gentoo believes most users will use.

Sometimes an optional step can be followed. Such steps are marked as "Optional: " and are therefore not needed to install Gentoo. However, some optional steps are dependent on a previous decision made. The instructions will inform the reader when this happens, both when the decision is made, and right before the optional step is described.

What are the options

Gentoo can be installed in many different ways. It can be downloaded and installed from one of Gentoo's Installation CDs, from a distribution already installed, from a non-Gentoo bootable CD (such as Knoppix), from a netbooted environment, from a rescue floppy, etc.

This document covers the installation using a Gentoo Installation CD or, in certain cases, netbooting.

> **Note**
> For help on the other installation approaches, including using non-Gentoo CDs, please read our Alternative Installation Guide (/wiki/Installation_alternatives).

We also provide a Gentoo Installation Tips & Tricks (/wiki/Gentoo_installation_tips_and_tricks) document that might be useful to read as well.

Troubles

If a problem is found in the installation (or in the installation documentation), please visit our bugtracking system (https://bugs.gentoo.org) and check if the bug is known. If not, please create a bugreport for it so we can take care of it. Do not be afraid of the developers who are assigned to the bugs -- they (generally) don't eat people.

Note though that, although this document is architecture-specific, it might contain references to other architectures as well. This is due to the fact that large parts of the Gentoo Handbook use source code that is common for all architectures (to avoid duplication of efforts and starvation of development resources). We will try to keep this to a minimum to avoid confusion.

If there is some uncertainty whether or not the problem is a user-problem (some error made despite having read the documentation carefully) or a software-problem (some error we made despite having tested the installation/documentation carefully) everybody is welcome to join #gentoo on irc.freenode.net. Of course, everyone is welcome otherwise too as our chat channel covers the broad Gentoo spectrum.

Speaking of which, if there are any additional questions regarding Gentoo, check out our Frequently Asked Questions (/wiki/FAQ), available from the Gentoo Wiki (/wiki/Main_Page). There are also FAQs (https://forums.gentoo.org/viewforum.php?f=40) on the Gentoo Forums (https://forums.gentoo.org).

Hardware requirements

Before we start, we first list what hardware requirements are needed to successfully install Gentoo on a amd64 box.

<table>
<tr><th></th><th>Minimal CD</th><th>LiveDVD</th></tr>
<tr><th>CPU</th><td colspan="2">Any AMD64 CPU or EM64T (http://en.wikipedia.org/wiki/EMT64#Intel_64) CPU (Core 2 Duo & Quad processors are EM64T)</td></tr>
<tr><th>Memory</th><td>256 MB</td><td>512 MB</td></tr>
<tr><th>Diskspace</th><td colspan="2">2.5 GB (excluding swap space)</td></tr>
<tr><th>Swapspace</th><td colspan="2">At least 256 MB</td></tr>
</table>

The Gentoo AMD64 project site (https://www.gentoo.org/proj/en/base/amd64/) is a good place to be for more information about Gentoo's AMD64 support.

Gentoo Linux installation CD

Minimal installation CD

The Gentoo minimal installation CD is a bootable CD which contains a self-sustained Gentoo environment. It allows the user to boot Linux from the CD. During the boot process the hardware is detected and the appropriate drivers are loaded. The CD is maintained by Gentoo developers and allows anyone to install Gentoo if an active Internet connection is available.

The Minimal Installation CD is called `install-amd64-minimal-<release>.iso`.

The occasional Gentoo LiveDVD

Occasionally, a special DVD is crafted by the Gentoo Ten project which can be used to install Gentoo. The instructions further down this chapter target the Minimal Installation CD so might be a bit different. However, the LiveDVD (or any other bootable Linux environment) supports getting a root prompt by just invoking `sudo su -` or `sudo -i` on a terminal.

What are stages then?

A stage3 tarball is an archive containing a minimal Gentoo environment, suitable to continue the Gentoo installation using the instructions in this manual. Previously, the Gentoo Handbook described the installation using one of three stage tarballs. While Gentoo still offers stage1 and stage2 tarballs, the official installation method uses the stage3 tarball. If you are interested in performing a Gentoo installation using a stage1 or stage2 tarball, please read the Gentoo FAQ on How do I Install Gentoo Using a Stage1 or Stage2 Tarball (/wiki/FAQ#How_do_I_Install_Gentoo_Using_a_Stage1_or_Stage2_Tarball.3F)?

Stage3 tarballs can be downloaded from `releases/amd64/autobuilds/` on any of the official Gentoo mirrors (https://www.gentoo.org/downloads/mirrors/) and are not provided by the installation CD.

Downloading and burning the CD

Download the media

The default installation media that Gentoo Linux uses are the *minimal installation CDs*, which host a bootable, very small Gentoo Linux environment with the right tools to install Gentoo Linux from. The CD images themselves can be downloaded from one of the many mirrors (https://www.gentoo.org/downloads/mirrors/) available.

On those mirrors, the minimal installation CDs can be found as follows:

1. Go to the `releases/` directory
2. Select the right architecture, such as `amd64/`
3. Select the `autobuilds/` directory
4. Select the `current-iso/` directory

Inside this location, the installation CD file is the file with the `.iso` suffix. For instance, take a look at the following listing:

CODE **Example list of downloadable files at releases/amd64/autobuilds/current-iso/**

```
[DIR] hardened/                                          05-Dec-2014 01:42    -
[   ] install-amd64-minimal-20141204.iso                 04-Dec-2014 21:04  208M
[   ] install-amd64-minimal-20141204.iso.CONTENTS        04-Dec-2014 21:04  3.0K
[   ] install-amd64-minimal-20141204.iso.DIGESTS         04-Dec-2014 21:04  740
[TXT] install-amd64-minimal-20141204.iso.DIGESTS.asc     05-Dec-2014 01:42  1.6K
[   ] stage3-amd64-20141204.tar.bz2                      04-Dec-2014 21:04  198M
[   ] stage3-amd64-20141204.tar.bz2.CONTENTS             04-Dec-2014 21:04  4.6M
[   ] stage3-amd64-20141204.tar.bz2.DIGESTS              04-Dec-2014 21:04  720
[TXT] stage3-amd64-20141204.tar.bz2.DIGESTS.asc          05-Dec-2014 01:42  1.5K
```

In the above example, the `install-amd64-minimal-20141204.iso` file is the minimal installation CD itself. But as can be seen, other related files exist as well:

- A `.CONTENTS` file which is a text file listing all files available on the installation CD. This file can be useful to verify if particular firmware or drivers are available on the installation CD before downloading it.
- A `.DIGESTS` file which contains the hash of the ISO file itself, in various hashing formats/algorithms. This file can be used to verify if the downloaded ISO file is corrupt or not.
- A `.DIGESTS.asc` file which not only contains the hash of the ISO file (like the `.DIGESTS` file), but also a cryptographic signature of that file. This can be used to both verify if the downloaded ISO file is corrupt or not, as well as verify that the download is indeed provided by the Gentoo Release Engineering team and has not been tampered with.

Ignore the other files available at this location for now - those will come back when the installation has proceeded further. Download the `.ISO` file and, if verification of the download is wanted, download the `.DIGESTS.asc` file for the ISO file as well. The `.CONTENTS` file does not need to be downloaded as the installation instructions will not refer to this file anymore, and the `.DIGESTS` file should contain the same information as the `.DIGESTS.asc` file, except that the latter also contains a signature on top of it.

Verifying the downloaded files

Note
This is an optional step and not necessary to install Gentoo Linux. However, it is recommended as it ensures that the downloaded file is not corrupt and has indeed been provided by the Gentoo Infrastructure team.

Through the `.DIGESTS` and `.DIGESTS.asc` files, the validity of the ISO file can be confirmed using the right set of tools. This verification is usually done in two steps:

1. First, the cryptographic signature is validated to make sure that the installation file is provided by the Gentoo Release Engineering team
2. If the cryptographic signature validates, then the checksum is verified to make sure that the downloaded file itself is not corrupted

Microsoft Windows based verification

On a Microsoft Windows system, chances are low that the right set of tools to verify checksums and cryptographic signatures are in place.

To first verify the cryptographic signature, tools such as GPG4Win (http://www.gpg4win.org/) can be used. After installation, the public keys of the Gentoo Release Engineering team need to be imported. The list of keys is available on the signatures page (https://www.gentoo.org/downloads/signatures/). Once imported, the user can then verify the signature of the `.DIGESTS.asc` file.

> **Important**
> This does not verify that the `.DIGESTS` file is correct, only that the `.DIGESTS.asc` file is. That also implies that the checksum should be verified against the values in the `.DIGESTS.asc` file, which is why the instructions above only refer to downloading the `.DIGESTS.asc` file.

The checksum itself can be verified using the Hashcalc application (http://www.sinf.gr/en/hashcalc.html), although many others exist as well. Most of the time, these tools will show the user the calculated checksum, and the user is requested to verify this checksum with the value that is inside the `.DIGESTS.asc` file.

Linux based verification

On a Linux system, the most common method for verifying the cryptographic signature is to use the app-crypt/gnupg (http://packages.gentoo.org/package/app-crypt/gnupg) software. With this package installed, the following commands can be used to verify the cryptographic signature of the `.DIGESTS.asc` file.

First, download the right set of keys as made available on the signatures page (https://www.gentoo.org/downloads/signatures/):

user $ `gpg --recv-keys 0xBB572E0E2D182910`

```
gpg: requesting key 0xBB572E0E2D182910 from hkp server pool.sks-keyservers.net
gpg: key 0xBB572E0E2D182910: "Gentoo Linux Release Engineering (Automated Weekly Release Key) <rele
ng@gentoo.org>" 1 new signature
gpg: 3 marginal(s) needed, 1 complete(s) needed, classic trust model
gpg: depth: 0  valid:   3  signed:  20  trust: 0-, 0q, 0n, 0m, 0f, 3u
gpg: depth: 1  valid:  20  signed:  12  trust: 9-, 0q, 0n, 9m, 2f, 0u
gpg: next trustdb check due at 2018-09-15
gpg: Total number processed: 1
gpg:         new signatures: 1
```

Next verify the cryptographic signature of the `.DIGESTS.asc` file:

user $ `gpg --verify install-amd64-minimal-20141204.iso.DIGESTS.asc`

```
gpg: Signature made Fri 05 Dec 2014 02:42:44 AM CET
gpg:                using RSA key 0xBB572E0E2D182910
gpg: Good signature from "Gentoo Linux Release Engineering (Automated Weekly Release Key) <releng@g
entoo.org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 13EB BDBE DE7A 1277 5DFD  B1BA BB57 2E0E 2D18 2910
```

To be absolutely certain that everything is valid, verify the fingerprint shown with the fingerprint on the Gentoo signatures page (https://www.gentoo.org/downloads/signatures/).

With the cryptographic signature validated, next verify the checksum to make sure the downloaded ISO file is not corrupted. The `.DIGESTS.asc` file contains multiple hashing algorithms, so one of the methods to validate the right one is to first look at the checksum registered in the `.DIGESTS.asc` file. For instance, to get the SHA512 checksum:

```
user $ grep -A 1 -i sha512 install-amd64-minimal-20141204.iso.DIGESTS.asc
```

```
# SHA512 HASH
364d32c4f8420605f8a9fa3a0fc55864d5b0d1af11aa62b7a4d4699a427e5144b2d918225dfb7c5dec8d3f0fe2cddb7cc30
6da6f0cef4f01abec33eec74f3024  install-amd64-minimal-20141204.iso
--
# SHA512 HASH
0719a8954dc7432750de2e3076c8b843a2c79f5e60defe43fcca8c32ab26681dfb9898b102e211174a895ff4c8c41ddd9e9
a00ad6434d36c68d74bd02f19b57f  install-amd64-minimal-20141204.iso.CONTENTS
```

In the above output, two SHA512 checksums are shown - one for the `install-amd64-minimal-20141204.iso` file and one for its accompanying `.CONTENTS` file. Only the first checksum is of interest, as it needs to be compared with the calculated SHA512 checksum which can be generated as follows:

```
user $ sha512sum install-amd64-minimal-20141204.iso
```

```
364d32c4f8420605f8a9fa3a0fc55864d5b0d1af11aa62b7a4d4699a427e5144b2d918225dfb7c5dec8d3f0fe2cddb7cc30
6da6f0cef4f01abec33eec74f3024  install-amd64-minimal-20141204.iso
```

As both checksums match, the file is not corrupted and the installation can continue.

Burning

Of course, with just an ISO file downloaded, the Gentoo Linux installation cannot be started. The ISO file needs to be burned on a CD to boot from, and in such a way that its *content* is burned on the CD, not just the file itself. Below a few common methods are described - a more elaborate set of instructions can be found in Our FAQ on burning an ISO file (/wiki/FAQ#How_do_I_burn_an_ISO_file.3F).

Burning with Microsoft Windows

On Microsoft Windows, a number of tools exist that support burning ISOs on CDs.

- With EasyCD Creator, select *File*, *Record CD from CD image*. Then change the *Files of type* to *ISO image file*. Then locate the ISO file and click *Open*. After clicking on *Start recording* the ISO image will be burned correctly onto the CD-R.
- With Nero Burning ROM, cancel the wizard which automatically pops up and select *Burn Image* from the *File* menu. Select the image to burn and click *Open*. Now hit the *Burn* button and watch the brand new CD being burnt.

Burning with Linux

On Linux, the ISO file can be burned on a CD using the `cdrecord` command, part of the app-cdr/cdrtools (http://packages.gentoo.org/package/app-cdr/cdrtools) package.

For instance, to burn the ISO file on the CD in the `/dev/sr0` device (this is the first CD device on the system - substitute with the right device file if necessary):

```
user $ cdrecord dev=/dev/sr0 install--minimal-20141204.iso
```

Users that prefer a graphical user interface can use K3B, part of the app-cdr/k3b (http://packages.gentoo.org/package/app-cdr/k3b) package. In K3B, go to *Tools* and use *Burn CD Image*. Then follow the instructions provided by K3B.

Booting the CD

Booting the installation CD

Once the installation CD is burned, it is time to boot it. Remove all CDs from the CD drives, reboot the system and enter the BIOS or UEFI. This is usually done by hitting `DEL`, `F1` or `ESC`, which is highly depending on the system and motherboard used. Inside the BIOS or UEFI menu, change the boot order so that the CD-ROM is tried before the hard disk. Without this change, the system will just reboot from the hard disk, ignoring the CD-ROM.

