Hypervisor Details

Hypervisor: KVM
Architecture: x86_64
Emulator: /usr/bin/qemu-system-x86_64
Chipset: Q35
Firmware: BIOS

Chipset Options: `i440FX, Q35`
Firmware Options: BIOS, UEFI, UEFI x86-64:/usr/share/OVMF/OVMF_CODE_4M.ms.fd, UEFI x86-64:/usr/share/OVMF/OVMF_CODE_4M.secboot.fd, UEFI x86-64:/usr/share/OVMF/OVMF_CODE_4M.fd

Chipset Options: `i440FX, Q35`

When configuring a virtual machine in QEMU/KVM, you often have the choice between two main chipset options: `i440FX` and `Q35`. These options represent different virtual machine hardware configurations, and the choice between them depends on your specific use case and requirements. Here's a comparison of the `i440FX` and `Q35` chipsets:

i440FX:

1. **Description:**
   - Also known as the "440BX" chipset.
   - Represents an older, more traditional chipset.
2. **Characteristics:**
   - **Legacy Support:** Suitable for emulating traditional PC hardware with legacy features.
   - **BIOS:** Often used with BIOS firmware.
   - **PCI Support:** Supports PCI devices.
3. **Use Cases:**
   - **Compatibility:** Good for compatibility with older operating systems and applications.
   - **Legacy Systems:** Suitable for emulating legacy systems with BIOS firmware.

Q35:

1. **Description:**
   - Represents a more modern and feature-rich chipset.
2. **Characteristics:**
   - **UEFI Support:** Supports UEFI firmware, enabling modern boot mechanisms.
   - **PCI Express Support:** Supports PCI Express devices for faster data transfer.
   - **Advanced Features:** Offers advanced virtualization features.
3. **Use Cases:**
   - **Modern Systems:** Preferred for emulating modern hardware and systems.
   - **UEFI Boot:** Suitable for virtual machines that require UEFI boot capabilities.

Choosing Between i440FX and Q35:

- **Compatibility:** If you are running older operating systems or applications that require traditional BIOS support, `i440FX` might be a suitable choice.
- **Modern Systems:** If you are running more modern operating systems or want to take advantage of UEFI boot capabilities, PCI Express support, and other modern features, `Q35` is a better choice.
- **Default Machine Type:** Depending on your QEMU/KVM version, the default machine type might be one of these, and you can explicitly choose it during VM creation or editing.
- **Guest Operating System Support:** Consider the requirements of the guest operating system and the specific features it supports. Some operating systems may work better with one chipset type over the other.

In summary, the choice between `i440FX` and `Q35` depends on your specific use case, the requirements of the guest operating system, and whether you need compatibility with legacy BIOS or want to take advantage of modern virtualization features.

BIOS, UEFI, UEFI x86-64:/usr/share/OVMF/OVMF_CODE_4M.ms.fd, UEFI x86-64:/usr/share/OVMF/OVMF_CODE_4M.secboot.fd, UEFI x86-64:/usr/share/OVMF/OVMF_CODE_4M.fd

The terms "BIOS" and "UEFI" refer to firmware interfaces used in computers, including virtual machines. When setting up a virtual machine, you often need to choose the type of firmware the virtual machine will use. Here's an explanation of these terms and their variations:

1. BIOS (Basic Input/Output System):

- **Description:** BIOS is a traditional firmware interface used for booting the operating system in a computer.
- **Characteristics:**
  - Uses a Master Boot Record (MBR) for partitioning and booting.
  - Limited to 2.2 TB for bootable partitions.
  - Commonly used in older systems and legacy environments.

2. UEFI (Unified Extensible Firmware Interface):

- **Description:** UEFI is a more modern and flexible firmware interface designed to replace BIOS.
- **Characteristics:**
  - Supports both 32-bit and 64-bit architectures.
  - Uses the GUID Partition Table (GPT) for partitioning, allowing larger disk sizes.
  - Supports secure boot, which verifies the authenticity of the bootloader.
  - Offers a graphical user interface and modular architecture.

3. UEFI x86-64 (64-bit UEFI):

When choosing UEFI for a virtual machine, you often need to specify the specific UEFI firmware file. The variations include:

- **UEFI x86-64:**
  - This generally refers to the standard UEFI firmware for 64-bit architectures.

    - It provides the basic UEFI functionality for booting 64-bit operating systems.
- **UEFI x86-64:/usr/share/OVMF/OVMF_CODE_4M.ms.fd:**
    - Specifies the path to the OVMF (Open Virtual Machine Firmware) firmware file with a 4 MB code size, specifically for Microsoft Secure Boot.
    - Used when secure boot is enabled to ensure the integrity of the boot process.
- **UEFI x86-64:/usr/share/OVMF/OVMF_CODE_4M.secboot.fd:**
    - Similar to the previous variant but specifically labeled for "secure boot."
    - Used in conjunction with secure boot features.
- **UEFI x86-64:/usr/share/OVMF/OVMF_CODE_4M.fd:**
    - Refers to the standard OVMF firmware with a 4 MB code size, suitable for regular UEFI booting without specific secure boot requirements.

