Gentoo Installation

Booting

Download the iso from gentoo site, and burn it to a disc. Booting from the live installation media, boot into the gentoo kernel

```
boot: gentoo
```

Network

Check if the network is setup automatically with `ifconfig`. Test the network with ping.

```
ping -c 3 www.google.com
```

Partition Scheme (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 (ESP) is created. The instructions for parted below contain the necessary pointers to correctly handle this operation.

```
mkfs.fat -F 32 /dev/sda2
```

We are going to follow the following partitioning scheme for our gentoo installation.

Partition	Filesystem	Size
/dev/sda1	(bootloader)	2M
/dev/sda2	ext2 (or fat32 if UEFI is being used)	128M
/dev/sda3	(swap)	512M or higher
/dev/sda4	ext4	Rest of the disk

We use cfdisk for partitioning the disk

```
cfdisk /dev/sda
```

Creating Filesystems

We use ext4 filesystem for our partitions.

```
mkfs.ext4 /dev/sda2
mkfs.ext4 /dev/sda4
```

> mkfs.ext4 command comes from the package `sys-fs/e2fsprogs`
> mkfs.ntfs command comes from the package `sys-fs/ntfs-3g`

Acticating SWAP partition

```
mkswap /dev/sda3
swapon /dev/sda3
```

Mounting the ROOT partition

```
mount /dev/sda4 /mnt/gentoo
```

Installing the Gentoo installation files

Installing a stage tarball

We first set the date and time

```
ntpd -qg
date
```

We are going to use the multilib(both 32 and 64 bit support) tarball for our installation. We download the tarball using a terminal based web browser "links".

```
cd /mnt/gentoo
links https://www.gentoo.org/downloads/mirrors/
```

Unpacking the stage tarball

```
tar xpvf stage3-*.tar.xz --xattrs-include='*.*' --numeric-owner
```

Configuring compile options

ptimize Gentoo, it is possible to set a couple of variables which impacts the behavior of Portage, Gentoo's officially supported package manager. 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.

You can read up more on this on the [gentoo handbook.] (https://wiki.gentoo.org/wiki/Handbook:AMD64/Installation/St

I have make.conf file setup specifically for my system. And it is available in this repository. We are going to copy it to `/etc/portage` and use it. The same make.conf file would not work for all systems, as I have it setup specifically for my current running system.

Installing the Gentoo base system

Chrooting

Select a mirror for downloading your installation files

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

Gentoo ebuild repository

```
mkdir --parents /mnt/gentoo/etc/portage/repos.conf
cp /mnt/gentoo/usr/share/portage/config/repos.conf /mnt/gentoo/etc/
    portage/repos.conf/gentoo.conf
```

We already have a repos.conf in this repository setup as my personal preferences, so we can use that.

Copy DNS info

```
cp --dereference /etc/resolv.conf /mnt/gentoo/etc/
```

Mounting the necessary filesystems

```
mount --types proc /proc /mnt/gentoo/proc
mount --rbind /sys /mnt/gentoo/sys
mount --make-rslave /mnt/gentoo/sys
mount --rbind /dev /mnt/gentoo/dev
mount --make-rslave /mnt/gentoo/dev
```

Entering the new environment

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

Mounting the boot partition

```
mount /dev/sda2 /boot
```

Configuring Portage

Installing a Gentoo ebuild repository snapshot from the web

```
emerge-webrsync
```

We are also going to update the ebuild repository

```
emerge --sync
```

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. These settings are all maintained by Gentoo's Portage developers.

We check the profiles using the following command

```
eselect profile list
```

Viewing the list, we select a profile as per our needs

```
eselect profile set 2
```

Updating the @world set

At this point, it is wise to update the system's @world set so that a base can be established.

This following step is necessary so the system can apply any updates or USE flag changes which have appeared since the stage3 was built and from any profile selection:

```
emerge --ask --verbose --update --deep --newuse @world

OR, emerge -DuvaN @world
```

Configuring the USE variable

You can view your USE variables using the folowing

```
emerge --info
```

You can also readup on USE flags more here

```
less /var/db/repos/gentoo/profiles/use.desc
```

We already have our USE flags configured in our make.conf file. But Setting the flags is very important as it gives a lot of control over support for what is installed on your system and what is not, and makes your system lightweight.

Timezone

```
ls /usr/share/zoneinfo
echo "Europe/Brussels" > /etc/timezone
emerge --config sys-libs/timezone-data
```

Configure locales

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

Uncomment the following for English

```
en_US ISO-8859-1
en_US.UTF-8 UTF-8
```

And then, generate the locales

```
locale-gen
```

Locale selection

```
eselect locale list
```

From the list, select the prefered locale

```
eselect locale set <locale number>
```

Now, reload the environment.

```
env-update && source /etc/profile && export PS1="(chroot) ${PS1}"
```

Installing the sources

Now , we need to get a kernel for a system. For amd64-based systems Gentoo recommends the sys-kernel/gentoo-sources package.

```
emerge --ask sys-kernel/gentoo-sources
ls -l /usr/src/linux
```

Manual configuration

Alternative: Using genkernel

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

Edit the /etc/fstab file and add the following

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

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

Now we compile the kernel sources using genkernel

```
genkernel all
ls /boot/vmlinu* /boot/initramfs*
```

Installing firmware

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

Creating the fstab file

Get the disk UUIDs using `blkid` and put them in the /etc/fstab file. An example fstab file can look like

```
/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
```

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.

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

# IPv4 and IPv6 localhost aliases
127.0.0.1       localhost oculus desktop
::1             localhost oculus desktop

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

Root passwd

Set the root passwd using `passwd`

Adding a user for daily use

```
useradd -m -G users,wheel,audio -s /bin/bash larry
passwd larry
```

Networking setup

Installing a bootloader - GRUB2

For EFI users, the following variable must be set in the make.conf file

```
echo 'GRUB_PLATFORMS="efi-64"' >> /etc/portage/make.conf
emerge --ask --verbose sys-boot/grub:2

grub-install --target=x86_64-efi --efi-directory=/boot
grub-mkconfig -o /boot/grub/grub.cfg
```

Rebooting the system

```
exit
cd
umount -l /mnt/gentoo/dev{/shm,/pts,}
umount -R /mnt/gentoo
reboot
```