> **Important**
> When installing Gentoo with the purpose of using the UEFI interface instead of BIOS, it is recommended to boot with UEFI immediately. If not, then it might be necessary to create a bootable UEFI USB stick (or other medium) once before finalizing the Gentoo Linux installation.

Now place the installation CD in the CD-ROM drive and reboot. A boot prompt should be shown. At this screen, `Enter` will begin the boot process with the default boot options. To boot the installation CD with custom boot options, specify a kernel followed by boot options and then hit `Enter`.

At the boot prompt, users get the option of displaying the available kernels (`F1`) and boot options (`F2`). If no choice is made within 15 seconds (either displaying information or using a kernel) then the installation CD will fall back to booting from disk. This allows installations to reboot and try out their installed environment without the need to remove the CD from the tray (something well appreciated for remote installations).

We mentioned specifying a kernel. On the installation CD, several kernels are provided. The default one is called *gentoo*. Other kernels are for specific hardware needs and the *-nofb* variants disable framebuffer support.

The next table gives a short overview of the available kernels.

Kernel	Description
gentoo	Default kernel with support for K8 CPUs (including NUMA support) and EM64T CPUs
gentoo-nofb	Same as *gentoo* but without framebuffer support
memtest86	Test the local RAM for errors

Alongside the kernel, boot options help in tuning the boot process further.

Hardware options	
acpi=on	This loads support for ACPI and also causes the acpid daemon to be started by the CD on boot. This is only needed if the system requires ACPI to function properly. This is not required for Hyperthreading support.
acpi=off	Completely disables ACPI. This is useful on some older systems and is also a requirement for using APM. This will disable any Hyperthreading support of your processor.
console=X	This sets up serial console access for the CD. The first option is the device, usually ttyS0 on x86, followed by any connection options, which are comma separated. The default options are 9600,8,n,1.
dmraid=X	This allows for passing options to the device-mapper RAID subsystem. Options should be encapsulated in quotes.
doapm	This loads APM driver support. This also requires that `acpi=off`.
dopcmcia	This loads support for PCMCIA and Cardbus hardware and also causes the pcmcia cardmgr to be started by the CD on boot. This is only required when booting from PCMCIA/Cardbus devices.
doscsi	This loads support for most SCSI controllers. This is also a requirement for booting most USB devices, as they use the SCSI subsystem of the kernel.

sda=stroke	This allows the user to partition the whole hard disk even when the BIOS is unable to handle large disks. This option is only used on machines with an older BIOS. Replace sda with the device that requires this option.
ide=nodma	This forces the disabling of DMA in the kernel and is required by some IDE chipsets and also by some CDROM drives. If the system is having trouble reading from the IDE CDROM, try this option. This also disables the default hdparm settings from being executed.
noapic	This disables the Advanced Programmable Interrupt Controller that is present on newer motherboards. It has been known to cause some problems on older hardware.
nodetect	This disables all of the autodetection done by the CD, including device autodetection and DHCP probing. This is useful for doing debugging of a failing CD or driver.
nodhcp	This disables DHCP probing on detected network cards. This is useful on networks with only static addresses.
nodmraid	Disables support for device-mapper RAID, such as that used for on-board IDE/SATA RAID controllers.
nofirewire	This disables the loading of Firewire modules. This should only be necessary if your Firewire hardware is causing a problem with booting the CD.
nogpm	This disables gpm console mouse support.
nohotplug	This disables the loading of the hotplug and coldplug init scripts at boot. This is useful for doing debugging of a failing CD or driver.
nokeymap	This disables the keymap selection used to select non-US keyboard layouts.
nolapic	This disables the local APIC on Uniprocessor kernels.
nosata	This disables the loading of Serial ATA modules. This is used if the system is having problems with the SATA subsystem.
nosmp	This disables SMP, or Symmetric Multiprocessing, on SMP-enabled kernels. This is useful for debugging SMP-related issues with certain drivers and motherboards.
nosound	This disables sound support and volume setting. This is useful for systems where sound support causes problems.
nousb	This disables the autoloading of USB modules. This is useful for debugging USB issues.
slowusb	This adds some extra pauses into the boot process for slow USB CDROMs, like in the IBM BladeCenter.
Logical volume/device management	
dolvm	This enables support for Linux's Logical Volume Management.
Other options	
debug	Enables debugging code. This might get messy, as it displays a lot of data to the screen.
docache	This caches the entire runtime portion of the CD into RAM, which allows the user to umount `/mnt/cdrom` and mount another CDROM. This option requires that there is at least twice as much available RAM as the size of the CD.
doload=X	This causes the initial ramdisk to load any module listed, as well as dependencies. Replace X with the module name. Multiple modules can be specified by a comma-separated list.
dosshd	Starts sshd on boot, which is useful for unattended installs.
passwd=foo	Sets whatever follows the equals as the root password, which is required for *dosshd* since the root

	password is by default scrambled.
noload=X	This causes the initial ramdisk to skip the loading of a specific module that may be causing a problem. Syntax matches that of doload.
nonfs	Disables the starting of portmap/nfsmount on boot.
nox	This causes an X-enabled LiveCD to not automatically start X, but rather, to drop to the command line instead.
scandelay	This causes the CD to pause for 10 seconds during certain portions the boot process to allow for devices that are slow to initialize to be ready for use.
scandelay=X	This allows the user to specify a given delay, in seconds, to be added to certain portions of the boot process to allow for devices that are slow to initialize to be ready for use. Replace X with the number of seconds to pause.

Note
The CD will check for `no*` options before `do*` options, so that options can be overriden in the exact order specified.

Now boot the CD, select a kernel (if the default `gentoo` kernel does not suffice) and boot options. As an example, we boot the `gentoo` kernel, with `dopcmcia` as a kernel parameter:

```
boot: gentoo dopcmcia
```

Next the user will be greeted with a boot screen and progress bar. If the installation is done on a system with a non-US keyboard, make sure to immediately press `Alt` + `F1` to switch to verbose mode and follow the prompt. If no selection is made in 10 seconds the default (US keyboard) will be accepted and the boot process will continue. Once the boot process completes, the user is automatically logged in to the "Live" Gentoo Linux environment as the *root* user, the super user. A root prompt is displayed on the current console, and one can switch to other consoles by pressing `Alt` + `F2`, `Alt` + `F3` and `Alt` + `F4`. Get back to the one started on by pressing `Alt` + `F1`.

Extra hardware configuration

When the Installation CD boots, it tries to detect all the hardware devices and loads the appropriate kernel modules to support the hardware. In the vast majority of cases, it does a very good job. However, in some cases it may not auto-load the kernel modules needed by the system. If the PCI auto-detection missed some of the system's hardware, the appropriate kernel modules have to be loaded manually.

In the next example the 8139too module (which supports certain kinds of network interfaces) is loaded:

```
root # modprobe 8139too
```

Optional: User accounts

If other people need access to the installation environment, or there is need to run commands as a non-root user on the installation CD (such as to chat using `irssi` without root privileges for security reasons), then an additional user account needs to be created and the root password set to a strong password.

To change the root password, use the passwd utility:

```
root # passwd
```

```
New password: (Enter your new password)
Re-enter password: (Re-enter your password)
```

To create a user account, first enter their credentials, followed by the account's password. The `useradd` and `passwd` commands are used for these tasks.

In the next example, a user called *john* is created:

```
root # useradd -m -G users john
root # passwd john
```

```
New password: (Enter john's password)
Re-enter password: (Re-enter john's password)
```

To switch from the (current) *root* user to the newly created user account, use the `su` command:

```
root # su - john
```

Optional: Viewing documentation while installing

To view the Gentoo handbook during the installation, first create a user account as described above. Then press `Alt` + `F2` to go to a new terminal.

During the installation, the `links` command can be used to browse the Gentoo handbook - of course only from the moment that the Internet connection is working.

```
user $ links                                           (https://wiki.gentoo.org/wiki/Handbook:AMD64)
```

To go back to the original terminal, press `Alt` + `F1`.

Optional: Starting the SSH daemon

To allow other users to access the system during the installation (perhaps to support during an installation, or even do it remotely), a user account needs to be created (as was documented earlier on) and the SSH daemon needs to be started.

To fire up the SSH daemon, execute the following command:

```
root # /etc/init.d/sshd start
```

Note

If users log on to the system, they will get a message that the host key for this system needs to be confirmed (through what is called a fingerprint). This is to be expected as it is the first time people log on to the system. However, later when the system is set up and someone logs on to the newly created system, the SSH client will warn that the host key has been changed. This is because the user now log on to - for SSH - a different server (namely the freshly installed Gentoo system rather than the live environment that the installation is currently using). Follow the instructions given on the screen then to replace the host key on the client system.

To be able to use sshd, the network needs to function properly. Continue with the chapter on Configuring the network (/wiki/Handbook:AMD64/Installation/Networking).

Automatic network detection

Maybe it just works?

If the system is plugged into an Ethernet network with a DHCP server, it is very likely that the networking configuration has already been set up automatically. If so, then the many included network-aware commands on the installation CD such as `ssh`, `scp`, `ping`, `irssi`, `wget` and `links`, among others, will work immediately.

If networking has been configured, the `ifconfig` command should list some network interfaces besides `lo`, such as `eth0`:

```
root # ifconfig
```

```
(...)
eth0      Link encap:Ethernet  HWaddr 00:50:BA:8F:61:7A
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::50:ba8f:617a/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1498792 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1284980 errors:0 dropped:0 overruns:0 carrier:0
          collisions:1984 txqueuelen:100
          RX bytes:485691215 (463.1 Mb)  TX bytes:123951388 (118.2 Mb)
          Interrupt:11 Base address:0xe800
```

The interface name on the system can be quite different from `eth0`. Recent installation media might show regular network interfaces names like `eno0`, `ens1` or `enp5s0`. Just seek the interface in the `ifconfig` output that has an IP address related to the local network.

In the remainder of this document, we will assume that the interface is called `eth0`.

Optional: Configure any proxies

If the Internet is accessed through a proxy, then it is necessary to set up proxy information during the installation. It is very easy to define a proxy: just define a variable which contains the proxy server information.

In most cases, it is sufficient to define the variables using the server hostname. As an example, we assume the proxy is called proxy.gentoo.org and the port is 8080.

To set up an HTTP proxy (for HTTP and HTTPS traffic):

```
root # export http_proxy="                              (http://proxy.gentoo.org:8080)"
```

To set up an FTP proxy:

```
root # export ftp_proxy="                              (ftp://proxy.gentoo.org:8080)"
```

To set up an RSYNC proxy:

```
root # export RSYNC_PROXY="proxy.gentoo.org:8080"
```

If the proxy requires a username and password, use the following syntax for the variable:

CODE **Adding username/password to the proxy variable**

```
http://username:password@proxy.gentoo.org:8080
```

Testing the network

Try pinging your ISP's DNS server (found in `/etc/resolv.conf`) and a web site of choice. This ensures that the network is functioning properly and that the network packets are reaching the net, DNS name resolution is working correctly, etc.

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

If this all works, then the remainder of this chapter can be skipped to jump right to the next step of the installation instructions (Preparing the disks (/wiki/Handbook:AMD64/Installation/Disks)).

Automatic network configuration

If the network doesn't work immediately, some installation media allow the user to use `net-setup` (for regular or wireless networks), `pppoe-setup` (for ADSL users) or `pptp` (for PPTP users).

If the installation medium does not contain any of these tools, continue with the Manual network configuration.

- Regular Ethernet users should continue with Default: Using net-setup
- ADSL users should continue with Alternative: Using PPP
- PPTP users should continue with Alternative: Using PPTP

Default: Using net-setup

The simplest way to set up networking if it didn't get configured automatically is to run the `net-setup` script:

```
root # net-setup eth0
```

`net-setup` will ask some questions about the network environment. When all is done, the network connection should work. Test the network connection as stated before. If the tests are positive, congratulations! Skip the rest of this section and continue with Preparing the disks (/wiki/Handbook:AMD64/Installation/Disks).

If the network still doesn't work, continue with Manual network configuration.

Alternative: Using PPP

Assuming PPPoE is needed to connect to the Internet, the installation CD (any version) has made things easier by including ppp. Use the provided `pppoe-setup` script to configure the connection. During the setup the Ethernet device that is connected to your ADSL modem, the username and password, the IPs of the DNS servers and if a basic firewall is needed or not will be asked.

```
root # pppoe-setup
```

```
root # pppoe-start
```

If something goes wrong, double-check that the username and password are correct by looking at `etc/ppp/pap-secrets` or `/etc/ppp/chap-secrets` and make sure to use the right Ethernet device. If the Ethernet device doesn't exist, the appropriate network modules need to be loaded. In that case continue with Manual network configuration as it will explain how to load the appropriate network modules there.

If everything worked, continue with Preparing the disks (/wiki/Handbook:AMD64/Installation/Disks).

Alternative: Using PPTP

If PPTP support is needed, use `pptpclient` which is provided by the installation CDs. But first make sure that the configuration is correct. Edit `/etc/ppp/pap-secrets` or `/etc/ppp/chap-secrets` so it contains the correct username/password combination:

```
root # nano -w /etc/ppp/chap-secrets
```

Then adjust `/etc/ppp/options.pptp` if necessary:

```
root # nano -w /etc/ppp/options.pptp
```

When all that is done, just run `pptp` (along with the options that couldn't be set in `options.pptp`) to connect the server:

```
root # pptp <server ip>
```

Now continue with Preparing the disks (/wiki/Handbook:AMD64/Installation/Disks).

Manual network configuration

Loading the appropriate network modules

When the Installation CD boots, it tries to detect all the hardware devices and loads the appropriate kernel modules (drivers) to support the hardware. In the vast majority of cases, it does a very good job. However, in some cases, it may not auto-load the kernel modules needed.

If `net-setup` or `pppoe-setup` failed, then it is possible that the network card wasn't found immediately. This means users may have to load the appropriate kernel modules manually.