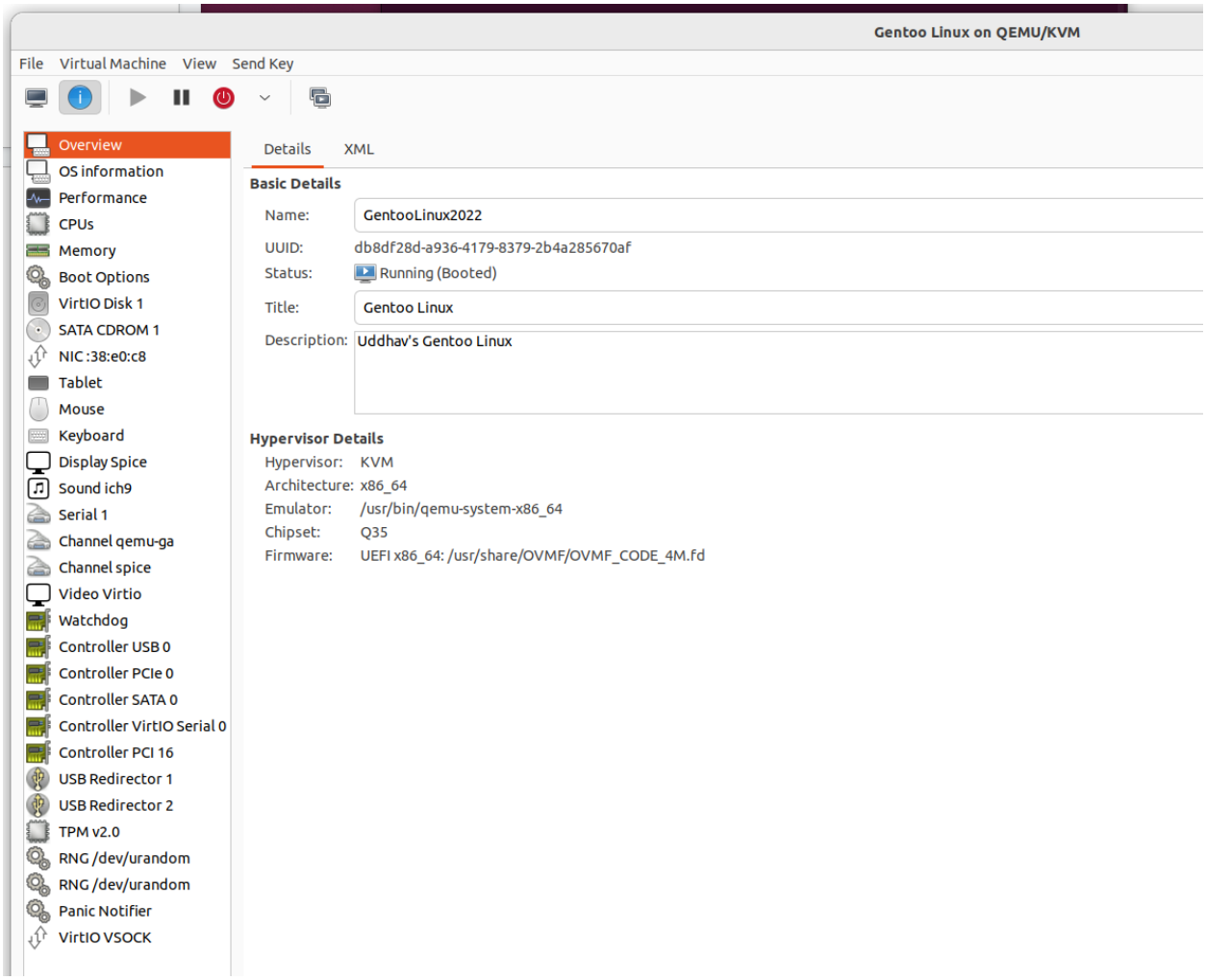
Explanation of Paths:

- `/usr/share/OVMF/`: This is the directory where the OVMF firmware files are commonly stored on Linux systems.
- `OVMF_CODE_4M.ms.fd`, `OVMF_CODE_4M.secboot.fd`, `OVMF_CODE_4M.fd`: These are specific firmware files with different configurations, sizes, and security features.

Choosing the Right Firmware:

- **Secure Boot:** If you need secure boot features, choose the variant with `.ms.fd` or `.secboot.fd`.
- **Regular Boot:** For regular UEFI booting without secure boot, you can use the variant without specific labels (e.g., `OVMF_CODE_4M.fd`).

When setting up a virtual machine, select the UEFI variant that best fits your requirements, considering factors like secure boot, operating system compatibility, and specific security needs.



`OVMF_CODE_4M.fd` typically refers to a file that contains the UEFI firmware code for a virtual machine. Specifically, in the context of the Open Virtual Machine Firmware (OVMF), this file contains the code that implements the UEFI firmware for virtual machines running in a virtualization environment.