To find out what kernel modules are provided for networking, use `ls` :

```
root # ls /lib/modules/`uname -r`/kernel/drivers/net
```

If a driver is found for the network device, use `modprobe` to load the kernel module. For instance, to load the `pcnet32` module:

```
root # modprobe pcnet32
```

To check if the network card is now detected, use `ifconfig` . A detected network card would result in something like this (again, `eth0` here is just an example):

```
root # ifconfig eth0
```

```
eth0      Link encap:Ethernet  HWaddr FE:FD:00:00:00:00
          BROADCAST NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

If however the following error is shown, the network card is not detected:

```
root # ifconfig eth0
```

```
eth0: error fetching interface information: Device not found
```

The available network interface names on the system can be listed through the `/sys` file system:

```
root # ls /sys/class/net
```

```
dummy0  eth0  lo  sit0  tap0  wlan0
```

In the above example, 6 interfaces are found. The `eth0` one is most likely the (wired) Ethernet adapter whereas `wlan0` is the wireless one.

Assuming that the network card is now detected, retry `net-setup` or `pppoe-setup` again (which should work now), but for the hardcore people we explain how to configure the network manually as well.

Select one of the following sections based on your network setup:

- Using DHCP for automatic IP retrieval
- Preparing for wireless access if a wireless network is used
- Understanding network terminology explains the basics about networking
- Using ifconfig and route explains how to set up networking manually

Using DHCP

DHCP (Dynamic Host Configuration Protocol) makes it possible to automatically receive networking information (IP address, netmask, broadcast address, gateway, nameservers etc.). This only works if a DHCP server is in the network (or if the ISP provider provides a DHCP service). To have a network interface receive this information automatically, use `dhcpcd` :

```
root # dhcpcd eth0
```

Some network administrators require that the hostname and domainname provided by the DHCP server is used by the system. In that case, use:

```
root # dhcpcd -HD eth0
```

If this works (try pinging some Internet server, like Google), then everything is set and ready to continue. Skip the rest of this section and continue with Preparing the disks (/wiki/Handbook:AMD64/Installation/Disks).

Preparing for wireless access

> **Note**
> Support for the `iwconfig` command might be architecture-specific. If the command is not available, please follow the instructions of the linux-wlan-ng project (ftp://ftp.linux-wlan.org/pub/linux-wlan-ng/README).

When using a wireless (802.11) card, the wireless settings need to be configured before going any further. To see the current wireless settings on the card, one can use `iwconfig` . Running `iwconfig` might show something like:

```
root # iwconfig eth0
```

```
eth0      IEEE 802.11-DS  ESSID:"GentooNode"
          Mode:Managed  Frequency:2.442GHz  Access Point: 00:09:5B:11:CC:F2
          Bit Rate:11Mb/s   Tx-Power=20 dBm   Sensitivity=0/65535
          Retry limit:16   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality:25/10  Signal level:-51 dBm  Noise level:-102 dBm
          Rx invalid nwid:5901 Rx invalid crypt:0 Rx invalid frag:0 Tx
          excessive retries:237 Invalid misc:350282 Missed beacon:84
```

> **Note**
> Some wireless cards may have a device name of `wlan0` or `ra0` instead of `eth0`. Run `iwconfig` without any command-line parameters to determine the correct device name.

For most users, there are only two settings that might be important to change, the ESSID (aka wireless network name) or the WEP key. If the ESSID and Access Point address listed are already those of the environment's access point and the environment is not not using WEP, then the wireless configuration is already working.

To change the ESSID, or add a WEP key, issue the following commands.

- To set the network name to *GentooNode*:

```
root # iwconfig eth0 essid GentooNode
```

- To set a hex WEP key:

```
root # iwconfig eth0 key 1234123412341234abcd
```

To set an ASCII WEP key, prefix the key with `s:` :

```
root # iwconfig eth0 key s:some-password
```

> **Note**

If the wireless network is set up with WPA or WPA2, then `wpa_supplicant` needs to be used. For more information on configuring wireless networking in Gentoo Linux, please read the Wireless networking chapter (/wiki/Handbook:AMD64/Networking/Wireless) in the Gentoo Handbook.

Confirm the wireless settings again by using `iwconfig`. Once wireless is working,continue configuring the IP level networking options as described in the next section (Understanding network terminology) or use the `net-setup` tool as described previously.

Understanding network terminology

Note
If the IP address, broadcast address, netmask and nameservers are known, then skip this subsection and continue with Using ifconfig and route.

If all of the above fails, the network will need to be configured manually. This is not difficult at all. However, some knowledge of network terminology and basic concepts might be necessary. After reading this section, users will know what a gateway is, what a netmask serves for, how a broadcast address is formed and why systems need nameservers.

In a network, hosts are identified by their IP address (Internet Protocol address). Such an address is perceived as a combination of four numbers between 0 and 255. Well, at least when using IPv4 (IP version 4). In reality, such an IPv4 address consists of 32 bits (ones and zeros). Let's view an example:

CODE **Example of an IPv4 address**

```
IP Address (numbers):   192.168.0.2
IP Address (bits):      11000000 10101000 00000000 00000010
                        -------- -------- -------- --------
                           192      168       0        2
```

Note
The successor of IPv4, IPv6, uses 128 bits (ones and zeros). In this section, the focus is on IPv4 addresses.

Such an IP address is unique to a host as far as all accessible networks are concerned (i.e. every host that one wants to be able to reach must have a unique IP address). In order to distinguish between hosts inside and outside a network, the IP address is divided in two parts: the network part and the host part.

The separation is written down with the netmask, a collection of ones followed by a collection of zeros. The part of the IP that can be mapped on the ones is the network-part, the other one is the host-part. As usual, the netmask can be written down as an IP address.

CODE **Example of network/host separation**

```
IP address:    192      168      0        2
            11000000 10101000 00000000 00000010
Netmask:    11111111 11111111 11111111 00000000
               255      255     255       0
           +--------------------------+--------+
                  Network                Host
```

In other words, 192.168.0.14 is still part of the example network, but 192.168.1.2 is not.

The broadcast address is an IP address with the same network-part as the network, but with only ones as host-part. Every host on the network listens to this IP address. It is truly meant for broadcasting packets.

CODE **Broadcast address**

```
IP address:    192      168      0        2
            11000000 10101000 00000000 00000010
Broadcast:  11000000 10101000 00000000 11111111
               192      168      0       255
           +--------------------------+--------+
                     Network             Host
```

To be able to surf on the Internet, each computer in the network must know which host shares the Internet connection. This host is called the gateway. Since it is a regular host, it has a regular IP address (for instance 192.168.0.1).

Previously we stated that every host has its own IP address. To be able to reach this host by a name (instead of an IP address) we need a service that translates a name (such as dev.gentoo.org) to an IP address (such as 64.5.62.82). Such a service is called a *name service*. To use such a service, the necessary name servers need to be defined in `/etc/resolv.conf`.

In some cases, the gateway also serves as a nameserver. Otherwise the nameservers provided by the ISP need to be entered in this file.

To summarize, the following information is needed before continuing:

Network Item	Example
The system IP address	192.168.0.2
Netmask	255.255.255.0
Broadcast	192.168.0.255
Gateway	192.168.0.1
Nameserver(s)	195.130.130.5, 195.130.130.133

Using ifconfig and route

Setting up the network consists of three steps.

1. Assign an IP address using `ifconfig`
2. Set up routing to the gateway using `route`
3. Finish up by placing the nameserver IPs in `/etc/resolv.conf`

To assign an IP address, the IP address, broadcast address and netmask are needed. Then execute the following command, substituting `${IP_ADDR}` with the right IP address, `${BROADCAST}` with the right broadcast address and `${NETMASK}` with the right netmask:

```
root # ifconfig eth0 ${IP_ADDR} broadcast ${BROADCAST} netmask ${NETMASK} up
```

Set up routing using `route`. Substitute `${GATEWAY}` with the right gateway IP address:

```
root # route add default gw ${GATEWAY}
```

Now open `/etc/resolv.conf`:

```
root # nano -w /etc/resolv.conf
```

Fill in the nameserver(s) using the following as a template. Make sure to substitute `${NAMESERVER1}` and `${NAMESERVER2}` with the appropriate nameserver addresses:

CODE **Default template to use for /etc/resolv.conf**

```
nameserver ${NAMESERVER1}
nameserver ${NAMESERVER2}
```

That's it. Now test the network by pinging some Internet server (like Google). If this works, congratulations then. Continue with Preparing the disks (/wiki/Handbook:AMD64/Installation/Disks).

Introduction to block devices

Block devices

Let's take a good look at disk-oriented aspects of Gentoo Linux and Linux in general, including Linux filesystems, partitions and block devices. Once the ins and outs of disks and filesystems are understood, we will set up partitions and filesystems for the Gentoo Linux installation.

To begin, let's look at block devices. The most famous block device is probably the one that represents the first drive in a Linux system, namely `/dev/sda`. SCSI and Serial ATA drives are both labeled `/dev/sd*`; even IDE drives are labeled `/dev/sd*` with the new libata framework in the kernel. When using the old device framework, then the first IDE drive is `/dev/hda`.

The block devices above represent an abstract interface to the disk. User programs can use these block devices to interact with your disk without worrying about whether the drives are IDE, SCSI or something else. The program can simply address the storage on the disk as a bunch of contiguous, randomly-accessible 512-byte blocks.

Partitions

Although it is theoretically possible to use a full disk to house a Linux system, this is almost never done in practice. Instead, full disk block devices are split up in smaller, more manageable block devices. On AMD64 systems, these are called partitions. There are currently two standard partitioning technologies in use: MBR and GPT.

MBR

The *MBR (Master Boot Record)* setup uses 32-bit identifiers for the start sector and length of the partitions, and supports three partition types: primary, extended and logical. Primary partitions have their information stored in the master boot record itself - a very small (usually 512 bytes) location at the very beginning of a disk. Due to this small space, only four primary partitions are supported (for instance, `/dev/sda1` to `/dev/sda4`).

To support more partitions, one of the primary partitions can be marked as an extended partition. This partition can then contain logical partitions (partitions within a partition).

Each partition is limited to 2 TB in size (due to the 32-bit identifiers). Also, the MBR setup does not provide any backup-MBR, so if an application or user overwrites the MBR, all partition information is lost.

GPT

The *GPT (GUID Partition table)* setup uses 64-bit identifiers for the partitions. The location in which it stores the partition information is also much bigger than the 512 bytes of an MBR, and there is no limit on the amount of partitions. Also the size of a partition is bounded by a much greater limit (almost 8 ZB - yes, zettabytes).

When a system's software interface between the operating system and firmware is UEFI (instead of BIOS), GPT is almost mandatory as compatibility issues will arise with MBR here.

GPT also has the advantage that it has a backup GPT at the end of the disk, which can be used to recover damage of the primary GPT at the beginning. GPT also carries CRC32 checksums to detect errors in the header and partition tables.

GPT or MBR

From the description above, one might think that using GPT should always be the recommended approach. But there are a few caveats with this.

Using GPT on a BIOS-based computer works, but then one cannot dual-boot with a Microsoft Windows operating system. The reason is that Microsoft Windows will boot in EFI mode if it detects a GPT partition label.

Some buggy BIOSes or EFIs configured to boot in BIOS/CSM/legacy mode might also have problems with booting from GPT labeled disks. If that is the case, it might be possible to work around the problem by adding the boot/active flag on the protective MBR partition which has to be done through `fdisk` with the `-t dos` option to force it to read the partition table using the MBR format.

In this case, launch `fdisk` and toggle the flag using `a` on the first partition (`1`). Then write the changes to the disk (`w`) and exit the `fdisk` application:

user $ fdisk -t dos /dev/sda

```
Welcome to fdisk (util-linux 2.24.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): a
Partition number (1-4): 1

Command (m for help): w
```

Using UEFI

When installing Gentoo on a system that uses UEFI to boot the operating system (instead of BIOS), then it is important that an EFI system partition is created. The instructions for `parted` below contain the necessary pointers for this.

The EFI system partition also needs to be a FAT32 partition (or *vfat* as it is shown on Linux systems). The instructions at the end of this chapter use *ext2* as the example file system for the `/boot/` partition. Make sure to use *vfat*, like so:

root # mkfs.vfat /dev/sda2

Warning
If the boot partition is not using the FAT32 (vfat) file system, then the systems UEFI firmware will not be able to find the Linux kernel and boot the system!

Advanced storage

The AMD64 Installation CDs provide support for LVM2. LVM2 increases the flexibility offered by the partitioning setup. During the installation instructions, we will focus on "regular" partitions, but it is still good to know LVM2 is supported as well.

Default partitioning scheme

Throughout the remainder of the handbook, the following partitioning scheme is used. If this suffices, then the reader can immediately jump to Default: Using parted to partition the disk or Alternative: Using fdisk to partition the disk. Both are partitioning tools, `fdisk` is well known and stable and recommended for the MBR partition layout, while `parted` is more recent and recommended for GPT layouts.

Partition	Filesystem	Size	Description
/dev/sda1	(bootloader)	2M	BIOS boot partition
/dev/sda2	ext2 (or vfat)	128M	Boot partition
/dev/sda3	(swap)	512M or higher	Swap partition
/dev/sda4	ext4	Rest of the disk	Root partition

Before going to the creation instructions, the first set of sections will describe in more detail how partitioning schemes can be created and what the common pitfalls are.

Designing a partition scheme

How many and how big

The number of partitions is highly dependent on the environment. For instance, if there are lots of users, then it is advised to have `/home/` separate as it increases security and makes backups easier. If Gentoo is being installed to perform as a mail server, then `/var/` should be separate as all mails are stored inside `/var/`. A good choice of filesystem will then maximize the performance. Game servers will have a separate `/opt/` as most gaming servers are installed there. The reason is similar for `/home/`: security and backups. In most situations, `/usr/` is to be kept big: not only will it contain the majority of applications, the Portage tree alone takes around 500 Mbyte excluding the various sources that are stored in it.

As can be seen, it very much depends on what the administrator want to achieve. Separate partitions or volumes have the following advantages:

- Choose the best performing filesystem for each partition or volume
- The entire system cannot run out of free space if one defunct tool is continuously writing files to a partition or volume
- If necessary, file system checks are reduced in time, as multiple checks can be done in parallel (although this advantage is more with multiple disks than it is with multiple partitions)
- Security can be enhanced by mounting some partitions or volumes read-only, `nosuid` (setuid bits are ignored), `noexec` (executable bits are ignored) etc.

However, multiple partitions have disadvantages as well. If not configured properly, the system might have lots of free space on one partition and none on another. Another nuisance is that separate partitions - especially for important mount points like `/usr/` or `/var/` - often require the administrator to boot with an initramfs to mount the partition before other boot scripts start. This isn't always the case though, so results may vary.

There is also a 15-partition limit for SCSI and SATA unless the disk uses GPT labels.

What about swap space

There is no perfect value for the swap partition. The purpose of swap space is to provide disk storage to the kernel when internal memory (RAM) is under pressure. A swap space allows for the kernel to move memory pages that are not likely to be accessed soon to disk (swap or page-out), freeing memory. Of course, if that memory is suddenly needed, these pages need to be put back in memory (page-in) which will take a while (as disks are very slow compared to internal memory).

When the system is not going to run memory intensive applications or the system has lots of memory available, then it probably does not need much swap space. However, swap space is also used to store the entire memory in case of hibernation. If the system is going to need hibernation, then a bigger swap space necessary, often at least the amount of memory installed in the system.

What is the BIOS boot partition

A BIOS boot partition is a very small (1 to 2 MB) partition in which boot loaders like GRUB2 can put additional data that doesn't fit in the allocated storage (a few hundred bytes in case of MBR) and cannot be placed elsewhere.

Such partitions are not always necessary, but considering the low space consumption and the difficulties we have with documenting the plethora of partitioning differences otherwise, it is recommended to create it in either case.

For completeness, the BIOS boot partition is needed when GPT partition layout is used with GRUB2, or when the MBR partition layout is used with GRUB2 when the first partition starts earlier than the 1 MB location on the disk.

Default: Using parted to partition the disk

In this chapter, the example partition layout mentioned earlier in the instructions will be used:

Partition	Description
/dev/sda1	BIOS boot partition
/dev/sda2	Boot partition
/dev/sda3	Swap partition
/dev/sda4	Root partition

Change the partition layout according to personal preference.

Viewing the current partition layout with parted

The `parted` application offers a simple interface for partitioning the disks and supports very large partitions (more than 2 TB). Fire up `parted` against the disk (in our example, we use `/dev/sda`). It is recommended to ask `parted` to use optimal partition alignment:

root # `parted -a optimal /dev/sda`

```
GNU Parted 2.3
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
```

Alignment means that partitions are started on well-known boundaries within the disk, ensuring that operations on the disk from the operating system level (retrieve pages from the disk) use the least amount of internal disk operations. Misaligned partitions might require the disk to fetch two pages instead of one even if the operating system asked for a single page.

To find out about all options supported by parted, type `help` and press return.

Setting the GPT label

Most disks on x86/amd64 are prepared using an *msdos* label. Using `parted`, the command to put a GPT label on the disk is `mklabel gpt`:

Warning
Changing the partition type will remove all partitions from the disk. All data on the disk will be lost.

```
(parted) mklabel gpt
```

To have the disk with MBR layout, use `mklabel msdos`.

Removing all partitions with parted

If this isn't done yet (for instance through the `mklabel` operation earlier, or because the disk is a freshly formatted one), first remove all existing partitions from the disk. Type `print` to view the current partitions, and `rm NUMBER` where NUMBER is the partition to remove.

```
(parted) rm 2
```

Do the same for all other partitions that aren't needed. However, make sure to not make any mistakes here - parted executes the changes immediately (unlike `fdisk` which stages them, allowing a user to "undo" his changes before saving or exiting fdisk).

Creating the partitions

Now create the partitions. Creating partitions with `parted` isn't very difficult - all we need to do is inform parted about the following settings:

- The partition type to use. This usually is *primary*. If the msdos partition label is used, keep in mind that there can be no more than 4 primary partitions. If more than 4 partitions are needed, make one of the first four partitions extended and create logical partitions inside it.
- The start location of a partition (which can be expressed in MB, GB, ...)
- The end location of the partition (which can be expressed in MB, GB, ...)

First, tell parted that the size unit we work with is megabytes (actually mebibytes, abbreviated as MiB which is the "standard" notation, but we will use MB in the text throughout as it is much more common):

```
(parted) unit mib
```

Now create a 2 MB partition that will be used by the GRUB2 boot loader later. Use the `mkpart` command for this, and inform `parted` to start from 1 MB and end at 3 MB (creating a partition of 2 MB in size).

```
(parted) mkpart primary 1 3
(parted) name 1 grub
(parted) set 1 bios_grub on
(parted) print
```

```
Model: Virtio Block Device (virtblk)
Disk /dev/sda: 20480MiB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number   Start     End       Size     File system  Name   Flags
 1       1.00MiB   3.00MiB   2.00MiB               grub   bios_grub
```

Do the same for the boot partition (128 MB), swap partition (in the example, 512 MB) and the root partition that spans the remaining disk (for which the end location is marked as -1, meaning the end of the disk minus one MB, which is the farthest a partition can go).

```
(parted) mkpart primary 3 131
(parted) name 2 boot
```

```
(parted) mkpart primary 131 643
(parted) name 3 swap
(parted) mkpart primary 643 -1
(parted) name 4 rootfs
```

When using the UEFI interface to boot the system (instead of BIOS), mark the boot partition as the EFI System Partition. Parted does this automatically when the *boot* option is set on the partition:

```
(parted) set 2 boot on
```

The end result looks like so:

```
(parted) print
```

```
Model: Virtio Block Device (virtblk)
Disk /dev/sda: 20480MiB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number   Start     End      Size     File system  Name    Flags
 1       1.00MiB   3.00MiB  2.00MiB               grub    bios_grub
 2       3.00MiB   131MiB   128MiB                boot
 3       131MiB    643MiB   512MiB                swap
 4       643MiB    20479MiB 19836MiB              rootfs
```

Note
On an UEFI installation, the boot flag will show up on the boot partition.

Use the `quit` command to exit parted.

Alternative: Using fdisk to partition the disk

Note
Although recent `fdisk` should support GPT, it has still shown to have some issues with it. The instructions given below assume that the partition layout being used is MBR.

The following parts explain how to create the example partition layout using `fdisk`. The example partition layout was mentioned earlier:

Partition	Description
/dev/sda1	BIOS boot partition
/dev/sda2	Boot partition
/dev/sda3	Swap partition
/dev/sda4	Root partition

Change the partition layout according to personal preference.

Viewing the current partition layout

`fdisk` is a popular and powerful tool to split a disk into partitions. Fire up `fdisk` against the disk (in our example, we use `/dev/sda`):

root # fdisk /dev/sda

Note
To use GPT support, add `-t gpt`. It is recommended to closely investigate the `fdisk` output in case more recent developments in `fdisk` change its default behavior of defaulting to MBR. The remainder of the instructions assume an MBR layout.

Type `p` to display the disk's current partition configuration:

Command (m for help): p

```
Disk /dev/sda: 240 heads, 63 sectors, 2184 cylinders
Units = cylinders of 15120 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sda1   *         1        14    105808+  83  Linux
/dev/sda2            15        49    264600   82  Linux swap
/dev/sda3            50        70    158760   83  Linux
/dev/sda4            71      2184  15981840    5  Extended
/dev/sda5            71       209   1050808+  83  Linux
/dev/sda6           210       348   1050808+  83  Linux
/dev/sda7           349       626   2101648+  83  Linux
/dev/sda8           627       904   2101648+  83  Linux
/dev/sda9           905      2184   9676768+  83  Linux
```

This particular disk is configured to house seven Linux filesystems (each with a corresponding partition listed as "Linux") as well as a swap partition (listed as "Linux swap").

Removing all partitions with fdisk

First remove all existing partitions from the disk. Type `d` to delete a partition. For instance, to delete an existing `/dev/sda1`:

Command (m for help): d

```
Partition number (1-4): 1
```

The partition has now been scheduled for deletion. It will no longer show up when printing the list of partitions (`p`, but it will not be erased until the changes have been saved. This allows users to abort the operation if a mistake was made - in that case, type `q` immediately and hit enter and the partition will not be deleted.

Repeatedly type `p` to print out a partition listing and then type `d` and the number of the partition to delete it. Eventually, the partition table will be empty:

Command (m for help): p

```
Disk /dev/sda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

Device Boot    Start       End    Blocks   Id  System
```

Now that the in-memory partition table is empty, we're ready to create the partitions.

Creating the BIOS boot partition

First create a very small BIOS boot partition. Type `n` to create a new partition, then `p` to select a primary partition, followed by `1` to select the first primary partition. When prompted for the first sector, make sure it starts from 2048 (which is needed for the boot loader) and hit enter. When prompted for the last sector, type +2M to create a partition 2 Mbyte in size:

> **Note**
> The start from sector 2048 is a fail-safe in case the boot loader does not detect this partition as being available for its use.

Command (m for help): n

```
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (64-10486533532, default 64): 2048
Last sector, +sectors +size{M,K,G} (4096-10486533532, default 10486533532): +2M
```

Mark the partition for EFI purposes:

Command (m for help): t

```
Selected partition 1
Hex code (type L to list codes): 4
Changed system type of partition 1 to 4 (BIOS boot)
```

> **Note**
> Using EFI with MBR partition layout is discouraged. If an EFI capable system is used, please use GPT layout.

Creating the boot partition

Now create a small boot partition. Type `n` to create a new partition, then `p` to select a primary partition, followed by `2` to select the second primary partition. When prompted for the first sector, accept the default by hitting `Enter`. When prompted for the last sector, type +128M to create a partition 128 Mbyte in size:

Command (m for help): n

```
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First sector (5198-10486533532, default 5198): (Hit enter)
Last sector, +sectors +size{M,K,G} (4096-10486533532, default 10486533532): +128M
```

Now, when pressing `p`, the following partition printout is displayed:

Command (m for help): p

```
Disk /dev/sda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sda1             1         3      5198+  ef  EFI (FAT-12/16/32)
/dev/sda2             3        14    105808+  83  Linux
```

Type `a` to toggle the bootable flag on a partition and select `2`. After pressing `p` again, notice that an * is placed in the "Boot" column.

Creating the swap partition

To create the swap partition, type `n` to create a new partition, then `p` to tell fdisk to create a primary partition. Then type `3` to create the third primary partition, `/dev/sda3`. When prompted for the first sector, hit `Enter`. When prompted for the last sector, type +512M (or any other size needed for the swap space) to create a partition 512MB in size.

After all this is done, type `t` to set the partition type, `3` to select the partition just created and then type in *82* to set the partition type to "Linux Swap".

Creating the root partition

Finally, to create the root partition, type `n` to create a new partition, then `p` to tell `fdisk` to create a primary partition. Then type `4` to create the fourth primary partition, `/dev/sda4`. When prompted for the first sector, hit `Enter`. When prompted for the last sector, hit `Enter` to create a partition that takes up the rest of the remaining space on the disk. After completing these steps, typing `p` should display a partition table that looks similar to this:

Command (m for help): p

```
Disk /dev/sda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sda1             1         3      5198+  ef  EFI (FAT-12/16/32)
/dev/sda2   *         3        14    105808+  83  Linux
/dev/sda3            15        81    506520   82  Linux swap
/dev/sda4            82      3876  28690200   83  Linux
```

Saving the partition layout

To save the partition layout and exit `fdisk`, type `w`.

Command (m for help): w

With the partitions created, it is now time to put filesystems on them.

Creating file systems

Introduction

Now that the partitions are created, it is time to place a filesystem on them. In the next section the various file systems that Linux supports are described. Readers that already know which filesystem to use can continue with Applying a filesystem to a partition. The others should read on to learn about the available filesystems...

Filesystems

Several filesystems are available. Some of them are found stable on the amd64 architecture - it is advised to read up on the filesystems and their support state before selecting a more experimental one for important partitions.

ext2
This is the tried and true Linux filesystem but doesn't have metadata journaling, which means that routine ext2 filesystem checks at startup time can be quite time-consuming. There is now quite a selection of newer-generation journaled filesystems that can be checked for consistency very quickly and are thus generally preferred over their non-journaled counterparts. Journaled filesystems prevent long delays when the system is booted and the filesystem happens to be in an inconsistent state.

ext3
The journaled version of the ext2 filesystem, providing metadata journaling for fast recovery in addition to other enhanced journaling modes like full data and ordered data journaling. It uses an HTree index that enables high performance in almost all situations. In short, ext3 is a very good and reliable filesystem.

ext4
Initially created as a fork of ext3, ext4 brings new features, performance improvements and removal of size limits with moderate changes to the on-disk format. It can span volumes up to 1 EB and with maximum file size of 16 TB. Instead of the classic ext2/3 bitmap block allocation ext4 uses extents, which improve large file performance and reduce fragmentation. Ext4 also provides more sophisticated block allocation algorithms (delayed allocation and multiblock allocation) giving the filesystem driver more ways to optimize the layout of data on the disk. Ext4 is the recommended all-purpose all-platform filesystem.

JFS
IBM's high-performance journaling filesystem. JFS is a light, fast and reliable B+tree-based filesystem with good performance in various conditions.

ReiserFS
A B+tree-based journaled filesystem that has good overall performance, especially when dealing with many tiny files at the cost of more CPU cycles. ReiserFS appears to be less maintained than other filesystems.

XFS
A filesystem with metadata journaling which comes with a robust feature-set and is optimized for scalability. XFS seems to be less forgiving to various hardware problems.

vfat
Also known as FAT32, is supported by Linux but does not support any permission settings. It is mostly used for interoperability with other operating systems (mainly Microsoft Windows) but is also a necessity for some system firmware (like UEFI).

When using ext2, ext3 or ext4 on a small partition (less than 8GB), then the file system must be created with the proper options to reserve enough inodes. The `mke2fs` application uses the "bytes-per-inode" setting to calculate how many inodes a file system should have. On smaller partitions, it is advised to increase the calculated number of inodes.

On ext2, this can be done using the following command:

```
root # mke2fs -T small /dev/<device>
```

On ext3 and ext4, add the `-j` option to enable journaling:

```
root # mke2fs -j -T small /dev/<device>
```

This will generally quadruple the number of inodes for a given file system as its "bytes-per-inode" reduces from one every 16kB to one every 4kB. This can be tuned even further by providing the ratio:

```
root # mke2fs -i <ratio> /dev/<device>
```

Applying a filesystem to a partition

To create a filesystem on a partition or volume, there are tools available for each possible filesystem:

Filesystem	Creation Command
ext2	`mkfs.ext2`
ext3	`mkfs.ext3`
ext4	`mkfs.ext4`
reiserfs	`mkreiserfs`
xfs	`mkfs.xfs`
jfs	`mkfs.jfs`
vfat	`mkfs.vfat`

For instance, to have the boot partition (`/dev/sda2`) in ext2 and the root partition (`/dev/sda4`) in ext4 as used in the example partition structure, the following commands would be used:

```
root # mkfs.ext2 /dev/sda2
root # mkfs.ext4 /dev/sda4
```

Now create the filesystems on the newly created partitions (or logical volumes).

Activating the Swap Partition

`mkswap` is the command that is used to initialize swap partitions:

```
root # mkswap /dev/sda3
```

To activate the swap partition, use `swapon` :

```
root # swapon /dev/sda3
```

Create and activate the swap with the commands mentioned above.

Mounting

Now that the partitions are initialized and are housing a filesystem, it is time to mount those partitions. Use the `mount` command, but don't forget to create the necessary mount directories for every partition created. As an example we mount the root and boot partition:

```
root # mount /dev/sda4 /mnt/gentoo
root # mkdir /mnt/gentoo/boot
root # mount /dev/sda2 /mnt/gentoo/boot
```

> **Note**
>
> If `/tmp/` needs to reside on a separate partition, be sure to change its permissions after mounting:
>
> ```
> root # chmod 1777 /mnt/gentoo/tmp
> ```
>
> This also holds for `/var/tmp`.

Later in the instructions the proc filesystem (a virtual interface with the kernel) as well as other kernel psuedo-filesystems will be mounted. But first we install the Gentoo installation files (/wiki/Handbook:AMD64/Installation/Stage).

Installing a stage tarball

Setting the date and time

Before installing Gentoo, make sure that the date and time are set correctly. A mis-configured clock may lead to strange results in the future!

To verify the current date/time, run date:

root # `date`

```
Fri Mar 29 16:21:18 UTC 2005
```

If the date/time displayed is wrong, update it using the `date MMDDhhmmYYYY` syntax (Month, Day, hour, minute and Year). At this stage, it is recommended to use UTC time. Later on during the installation, the timezone will be defined.

For instance, to set the date to March 29th, 16:21 in the year 2014:

root # `date 032916212014`

Downloading the stage tarball

Go to the Gentoo mountpoint where the root file system is mounted (most likely `/mnt/gentoo`):

root # `cd /mnt/gentoo`

Depending on the installation medium, there are a couple of tools available to download a stage. One of these tools is `links`, a non-graphical, menu-driven browser. To download a stage, surf to the Gentoo mirror list like so:

root # `links (https://www.gentoo.org/downloads/mirrors/)`

To use an HTTP proxy with `links`, pass on the URL with the *-http-proxy* option:

root # `links -http-proxy proxy.server.com:8080 (https://www.gentoo.org/downloads/mirrors/)`

Next to `links` there is also the `lynx` browser. Like `links` it is a non-graphical browser but it is not menu-driven.

root # `lynx (https://www.gentoo.org/downloads/mirrors/)`

If a proxy needs to be defined, export the `http_proxy` and/or `ftp_proxy` variables:

root # `export http_proxy=" (http://proxy.server.com:port)"`

root # `export ftp_proxy=" (http://proxy.server.com:port)"`

On the mirror list, select a mirror close by. Usually HTTP mirrors suffice, but other protocols are available as well. Move to the `releases/amd64/autobuilds/` directory. There all available stage files are displayed (they might be stored within subdirectories named after the individual sub-architectures). Select one and press `D` to download.

Like with the minimal installation CDs, additional downloads are available:

- A `.CONTENTS` file that contains a list of all files inside the stage tarball
- A `.DIGESTS` file that contains checksums of the stage file, in different algorithms
- A `.DIGESTS.asc` file that, like the `.DIGESTS` file, contains checksums of the stage file in different algorithms, but is also cryptographically signed to ensure it is provided by the Gentoo project

When finished, press `Q` to quit the browser.

After downloading the stage file, it is possible to verify the integrity of the downloaded stage tarball. Use `openssl` and compare the output with the checksums provided by the `.DIGESTS` or `.DIGESTS.asc` file.

For instance, to validate the SHA512 checksum:

```
root # openssl dgst -r -sha512 stage3-amd64-<release>.tar.bz2
```

Another way is to use the `sha512sum` command:

```
root # sha512sum stage3-amd64-<release>.tar.bz2
```

To validate the Whirlpool checksum:

```
root # openssl dgst -r -whirlpool stage3-amd64-<release>.tar.bz2
```

Compare the output of these commands with the value registered in the `.DIGESTS(.asc)` files. The values need to match, otherwise the downloaded file might be corrupt (or the digests file is).

Just like with the ISO file, it is also possible to verify the cryptographic signature of the `.DIGESTS.asc` file using `gpg` to make sure the checksums have not been tampered with:

```
root # gpg --verify stage3-amd64-<release>.tar.bz2.DIGESTS.asc
```

Unpacking the stage tarball

Now unpack the downloaded stage onto the system. We use `tar` to proceed:

```
root # tar xvjpf stage3-*.tar.bz2 --xattrs
```

Make sure that the same options (`xvjpf` and `--xattrs`) are used. The x stands for Extract, the v for Verbose to see what happens during the extraction process (optional), the j for Decompress with bzip2, the p for Preserve permissions and the f to denote that we want to extract a File, not standard input. Finally, the `--xattrs` is to include the extended attributes stored in the archive as well.

Now that the stage is installed, continue with Configuring the compile options.

Configuring compile options

Introduction

To optimize Gentoo, it is possible to set a couple of variables which impact Portage behavior. All those variables can be set as environment variables (using `export`) but that isn't permanent. To keep the settings, Portage reads in the `/etc/portage/make.conf` file, a configuration file for Portage.

> **Note**
> A commented listing of all possible variables can be found in `/mnt/gentoo/usr/share/portage/config/make.conf.example`. For a successful Gentoo installation only the variables that are mentioned below need to be set.

Fire up an editor (in this guide we use `nano`) to alter the optimization variables we will discuss hereafter.

```
root # nano -w /mnt/gentoo/etc/portage/make.conf
```

From the `make.conf.example` file it is obvious how the file should be structured: commented lines start with "#", other lines define variables using the VARIABLE="content" syntax. Several of those variables are discussed next.

CFLAGS and CXXFLAGS

The CFLAGS and CXXFLAGS variables define the optimization flags for the gcc C and C++ compiler respectively. Although those are defined generally here, for maximum performance one would need to optimize these flags for each program separately. The reason for this is because every program is different. However, this is not manageable, hence the definition of these flags in the `make.conf` file.

In `make.conf` one should define the optimization flags that will make the system the most responsive generally. Don't place experimental settings in this variable; too much optimization can make programs behave bad (crash, or even worse, malfunction).

We will not explain all possible optimization options. To understand them all, read the GNU Online Manual(s) (http://gcc.gnu.org/onlinedocs/) or the gcc info page (`info gcc` -- only works on a working Linux system). The `make.conf.example` file itself also contains lots of examples and information; don't forget to read it too.

A first setting is the `-march=` or `-mtune=` flag, which specifies the name of the target architecture. Possible options are described in the `make.conf.example` file (as comments). A commonly used value is *native* as that tells the compiler to select the target architecture of the current system (the one users are installing Gentoo on).

A second one is the `-O` flag (that is a capital O, not a zero), which specifies the gcc optimization class flag. Possible classes are s (for size-optimized), 0 (zero - for no optimizations), 1, 2 or even 3 for more speed-optimization flags (every class has the same flags as the one before, plus some extras). `-O2` is the recommended default. `-O3` is known to cause problems when used system-wide, so we recommend to stick to `-O2` .

Another popular optimization flag is `-pipe` (use pipes rather than temporary files for communication between the various stages of compilation). It has no impact on the generated code, but uses more memory. On systems with low memory, gcc might get killed. In that case, do not use this flag.

Using `-fomit-frame-pointer` (which doesn't keep the frame pointer in a register for functions that don't need one) might have serious repercussions on the debugging of applications.

When the CFLAGS and CXXFLAGS variables are defined, combine the several optimization flags in one string. The default values contained in the stage3 archive that is unpacked should be good enough. The following one is just an example:

CODE **Example CFLAGS and CXXFLAGS variables**

```
CFLAGS="-march=native -O2 -pipe"
# Use the same settings for both variables
CXXFLAGS="${CFLAGS}"
```

Note
The GCC optimization guide (/wiki/GCC_optimization) has more information on how the various compilation options can affect a system.

MAKEOPTS

The `MAKEOPTS` variable defines how many parallel compilations should occur when installing a package. A good choice is the number of CPUs (or CPU cores) in the system plus one, but this guideline isn't always perfect.

CODE **Example MAKEOPTS declaration in make.conf**

```
MAKEOPTS="-j2"
```

Ready, set, go

Update the /mnt/gentoo/etc/portage/make.conf file to match personal preference and save (nano users would hit Ctrl+X).

Then continue with Installing the Gentoo base system (/wiki/Handbook:AMD64/Installation/Base).

Chrooting

Optional: Selecting mirrors

Warning
app-portage/mirrorselect (http://packages.gentoo.org/package/app-portage/mirrorselect) has not been updated to handle modifying the target chroots repos.conf/gentoo.conf file yet. Also, the SYNC variable in make.conf is deprecated and no longer used by portage. This section needs to be updated, please skip for the time being...

In order to download source code quickly it is recommended to select a fast mirror. Portage will look in the make.conf file for the GENTOO_MIRRORS variable and use the mirrors listed therein. It is possible to surf to the Gentoo mirror list and search for a mirror (or mirrors) that is close to the system's physical location (as those are most frequently the fastest ones). However, we provide a nice tool called mirrorselect which provides users with a nice interface to select the mirrors needed. Just navigate to the mirrors of choice and press Spacebar to select one or more mirrors.

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

A second important setting is the SYNC setting in make.conf. This variable contains the rsync server to use when updating the portage tree (the collection of ebuilds and related files containing all the information portage needs to download and install software). Again, it is possible to manually enter a SYNC server, but mirrorselect can ease that operation considerably:

```
root # mirrorselect -i -r -o >> /mnt/gentoo/etc/portage/make.conf
```

After running mirrorselect it is advisable to double-check the settings in /mnt/gentoo/etc/portage/make.conf !

Note
When manually selecting a SYNC server in make.conf, check out the community mirrors list for the mirrors closest to you. It is recommended to choose a rotation link, such as rsync.us.gentoo.org, rather than choosing a single mirror. This helps spread out the load and provides a failsafe in case a specific mirror is offline.

Copy DNS info

One thing still remains to be done before entering the new environment and that is copying over the DNS information in /etc/resolv.conf. This needs to be done to ensure that networking still works even after entering the new environment. /etc/resolv.conf contains the name servers for the network.

To copy this information, it is recommended to use the `-L` option to the `cp` command. This ensures that, if `/etc/resolv.conf` is a symbolic link, that the link's target file is copied instead of the symbolic link itself. Otherwise in the new environment the symbolic link would point to a non-existing file (as the link's target is most likely not available inside the new environment).

```
root # cp -L /etc/resolv.conf /mnt/gentoo/etc/
```

Mounting the necessary filesystems

In a few moments, the Linux root will be changed towards the new location. To make sure that the new environment works properly, certain filesystems need to be made available there as well.

The filesystems that need to be made available are:

- `/proc/` which is a pseudo-filesystem (it looks like regular files, but is actually generated on-the-fly) from which the Linux kernel exposes information to the environment
- `/sys/` which is a pseudo-filesystem, like `/proc/` which it was once meant to replace, and is more structured than `/proc/`
- `/dev/` is a regular file system, partially managed by the Linux device manager (usually `udev`), which contains all device files

The `/proc/` location will be mounted on `/mnt/gentoo/proc/` whereas the other two are bind-mounted. The latter means that, for instance, `/mnt/gentoo/sys/` will actually *be* `/sys/` (it is just a second entry point to the same filesystem) whereas `/mnt/gentoo/proc/` is a new mount (instance so to speak) of the filesystem.

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

Note
The `--make-rslave` operations are needed for systemd support later in the installation.

Warning
When using non-Gentoo installation media, this might not be sufficient. Some distributions make `/dev/shm` a symbolic link to `/run/shm/` which, after the chroot, becomes invalid. Making `/dev/shm/` a proper tmpfs mount up front can fix this:

```
root # rm /dev/shm && mkdir /dev/shm
root # mount -t tmpfs -o nosuid,nodev,noexec shm /dev/shm
```

Also ensure that mode 1777 is set

```
root #  chmod 1777 /dev/shm
```

Entering the new environment

Now that all partitions are initialized and the base environment installed, it is time to enter the new installation environment by chrooting into it. This means that the session will change its root (most top-level location that can be accessed) from the current installation environment (installation CD or other installation medium) to the installation system (namely the initialized partitions). Hence the name, *change root* or *chroot*.

This chrooting is done in three steps.

1. The root location is changed from `/` (on the installation medium) to `/mnt/gentoo/` (on the partitions) using

chroot

2. Some settings (those in `/etc/profile`) are reloaded in memory using the `source` command
3. The primary prompt is changed to help us remember that this session is inside a chroot environment.

```
root # chroot /mnt/gentoo /bin/bash
root # source /etc/profile
root # export PS1="(chroot) $PS1"
```

From this point, all actions performed are immediately on the new Gentoo Linux environment. Of course it is far from finished, which is why the installation still has some sections left :-)

Configuring portage

Installing a portage snapshot

Next step is to install a portage snapshot, a collection of files that inform portage what software titles are available to install, which profiles the administrator can select, etc.

The use of `emerge-webrsync` is recommended. This will fetch the latest portage snapshot (which Gentoo releases on a daily basis) from one of Gentoo's mirrors and install it onto the system.

```
root # emerge-webrsync
```

> **Note**
> During this operation, `emerge-webrsync` might complain about a missing `/usr/portage/` location. This is to be expected and nothing to worry about - the tool will create the location.

From this point onward, portage might mention that certain updates are recommended to be executed. This is because certain system packages installed through the stage3 file might have newer versions available, and portage is now aware of this because a new portage snapshot is installed. This can be safely ignored for now; the updates can be triggered after the Gentoo installation has finished.

Optional: Updating the portage tree

It is possible to update the portage tree to the latest version. The previous `emerge-webrsync` command will have installed a very recent portage snapshot (usually recent up to 24h) so this step is definitely optional.

Suppose there is a need for the last package updates (up to 1 hour), then use `emerge --sync` . This command will use the rsync protocol to update the portage tree (which was fetched earlier on through `emerge-webrsync`) to the latest state.

```
root # emerge --sync
```

On slow terminals, like some framebuffers or serial consoles, it is recommended to use the `--quiet` option to speed up the process:

```
root # emerge --sync --quiet
```

Reading news items

When a portage tree is synchronized to the system, portage might warn the user with the following:

CODE **Portage informing the user about news items**

```
* IMPORTANT: 2 news items need reading for repository 'gentoo'.
* Use eselect news to read news items.
```

Portage news items were created to provide a communication medium to push critical messages to users via the rsync tree. To manage them, use `eselect news` . The `eselect` application is a Gentoo application that allows for a common management interface towards system changes and operations. In this case, `eselect` is asked to use its `news` module.

For the `news` module, three operations are most used:

- With `list` an overview of the available news items is displayed
- With `read` the news items can be read
- With `purge` news items can be removed once they have been read and will not be reread anymore

```
root # eselect news list
```

```
root # eselect news read
```

More information about the newsreader is available through its manual page:

```
root # man news.eselect
```

Choosing the right profile

A *profile* is a building block for any Gentoo system. Not only does it specify default values for `USE` , `CFLAGS` and other important variables, it also locks the system to a certain range of package versions. This is all maintained by the Gentoo developers.

You can see what profile the system is currently using with `eselect` , now using the `profile` module:

```
root # eselect profile list
```

```
Available profile symlink targets:
  [1]   default/linux/amd64/13.0 *
  [2]   default/linux/amd64/13.0/desktop
  [3]   default/linux/amd64/13.0/desktop/gnome
  [4]   default/linux/amd64/13.0/desktop/kde
```

Note
The output of the command is just an example and evolves over time.

As can be seen, there are also desktop subprofiles available for some architectures.

After viewing the available profiles for the amd64 architecture, users can select a different profile to use:

```
root # eselect profile set 2
```

In order to select a pure 64-bit environment, with no 32-bit applications or libraries, use a non-multilib profile:

```
root # eselect profile list
```

```
Available profile symlink targets:
  [1]   default/linux/amd64/13.0 *
  [2]   default/linux/amd64/13.0/desktop
  [3]   default/linux/amd64/13.0/desktop/gnome
  [4]   default/linux/amd64/13.0/desktop/kde
  [5]   default/linux/amd64/13.0/no-multilib
```

Next select the *no-multilib* profile:

```
root # eselect profile set 5
```

root # eselect profile list

```
Available profile symlink targets:
  [1]   default/linux/amd64/13.0
  [2]   default/linux/amd64/13.0/desktop
  [3]   default/linux/amd64/13.0/desktop/gnome
  [4]   default/linux/amd64/13.0/desktop/kde
  [5]   default/linux/amd64/13.0/no-multilib *
```

> **Note**
> The developer subprofile is specifically for Gentoo Linux development and is not meant to be used by regular users.

Configuring the USE variable

`USE` is one of the most powerful variables Gentoo provides to its users. Several programs can be compiled with or without optional support for certain items. For instance, some programs can be compiled with gtk-support, or with qt-support. Others can be compiled with or without SSL support. Some programs can even be compiled with framebuffer support (svgalib) instead of X11 support (X-server).

Most distributions compile their packages with support for as much as possible, increasing the size of the programs and startup time, not to mention an enormous amount of dependencies. With Gentoo users can define what options a package should be compiled with. This is where `USE` comes into play.

In the `USE` variable users define keywords which are mapped onto compile-options. For instance, `ssl` will compile ssl-support in the programs that support it. `-X` will remove X-server support (note the minus sign in front). `gnome gtk -kde -qt4` will compile programs with gnome (and gtk) support, and not with kde (and qt) support, making the system fully tweaked for GNOME (if the architecture supports it).

The default USE settings are placed in the `make.defaults` files of the Gentoo profile used by the system. Gentoo uses a (complex) inheritance system for its profiles, which we will not dive into at this stage. The easiest way to check the currently active `USE` settings is to run `emerge --info` and select the line that starts with *USE*:

root # emerge --info | grep ^USE

```
USE="X acl alsa amd64 berkdb bindist bzip2 cli cracklib crypt cxx dri ..."
```

> **Note**
> The above example is truncated, the actual list of USE settings is much, much larger.

A full description on the available USE flags can be found on the system in `/usr/portage/profiles/use.desc`.

root # less /usr/portage/profiles/use.desc

Inside the `less` command, scrolling can be done using the ↑ and ↓ keys, and exited by pressing q.

As an example we show a USE setting for a KDE-based system with DVD, ALSA and CD Recording support:

root # nano -w /etc/portage/make.conf

FILE `/etc/portage/make.conf` **Enabling USE for a KDE-based system with DVD, ALSA and cd recording support**

```
USE="-gtk -gnome qt4 kde dvd alsa cdr"
```

When `USE` is defined in `/etc/portage/make.conf` it is *added* (or *removed* if the USE flag starts with the `-` sign) from that default list. Users who want to ignore any default USE settings and manage it completely themselves should start the `USE` definition in `make.conf` with `-*` :

FILE `/etc/portage/make.conf` **Ignoring default USE flags**

```
USE="-* X acl alsa ..."
```

Optional: Using systemd

The remainder of the Gentoo Handbook focuses on OpenRC as the default init support system. If systemd is wanted instead, or are planning to use Gnome 3.8 and later (which requires systemd), please consult the systemd (/wiki/Systemd) page as it elaborates on the different configuration settings and methods.

The Gentoo handbook can then be followed with that page in mind.

Timezone

Select the timezone for the system. Look for the available timezones in `/usr/share/zoneinfo/`, then write it in the `/etc/timezone` file.

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

Suppose the timezone of choice is *Europe/Brussels*:

```
root # echo "Europe/Brussels" > /etc/timezone
```

Please avoid the `/usr/share/zoneinfo/Etc/GMT*` timezones as their names do not indicate the expected zones. For instance, `GMT-8` is in fact `GMT+8`.

Next, reconfigure the sys-libs/timezone-data (http://packages.gentoo.org/package/sys-libs/timezone-data) package, which will update the `/etc/localtime` file for us, based on the `/etc/timezone` entry. The `/etc/localtime` file is used by the system C library to know the timezone the system is in.

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

Configure locales

Most users will want to use only one or two locales on their system.

Locales specify not only the language that the system should use to interact with the system, but also what the rules are for sorting strings, displaying dates and times, etc.

The locales that a system should support should be mentioned in `/etc/locale.gen`.

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

The following locales are an example to get both English (United States) and German (Germany) with the accompanying character formats (like UTF-8).

FILE `/etc/locale.gen` **Enabling US and DE locales with the appropriate character formats**

```
en_US ISO-8859-1
en_US.UTF-8 UTF-8
de_DE ISO-8859-1
de_DE@euro ISO-8859-15
```

Warning
We strongly suggest to use at least one UTF-8 locale because some applications may require it.

The next step is to run `locale-gen`. It will generate all the locales specified in the `/etc/locale.gen` file.

```
root # locale-gen
```

To verify that the selected locales are now available, run `locale -a`.

Once done, it is now time to set the system-wide locale settings. Again we use `eselect` for this, now with the `locale` module.

With `eselect locale list`, the available targets are displayed:

```
root # eselect locale list
```

```
Available targets for the LANG variable:
  [1] C
  [2] POSIX
  [3] en_US
  [4] en_US.iso88591
  [5] en_US.utf8
  [6] de_DE
  [7] de_DE.iso88591
  [8] de_DE.iso885915
  [9] de_DE.utf8
  [ ] (free form)
```

With `eselect locale set VALUE` the correct locale can be set:

```
root # eselect locale set 9
```

Manually, this can still be accomplished through the `/etc/env.d/02locale` file:

FILE **`/etc/env.d/02locale`** **Manually setting system locale definitions**

```
LANG="de_DE.UTF-8"
LC_COLLATE="C"
```

Make sure a locale is set, as the system would otherwise display warnings and errors during kernel builds and other software deployments later in the installation.

Now reload the environment:

```
root # env-update && source /etc/profile
```

We made a full Localization guide (/wiki/Localization/HOWTO) to help the user guide through this process. Another interesting article is the UTF-8 (/wiki/UTF-8) guide for very specific informations to enable UTF-8 on the system.

Installing the sources

The core around which all distributions are built is the Linux kernel. It is the layer between the user programs and the system hardware. Gentoo provides its users several possible kernel sources. A full listing with description is available at the Kernel overview page (/wiki/Kernel/Overview).

For amd64-based systems Gentoo recommends the sys-kernel/gentoo-sources (http://packages.gentoo.org/package/sys-kernel/gentoo-sources) package.

Choose an appropriate kernel source and install it using `emerge` :

```
root # emerge --ask sys-kernel/gentoo-sources
```

This will install the Linux kernel sources in `/usr/src/` in which a symbolic link called `linux` will be pointing to the installed kernel source:

```
root # ls -l /usr/src/linux
```

```
lrwxrwxrwx    1 root   root    12 Oct 13 11:04 /usr/src/linux -> linux-3.16.5-gentoo
```

Now it is time to configure and compile the kernel sources. There are two approaches for this:

1. either the kernel is manually configured and build, or
2. a tool called `genkernel` is used to automatically build and install the Linux kernel

We explain the manual configuration as the default choice here as it is the best way to optimize an environment.

Default: Manual configuration

Introduction

Manually configuring a kernel is often seen as the most difficult procedure a Linux user ever has to perform. Nothing is less true -- after configuring a couple of kernels no-one even remembers that it was difficult ;)

However, one thing is true: it is vital to know the system when a kernel is configured manually. Most information can be gathered by emerging sys-apps/pciutils (http://packages.gentoo.org/package/sys-apps/pciutils) which contains the `lspci` command:

```
root # emerge --ask sys-apps/pciutils
```

Note
Inside the chroot, it is safe to ignore any pcilib warnings (like *pcilib: cannot open /sys/bus/pci/devices*) that `lspci` might throw out.

Another source of system information is to run `lsmod` to see what kernel modules the installation CD uses as it might provide a nice hint on what to enable.

Now go to the kernel source directory and execute `make menuconfig` . This will fire up menu-driven configuration screen.

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

The Linux kernel configuration has many, many sections. Let's first list some options that must be activated (otherwise Gentoo will not function, or not function properly without additional tweaks). We also have a Gentoo kernel configuration guide (/wiki/Kernel/Gentoo_Kernel_Configuration_Guide) on the Gentoo wiki that might help out further.

Activating required options

Make sure that every driver that is vital to the booting of the system (such as SCSI controller, ...) is compiled in the kernel and not as a module, otherwise the system will not be able to boot completely.

Next select the exact processor type. It is also recommended to enable MCE features (if available) so that users are able to be notified of any hardware problems. On some architectures (such as x86_64), these errors are not printed to dmesg, but to `/dev/mcelog`. This requires the app-admin/mcelog (http://packages.gentoo.org/package/app-admin/mcelog) package.

Also select *Maintain a devtmpfs file system to mount at /dev* so that critical device files are already available early in the boot process.

KERNEL **Enabling devtmpfs support**

```
Device Drivers --->
  Generic Driver Options --->
    [*] Maintain a devtmpfs filesystem to mount at /dev
    [ ]   Automount devtmpfs at /dev, after the kernel mounted the rootfs
```

Now go to File Systems and select support for the filesystems you use. Don't compile the file system that is used for the root filesystem as module, otherwise the Gentoo system will not be able to mount the partition. Also select *Virtual memory* and */proc file system*.

KERNEL **Selecting necessary file systems**

```
File systems --->
(Select one or more of the following options as needed by your system)
  <*> Second extended fs support
  <*> Ext3 journalling file system support
  <*> The Extended 4 (ext4) filesystem
  <*> Reiserfs support
  <*> JFS filesystem support
  <*> XFS filesystem support
  ...
  Pseudo Filesystems --->
    [*] /proc file system support
    [*] Virtual memory file system support (former shm fs)
```

If PPPoE is used to connect to the Internet, or a dial-up modem is used, then enable the following options:

KERNEL **Selecting PPPoE necessary drivers**

```
Device Drivers --->
  Network device support --->
    <*> PPP (point-to-point protocol) support
    <*>   PPP support for async serial ports
    <*>   PPP support for sync tty ports
```

The two compression options won't harm but are not definitely needed, neither does the PPP over Ethernet option, that might only be used by ppp when configured to do kernel mode PPPoE.

Don't forget to include support in the kernel for the network (Ethernet or wireless) cards.

Most systems also have multiple cores at their disposal, so it is important to activate *Symmetric multi-processing support*:

KERNEL **Activating SMP support**

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

Note
In multi-core systems, each core counts as one processor.

If USB input devices (like keyboard or mouse) are used don't forget to enable those as well:

KERNEL **Activating USB Support for input devices**

```
Device Drivers --->
  [*] HID Devices  --->
    <*>   USB Human Interface Device (full HID) support
```

Architecture specific kernel configuration

Make sure to select IA32 Emulation if 32-bit programs should be supported (multilib). Gentoo will install a multilib system (mixed 32-bit/64-bit computing) by default, so unless a `no-multilib` profile is used, this option is required.

KERNEL **Selecting processor types and features**

```
Processor type and features  --->
   [ ] Machine Check / overheating reporting
   [ ]   Intel MCE Features
   [ ]   AMD MCE Features
  Processor family (AMD-Opteron/Athlon64)  --->
    ( ) Opteron/Athlon64/Hammer/K8
    ( ) Intel P4 / older Netburst based Xeon
    ( ) Core 2/newer Xeon
    ( ) Intel Atom
    ( ) Generic-x86-64
Executable file formats / Emulations  --->
   [*] IA32 Emulation
```

Enable GPT partition label support if that was used previously when partitioning the disk:

KERNEL **Enable support for GPT**

```
-*- Enable the block layer --->
    ...
    Partition Types --->
    [*] Advanced partition selection
      ...
      [*] EFI GUID Partition support
```

Enable EFI stub support and EFI variables in the Linux kernel if UEFI is used to boot the system:

KERNEL **Enable support for UEFI**

```
Processor type and features  --->
    [*] EFI runtime service support
    [*]   EFI stub support

Firmware Drivers  --->
    <*> EFI Variable Support via sysfs
```

Compiling and installing

With the configuration now done, it is time to compile and install the kernel. Exit the configuration and start the compilation process:

```
root # make && make modules_install
```

> **Note**
> It is possible to enable parallel builds using `make -jX` with X being the number of parallel tasks that the build process is allowed to launch. This is similar to the instructions about `/etc/portage/make.conf` earlier, with the `MAKEOPTS` variable.

When the kernel has finished compiling, copy the kernel image to `/boot/`. This is handled by the `make install` command:

```
root # make install
```

This will copy the kernel image into `/boot/` together with the `System.map` file and the kernel configuration file.

With UEFI systems, create the `/boot/efi/boot/` location, and then copy the kernel into this location, calling it `bootx64.efi`:

```
root # mkdir -p /boot/efi/boot
root # cp /boot/vmlinuz-* /boot/efi/boot/bootx64.efi
```

Optional: Building an initramfs

In certain cases it is necessary to build an initramfs - an initial ram-based file system. The most common reason is when important file system locations (like `/usr/` or `/var/`) are on separate partitions. With an initramfs, these partitions can be mounted using the tools available inside the initramfs.

Without an initramfs, there is a huge risk that the system will not boot up properly as the tools that are responsible for mounting the file systems need information that resides on those file systems. An initramfs will pull in the necessary files into an archive which is used right after the kernel boots, but before the control is handed over to the init tool. Scripts on the initramfs will then make sure that the partitions are properly mounted before the system continues booting.

To install an initramfs, install sys-kernel/genkernel (http://packages.gentoo.org/package/sys-kernel/genkernel) first, then have it generate an initramfs:

```
root # emerge genkernel
root # genkernel --install initramfs
```

In order to enable specific support in the initramfs, such as lvm or raid, add in the appropriate options to `genkernel`. See `genkernel --help` for more information. In the next example we enables support for LVM and software raid (mdadm):

```
root # genkernel --lvm --mdadm --install initramfs
```

The initramfs will be stored in `/boot/`. The resulting file can be found by simply listing the files starting with *initramfs*:

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

Now continue with Kernel modules.

Alternative: Using genkernel

If a manual configuration looks too daunting, then using `genkernel` is recommended. It will configure and build the kernel automatically.

`genkernel` works by configuring a kernel nearly identically to the way the installation CD kernel is configured. This means that when `genkernel` is used to build the kernel, the system will generally detect all hardware at boot-time, just like the installation CD does. Because `genkernel` doesn't require any manual kernel configuration, it is an ideal solution for those users who may not be comfortable compiling their own kernels.

Now, let's see how to use genkernel. First, emerge the sys-kernel/genkernel (http://packages.gentoo.org/package/sys-kernel/genkernel) ebuild:

```
root # emerge --ask sys-kernel/genkernel
```

Next, edit the `/etc/fstab` file so that the line containing `/boot/` as second field has the first field pointing to the right device. If the partitioning example from the handbook is followed, then this device is most likely /dev/sda2 with the ext2 file system. This would make the entry in the file look like so:

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

FILE `/etc/fstab` **Configuring the /boot mountpoint**

```
/dev/sda2       /boot   ext2    defaults        0 2
```

Note
Further in the Gentoo installation, `/etc/fstab` will be configured again. The `/boot` setting is needed right now as the `genkernel` application reads in this configuration.

Now, compile the kernel sources by running `genkernel all`. Be aware though, as `genkernel` compiles a kernel that supports almost all hardware, this compilation will take quite a while to finish!

Note
If the boot partition doesn't use ext2 or ext3 as filesystem it *might* be necessary to manually configure the kernel using `genkernel --menuconfig all` and add support for this particular filesystem in the kernel (i.e. not as a module). Users of LVM2 will probably want to add `--lvm` as an argument as well.

```
root # genkernel all
```

Once genkernel completes, a kernel, full set of modules and initial ram disk (initramfs) will be created. We will use the kernel and initrd when configuring a boot loader later in this document. Write down the names of the kernel and initrd as this information is used when the boot loader configuration file is edited. The initrd will be started immediately after booting to perform hardware autodetection (just like on the installation CD) before the "real" system starts up.

```
root # ls /boot/kernel* /boot/initramfs*
```

Kernel modules

Configuring the modules

List the modules that need to be loaded automatically in `/etc/conf.d/modules`. Extra options can be added to the modules too if necessary.

To view all available modules, run the following `find` command. Don't forget to substitute "<kernel version>" with the version of the kernel just compiled:

```
root # find /lib/modules/<kernel version>/ -type f -iname '*.o' -or -iname '*.ko' | less
```

For instance, to automatically load the `3c59x.ko` module (which is the driver for a specific 3Com network card family), edit the `/etc/conf.d/modules` file and enter the module name in it.

```
root # nano -w /etc/conf.d/modules
```

```
modules="3c59x"
```

Continue the installation with Configuring the system (/wiki/Handbook:AMD64/Installation/System).

Optional: Installing firmware

Some drivers require additional firmware to be installed on the system before they work. This is often the case for network interfaces, especially wireless network interfaces. Most of the firmware is packaged in sys-kernel/linux-firmware (http://packages.gentoo.org/package/sys-kernel/linux-firmware):

```
root # emerge --ask sys-kernel/linux-firmware
```

Filesystem information

About fstab

Under Linux, all partitions used by the system must be listed in `/etc/fstab`. This file contains the mount points of those partitions (where they are seen in the file system structure), how they should be mounted and with what special options (automatically or not, whether users can mount them or not, etc.)

Creating the fstab file

The `/etc/fstab` file uses a table-like syntax. Every line consists of six fields, separated by whitespace (space(s), tabs or a mixture). Each field has its own meaning:

1. The first field shows the partition described (the path to the device file)
2. The second field shows the mount point at which the partition should be mounted
3. The third field shows the filesystem used by the partition
4. The fourth field shows the mount options used by `mount` when it wants to mount the partition. As every filesystem has its own mount options, users are encouraged to read the mount man page (`man mount`) for a full listing. Multiple mount options are comma-separated.
5. The fifth field is used by dump to determine if the partition needs to be dumped or not. This can generally be left as 0 (zero).
6. The sixth field is used by fsck to determine the order in which filesystems should be checked if the system

wasn't shut down properly. The root filesystem should have 1 while the rest should have 2 (or 0 if a filesystem check isn't necessary).

Important
The default `/etc/fstab` file provided by Gentoo is not a valid fstab file but instead more of a template.

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

In the remainder of the text, we use the default `/dev/sd*` block device files as partition. Users can also opt to use the symbolic links in the `/dev/disk/by-id/` or `/dev/disk/by-uuid/` locations. These names are not likely to change, whereas the default block device files naming depends on a number of factors (such as how and in what order the disks are attached to the system). However, unless someone intends to fiddle with the disk ordering, one can continue with the default block device files safely.

Let us take a look at how to write down the options for the `/boot/` partition. This is just an example, and should be modified according to the partitioning decisions made earlier in the installation. In our amd64 partitioning example, `/boot/` is usually the `/dev/sda2` partition, with ext2 as filesystem. It needs to be checked during boot, so we would write down:

FILE `/etc/fstab` **An example /boot line for /etc/fstab**

```
/dev/sda2   /boot     ext2    defaults        0 2
```

Some users don't want their `/boot/` partition to be mounted automatically to improve their system's security. Those people should substitute *defaults* with *noauto*. This does mean that those users will need to manually mount this partition every time they want to use it.

Add the rules that match the previously decided partitioning scheme and append rules for devices such as CD-ROM drive(s), and of course, if other partitions or drives are used, for those too.

Below is a more elaborate example of an `/etc/fstab` file:

FILE `/etc/fstab` **A full /etc/fstab example**

```
/dev/sda2   /boot        ext2    defaults,noatime     0 2
/dev/sda3   none         swap    sw                   0 0
/dev/sda4   /            ext4    noatime              0 1

/dev/cdrom  /mnt/cdrom   auto    noauto,user          0 0
```

When *auto* is used in the third field, it makes the `mount` command guess what the filesystem would be. This is recommended for removable media as they can be created with one of many filesystems. The *user* option in the fourth field makes it possible for non-root users to mount the CD.

To improve performance, most users would want to add the *noatime* mount option, which results in a faster system since access times aren't registered (those are not needed generally anyway). This is also recommended for solid state drive (SSD) users, who should also enable the *discard* mount option (ext4 and btrfs only for now) which makes the `TRIM` command work.

Double-check the `/etc/fstab` file, save and quit to continue.

Networking information

Host and domain information

One of the choices the user has to make is name his/her PC. This seems to be quite easy, but lots of users are having difficulties finding the appropriate name for their Linux-pc. To speed things up, know that the decision is not final - it can be changed afterwards. In the examples below, the hostname *tux* is used within the domain *homenetwork*.

```
root # nano -w /etc/conf.d/hostname
```

```
# Set the hostname variable to the selected host name
hostname="tux"
```

Second, if a domainname is needed, set it in `/etc/conf.d/net`. This is only necessary if the ISP or network administrator says so, or if the network has a DNS server but not a DHCP server. Don't worry about DNS or domainnames if the system uses DHCP for dynamic IP address allocation and network configuration.

> **Note**
> The `/etc/conf.d/net` file does not exist by default, so needs to be created.

```
root # nano -w /etc/conf.d/net
```

```
# Set the dns_domain_lo variable to the selected domain name
dns_domain_lo="homenetwork"
```

> **Note**
> If no domainname is configured, then users will notice they get "This is hostname.(none)" messages at their login screen. This should then be fixed by editing `/etc/issue` and deleting the string `.\O` from that file.

If a NIS domain is needed (users that do not know this will not need one), define that one too:

```
root # nano -w /etc/conf.d/net
```

```
# Set the nis_domain_lo variable to the selected NIS domain name
nis_domain_lo="my-nisdomain"
```

> **Note**
> For more information on configuring DNS and NIS, please read the examples provided in `/usr/share/doc/netifrc-*/net.example.bz2` which can be read using `bzless`. Also, it might be interesting to install net-dns/openresolv (http://packages.gentoo.org/package/net-dns/openresolv) to help manage the DNS/NIS setup.

Configuring the network

During the Gentoo Linux installation, networking was already configured. However, that was for the installation CD itself and not for the installed environment. Right now, the network configuration is made for the installed Gentoo Linux system.

> **Note**
> More detailed information about networking, including advanced topics like bonding, bridging, 802.1Q VLANs or wireless networking is covered in the Gentoo Network Configuration section.

All networking information is gathered in /etc/conf.d/net. It uses a straightforward yet perhaps not intuitive syntax. But don't fear, everything is explained below. A fully commented example that covers many different configurations is available in /usr/share/doc/netifrc-*/net.example.bz2.

First install net-misc/netifrc (http://packages.gentoo.org/package/net-misc/netifrc):

```
root # emerge --ask --noreplace net-misc/netifrc
```

DHCP is used by default. For DHCP to work, a DHCP client needs to be installed. This is described later in Installing Necessary System Tools.

If the network connection needs to be configured because of specific DHCP options or because DHCP is not used at all, then open /etc/conf.d/net:

```
root # nano -w /etc/conf.d/net
```

Set both `config_eth0` and `routes_eth0` to enter IP address information and routing information:

Note
This assumes that the network interface will be called eth0. This is, however, very system dependent. It is recommended to assume that the interface is named the same as the interface name when booted from the installation media if the installation media is sufficiently recent. More information can be found in Network Interface Naming.

FILE /etc/conf.d/net **Static IP definition**

```
config_eth0="192.168.0.2 netmask 255.255.255.0 brd 192.168.0.255"
routes_eth0="default via 192.168.0.1"
```

To use DHCP, define `config_eth0` :

FILE /etc/conf.d/net **DHCP definition**

```
config_eth0="dhcp"
```

Please read /usr/share/doc/netifrc-*/net.example.bz2 for a list of all available options. Be sure to also read up on the DHCP client man page if specific DHCP options need to be set.

If the system has several network interfaces, then repeat the above steps for `config_eth1` , `config_eth2` , etc.

Now save the configuration and exit to continue.

Automatically start networking at boot

To have the network interfaces activated at boot, they need to be added to the default runlevel.

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

If the system has several network interfaces, then the appropriate net.* files need to be created just like we did with net.eth0.

If after booting the system we find out that the assumption about the network interface name (which is currently documented as `eth0`) was wrong, then execute the following steps to rectify this:

1. Update the /etc/conf.d/net file with the correct interface name (like `enp3s0` instead of `eth0`)
2. Create new symbolic link (like /etc/init.d/net.enp3s0)
3. Remove the old symbolic link (`rm /etc/init.d/net.eth0`)
4. Add the new one to the default runlevel

5. Remove the old one using `rc-update del net.eth0 default`

The hosts file

Next inform Linux about the network environment. This is defined in `/etc/hosts` and helps in resolving host names to IP addresses for hosts that aren't resolved by the nameserver.

```
root # nano -w /etc/hosts
```

FILE `/etc/hosts` **Filling in the networking information**

```
# This defines the current system and must be set
127.0.0.1     tux.homenetwork tux localhost

# Optional definition of extra systems on the network
192.168.0.5   jenny.homenetwork jenny
192.168.0.6   benny.homenetwork benny
```

Save and exit the editor to continue.

Optional: Get PCMCIA working

PCMCIA users should now install the sys-apps/pcmciautils (http://packages.gentoo.org/package/sys-apps/pcmciautils) package.

```
root # emerge --ask sys-apps/pcmciautils
```

System information

Root password

Set the root password using the `passwd` command.

```
root # passwd
```

The root Linux account is an all-powerful account, so pick a strong password. Later an additional regular user account will be created for daily operations.

Init and boot configuration

Gentoo (at least when using OpenRC) uses `/etc/rc.conf` to configure the services, startup, and shutdown of a system. Open up `/etc/rc.conf` and enjoy all the comments in the file. Review the settings and change where needed.

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

Next, open `/etc/conf.d/keymaps` to handle keyboard configuration. Edit it to configure and select the right keyboard.

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

Take special care with the `keymap` variable. If the wrong keymap is selected, then weird results will come up when typing on the keyboard.

Finally, edit `/etc/conf.d/hwclock` to set the clock options. Edit it according to personal preference.

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

If the hardware clock is not using UTC, then it is necessary to set `clock="local"` in the file. Otherwise the system might show clock skew behavior.

System logger

Some tools are missing from the stage3 archive because several packages provide the same functionality. It is now up to the user to choose which ones to install.

The first tool to decide on has to provide logging facilities for the system. Unix and Linux have an excellent history of logging capabilities -- if needed, everything that happens on the system can be logged in log files. This happens through the system logger.

Gentoo offers several system loggers to choose from. A few of these are:

- app-admin/sysklogd (http://packages.gentoo.org/package/app-admin/sysklogd), which is the traditional set of system logging daemons
- app-admin/syslog-ng (http://packages.gentoo.org/package/app-admin/syslog-ng), an advanced system logger
- app-admin/metalog (http://packages.gentoo.org/package/app-admin/metalog) which is a highly-configurable system logger

Others are available through portage as well - the number of available packages increases on a daily basis.

If sysklogd or syslog-ng is going to be used, it is recommended to install app-admin/logrotate (http://packages.gentoo.org/package/app-admin/logrotate) afterwards as those system loggers don't provide any rotation mechanism for the log files.

To install the system logger of choice, emerge it and have it added to the default runlevel using `rc-update`. The following example installs app-admin/syslog-ng (http://packages.gentoo.org/package/app-admin/syslog-ng)

```
root # emerge --ask app-admin/syslog-ng
root # rc-update add syslog-ng default
```

Optional: Cron daemon

Next is the cron daemon. Although it is optional and not required for every system, it is wise to install one.

A cron daemon executes scheduled commands. It is very handy if some command needs to be executed regularly (for instance daily, weekly or monthly).

Gentoo offers several possible cron daemons, including sys-process/bcron (http://packages.gentoo.org/package/sys-process/bcron), sys-process/dcron (http://packages.gentoo.org/package/sys-process/dcron), sys-process/fcron (http://packages.gentoo.org/package/sys-process/fcron) and sys-process/cronie (http://packages.gentoo.org/package/sys-process/cronie). Installing one of them is similar to installing a system logger. The following example uses sys-process/cronie (http://packages.gentoo.org/package/sys-process/cronie):

```
root # emerge --ask sys-process/cronie
root # rc-update add cronie default
```

If dcron or fcron are used, an additional initialization command needs to be executed:

```
root # crontab /etc/crontab
```

Optional: File indexing

In order to index the file system to provide faster file location capabilities, install sys-apps/mlocate (http://packages.gentoo.org/package/sys-apps/mlocate).

```
root # emerge --ask sys-apps/mlocate
```

Optional: Remote access

To be able to access the system remotely after installation, add the `sshd` init script to the default runlevel:

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

If serial console access is needed (which is possible in case of remote servers), uncomment the serial console section in `/etc/inittab`:

```
root # nano -w /etc/inittab
```

```
# SERIAL CONSOLES
s0:12345:respawn:/sbin/agetty 9600 ttyS0 vt100
s1:12345:respawn:/sbin/agetty 9600 ttyS1 vt100
```

Filesystem tools

Depending on the filesystems used, it is necessary to install the necessary file system utilities (for checking the filesystem integrity, creating additional file systems etc.). Please note that tools for managing ext2, ext3 or ext4 filesystems (e2fsprogs) are already installed as a part of the system.

The following table lists the tools to install if a certain file system is used:

Filesystem	Package
XFS	sys-fs/xfsprogs (http://packages.gentoo.org/package/sys-fs/xfsprogs)
ReiserFS	sys-fs/reiserfsprogs (http://packages.gentoo.org/package/sys-fs/reiserfsprogs)
JFS	sys-fs/jfsutils (http://packages.gentoo.org/package/sys-fs/jfsutils)
VFAT (FAT32, ...)	sys-fs/dosfstools (http://packages.gentoo.org/package/sys-fs/dosfstools)
Btrfs	sys-fs/btrfs-progs (http://packages.gentoo.org/package/sys-fs/btrfs-progs)

Networking tools

If there is no need for any additional networking tools, continue immediately with the section on Configuring a bootloader (/wiki/Handbook:AMD64/Installation/Bootloader).

Installing a DHCP client

Important
Although optional, the majority of users will find that they need a DHCP client to get on their network. Please install a DHCP client. If this is forgotten, then the system might not be able to get on the network, and thus cannot install a DHCP client afterwards as well.

In order for the system to automatically obtain an IP address for one or more network interface(s), it is necessary to install a DHCP client. We recommend the use of net-misc/dhcpcd (http://packages.gentoo.org/package/net-misc/dhcpcd) although many other DHCP clients are available through the portage tree as well:

```
root # emerge --ask net-misc/dhcpcd
```

Optional: Installing a PPPoE client

If PPP is used to connect to the internet, install the net-dialup/ppp (http://packages.gentoo.org/package/net-dialup/ppp) package:

```
root # emerge --ask net-dialup/ppp
```

Now continue with Configuring the bootloader (/wiki/Handbook:AMD64/Installation/Bootloader).

Selecting a boot loader

With the Linux kernel configured, system tools installed and configuration files edited, it is time to install the last important piece of a Linux installation: the boot loader.

The boot loader is responsible for firing up the Linux kernel upon boot - without it, the system would not know how to proceed when the power button has been pressed.

For amd64, we document how to configure either GRUB2 or LILO for BIOS based systems, and efibootmgr for UEFI systems.

Default: Using GRUB2

Previously, Gentoo Linux used what is now called *GRUB Legacy* as the recommended boot loader. As the name implies, the older GRUB package is no longer actively maintained and has been superseded by GRUB2. For more information about the legacy GRUB, please refer to its GRUB (/wiki/GRUB) article on the Gentoo Wiki.

Installing GRUB2

GRUB2 is provided through the sys-boot/grub (http://packages.gentoo.org/package/sys-boot/grub) package.

```
root # emerge --ask sys-boot/grub
```

The GRUB2 software is now installed on the system, but not activated yet.

Configuring GRUB2

Next, install the necessary GRUB2 files in `/boot/grub/`. Assuming the first disk (the one where the system boots from) is `/dev/sda`, the following command will do this:

```
root # grub2-install /dev/sda
```

Next, we can generate the GRUB2 configuration based on the user configuration specified in the `/etc/default/grub` file and `/etc/grub.d` scripts. In most cases, no configuration is needed by users as GRUB2 will automatically detect which kernel to boot (the highest one available in `/boot/`) and what the root file system is.

To generate the final GRUB2 configuration, run the `grub2-mkconfig` command:

```
root # grub2-mkconfig -o /boot/grub/grub.cfg
```

```
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-3.16.5-gentoo
Found initrd image: /boot/initramfs-genkernel-amd64-3.16.5-gentoo
done
```

The output of the command must mention that at least one Linux image is found, as those are needed to boot the system. If an initramfs is used or `genkernel` was used to build the kernel, the correct initrd image should be detected as well. If this is not the case, go to `/boot/` and check the contents using the `ls` command. If the files are indeed missing, go back to the kernel configuration and installation instructions.

Alternative: Using LILO

Installing LILO

LILO, the LInuxLOader, is the tried and true workhorse of Linux boot loaders. However, it lacks some features that GRUB has. The reason why LILO is still used is that, on some systems, GRUB doesn't work and LILO does. Of course, it is also used because some people know LILO and want to stick with it. Either way, Gentoo supports both.

Installing LILO is a breeze; just use emerge.

```
root # emerge --ask sys-boot/lilo
```

Configuring LILO

To configure LILO, first create `/etc/lilo.conf`:

```
root # nano -w /etc/lilo.conf
```

In the configuration file, sections are used to refer to the bootable kernel. Make sure that the kernel files (with kernel version) and initramfs files are known, as they need to be referred to in this configuration file.

> **Note**
> If the root filesystem is JFS, add an `append="ro"` line after each boot item since JFS needs to replay its log before it allows read-write mounting.

FILE `/etc/lilo.conf` **Example LILO configuration**

```
boot=/dev/sda             # Install LILO in the MBR
prompt                    # Give the user the chance to select another section
timeout=50                # Wait 5 (five) seconds before booting the default section
default=gentoo            # When the timeout has passed, boot the "gentoo" section

image=/boot/vmlinuz-3.16.5-gentoo
  label=gentoo            # Name we give to this section
  read-only               # Start with a read-only root. Do not alter!
  root=/dev/sda4          # Location of the root filesystem

image=/boot/vmlinuz-3.16.5-gentoo
  label=gentoo.rescue     # Name we give to this section
  read-only               # Start with a read-only root. Do not alter!
  root=/dev/sda4         # Location of the root filesystem
  append="init=/bin/bb"   # Launch the Gentoo static rescue shell

# The next two lines are only if you dualboot with a Windows system.
# In this example, Windows is hosted on /dev/sda6.
other=/dev/sda6
  label=windows
```

Note
If a different partitioning scheme and/or kernel image is used, adjust accordingly.

If an initramfs is necessary, then change the configuration by referring to this initramfs file and telling the initramfs where the real root device is at:

FILE `/etc/lilo.conf` **Adding initramfs information to a boot entry**

```
image=/boot/vmlinuz-3.16.5-gentoo
  label=gentoo
  read-only
  append="real_root=/dev/sda4"
  initrd=/boot/initramfs-genkernel-amd64-3.16.5-gentoo
```

If additional options need to be passed to the kernel, use an `append` statement. For instance, to add the `video` statement to enable framebuffer:

FILE `/etc/lilo.conf` **Adding video parameter to the boot options**

```
image=/boot/vmlinuz-3.16.5-gentoo
  label=gentoo
  read-only
  root=/dev/sda4
  append="video=uvesafb:mtrr,ywrap,1024x768-32@85"
```

Users that used `genkernel` should know that their kernels use the same boot options as is used for the installation CD. For instance, if SCSI device support needs to be enabled, add `doscsi` as kernel option.

Now save the file and exit.

To finish up, run `/sbin/lilo` so LILO can apply the `/etc/lilo.conf` settings to the system (i.e. install itself on the disk). Keep in mind that `/sbin/lilo` needs to be executed every time a new kernel is installed or a change has been made to the `lilo.conf` file.

```
root # /sbin/lilo
```

Alternative: Using efibootmgr

On UEFI based systems, the boot loader itself is the UEFI firmware of the system. Such systems do not need additional boot loaders to help boot the system, although EFI-based bootloaders do exist to extend the functionality of UEFI systems during boot.

The sys-boot/efibootmgr (http://packages.gentoo.org/package/sys-boot/efibootmgr) application is not a boot loader, but a tool to interact with the UEFI firmware and update its settings, so that the Linux kernel that was previously installed can be booted with additional options (if necessary), or to allow multiple boot entries. This interaction is done through the EFI variables (hence the need for the support of EFI vars in the past).

> **Note**
> efibootmgr is *not* a requirement to boot from an UEFI system. The Linux kernel itself can (and will) be booted immediately, and additional boot options can be made part of the Linux kernel itself (there is a kernel configuration that allows the user to specify the boot parameters). Even an initramfs can be made part of the kernel itself.

First install the software:

```
root # emerge --ask sys-boot/efibootmgr
```

Next, tell the UEFI firmware that a boot entry called "Gentoo" is to be created, which has the freshly installed Linux kernel booted:

```
root # efibootmgr --create --disk /dev/sda --part 2 --label "Gentoo" --loader "\efi\boot\bootx64.efi"
```

If an initial ram file system (initramfs) is used, add the proper boot option to it:

```
root # efibootmgr -c -d /dev/sda -p 2 -L "Gentoo" -l "\efi\boot\bootx64.efi" initrd='\initramfs-genkernel-amd64-3.16.5-gentoo'
```

> **Note**
> The use of `\` as directory separator is mandatory when using EFI definitions.

With these changes done, when the system reboots, a boot entry called "Gentoo" will be available.

Rebooting the system

Exit the chrooted environment and unmount all mounted partitions. Then type in that one magical command that initiates the final, true test: `reboot` .

```
root # exit
cdimage ~# cd
cdimage ~# umount -l /mnt/gentoo/dev{/shm,/pts,}
cdimage ~# umount /mnt/gentoo{/boot,/sys,/proc,}
cdimage ~# reboot
```

Of course, don't forget to remove the bootable CD, otherwise the CD might be booted again instead of the new Gentoo system.

Once rebooted in the freshly installed Gentoo environment, finish up with Finalizing the Gentoo installation (/wiki/Handbook:AMD64/Installation/Finalizing).

User administration

Adding a user for daily use

Working as root on a Unix/Linux system is dangerous and should be avoided as much as possible. Therefore it is strongly recommended to add a user for day-to-day use.

The groups the user is member of define what activities the user can perform. The following table lists a number of important groups:

Group	Description
audio	be able to access the audio devices
cdrom	be able to directly access optical devices
floppy	be able to directly access floppy devices
games	be able to play games
portage	be able to access portage restricted resources
usb	be able to access USB devices
video	be able to access video capturing hardware and doing hardware acceleration
wheel	be able to use su

For instance, to create a user called john who is member of the *wheel*, *users* and *audio* groups, log in as root first (only root can create users) and run `useradd` :

Login: root

```
Password: (Enter the root password)
```

root # useradd -m -G users,wheel,audio -s /bin/bash john

root # passwd john

```
Password: (Enter the password for john)
Re-enter password: (Re-enter the password to verify)
```

If a user ever needs to perform some task as root, they can use `su -` to temporarily receive root privileges. Another way is to use the sudo (/wiki/Sudo) package which is, if correctly configured, very secure.

Disk cleanup

Removing tarballs

With the Gentoo installation finished and the system rebooted, if everything has gone well, we can now remove the downloaded stage3 tarball from the hard disk. Remember that they were downloaded to the / directory.

```
root # rm /stage3-*.tar.bz2*
```

Where to go from here

Documentation

Where to go from here? What are the options now? What to explore first? Gentoo provides its users with lots of possibilities, and therefore lots of documented (and less documented) features.

Definitely take a look at the next part of the Gentoo Handbook entitled Working with Gentoo (/wiki/Handbook:AMD64/Working/Portage) which explains how to keep the software up to date, how to install more software, what USE flags are, how the Gentoo init system works, etc.

Apart from the handbook, you should also feel encouraged to explore other corners of the Gentoo Wiki to find additional, community-provided documentation. The Gentoo documentation team also offers a Documentation overview (/wiki/Project:Documentation/Overview) which lists a fine selection of Wiki articles. For instance, it refers to the localization guide (/wiki/Localization/HOWTO) to make a system feel more at home.

Gentoo online

Everyone is of course always welcome on our Gentoo forums (https://forums.gentoo.org) or on one of our many Gentoo IRC channels (https://www.gentoo.org/main/en/irc.xml).

We also have several mailing lists (https://www.gentoo.org/main/en/lists.xml) open to all our users. Information on how to join is contained in that page.

Enjoy your installation. :)