UEFI is a standardized interface (e.g., graphical UI UEFI) between the operating system and the firmware that provides several advantages over the older BIOS system. UEFI includes a feature called Secure Boot, which is designed to enhance the system'd security by ensuring that only digitally signed and trusted software components, including the OS and bootloader, are allowed to execute during the boot proces preventing the loading of malicious code during startup. UEFI supports GPT, which allows for the use of larger storage devices and more paritions and also more resilient to data corruption. UEFI can have network based firmware updates. More flexible, extensible and modular so that hardware vendors can add their own UEFI derivers and features. UEFI includes Compatiblity Support Module (CSM) to provide backward compatiblity with legacy BIOS systems. And through the Runtime Services, applications or OS interact with the UEFI Firmware.

The UEFI firmware is stored in a non-volatile memory chip on the computer's motherboard.

When PC gets powered on, the UEFI firmware is the first code that gets executed. That firmware initializes hardware, performs POST (Power-On Self-Test) to check hardware integrity, and initializes the UEFI Runtime Services. POST is a diagnostic process that checks the integrity of various hardware components, ensring that they are functioning correctly. If any issues are detected during the POST, the firmware may halt the boot process and display the error message. When POST is completed, the UEFI firmware initializes the UEFI Runtime Services. These Runtime Services provide a standardized interface for OS and UEFI applications to

interact with firmware during runtime. They include functions for managing system resources, accessing system information, and performing other runtime operations.

The UEFI firmware is typically stored on a flash memory chip on the motherboard. UEFI firmware looks for the UEFI System Paritition (ESP) on the storage devices. This ESP contains the bootloader and related files. Once it knows ESP partition, it looks for UEFI boot manager, which resides in ESP partition and is responsible for locating and launching UEFI applications (i.e., bootloaders). In the context of systemd-boot or any other UEFI bootloader, the bootloader binary is located on the ESP, and it is not part of the UEFI firmware code. In the case of systemd-boot, the firmware loads the `systemd-bootx64.efi` binary from the EFI System Partition.

The UEFI Runtime Services are not located on the EFI System Partition (ESP). Instead, they are part of the firmware itself and reside in a specific memory region. These services are implemented by the UEFI firmware to provide a set of functions that UEFI applications, including the operating system's UEFI runtime environment, can use.

UEFI Firmware -- UEFI Runtime -- OS / UEFI Applications

UEFI Applications: E.g., boot loaders, Systemd-boot manager.

systemd-bootx64.efi

The `systemd-bootx64.efi` file is the compiled binary for the systemd-boot UEFI bootloader, and its specific contents are executable machine code written in the EFI Byte Code (EBC) or x86-64 (64-bit) instruction set architecture.

The actual content of the `systemd-bootx64.efi` binary includes:

1. **Bootloader Logic:**
    - The core logic of the systemd-boot bootloader, which includes code for initializing the UEFI environment, managing the boot process, and presenting a boot menu.
2. **EFI Stub Loader Code:**
    - Implementation of the EFI stub loader functionality that allows systemd-boot to directly boot Linux kernels without the need for an additional bootloader.
3. **Boot Menu Logic:**
    - Code for handling the boot menu, including user interaction, displaying available boot entries, and processing user choices.
4. **Configuration Parsing Code:**

    Logic for reading and parsing configuration files, such as `loader.conf` and entries in the `/loader/entries/` directory on the EFI System Partition. These files contain configuration options and details about available boot entries.

    //loader.conf example

```
# Set the default boot entry
default UddhavGentooLinux
#Gentoo, UddhavGentooLinux, or any label you can use
# Set the timeout for the boot menu (in seconds)
timeout 5

# Specify the console font and mode
console-mode gfx
console-font latarcyrheb-sun32

# Disable the built-in editor for boot entries
```

```
editor no
```

```
/loader/entries/  # individual boot entries
|-- UddhavGentooLinux.conf
|-- Windows.conf
|-- Arch.conf
|-- ...
/loader/
|-- loader.conf  # for global configurations
```

Example of **UddhavGentooLinux**.conf

```
# /loader/entries/UddhavGentooLinux.conf
title Gentoo Linux
linux /vmlinuz-linux
initrd /initramfs-linux.img
options root=PARTUUID=01234567-89ab-cdef-0123-456789abcdef ro
```

initramfs is initial RAM file system that gets loaded into the RAM during boot process. It contains essentials files and drivers needed to mount the root filesystem and initiate the boot process. initramfs contains minimal set of essential kernel modules and drivers needed to access the storage device and other critical hardware components. This time root file system is not yet available. Before the kernel can mount the root filesystem, it need to identify and initialize the storage devices, such as hard drives. The initramfs contains tools and drivers necessary for this task. Wihtout initramfs, the kernel might not have the required drivers to acces the root filesystem, leading to a failure in the boot process. Once the kernel has identified the storage devices and loaded the necessary modules, initramfs assists in mounting the real root file system. This is a critical step in transitioning from the temporary initramfs to the actual root file system where the OS is installed. The initramfs can include custom scripts or programs that perform specific tasks required for the system to boot successfully. This can include tasks such as setting up encryption, configuring network interfaces, or loading additional modules based on hardware detection.

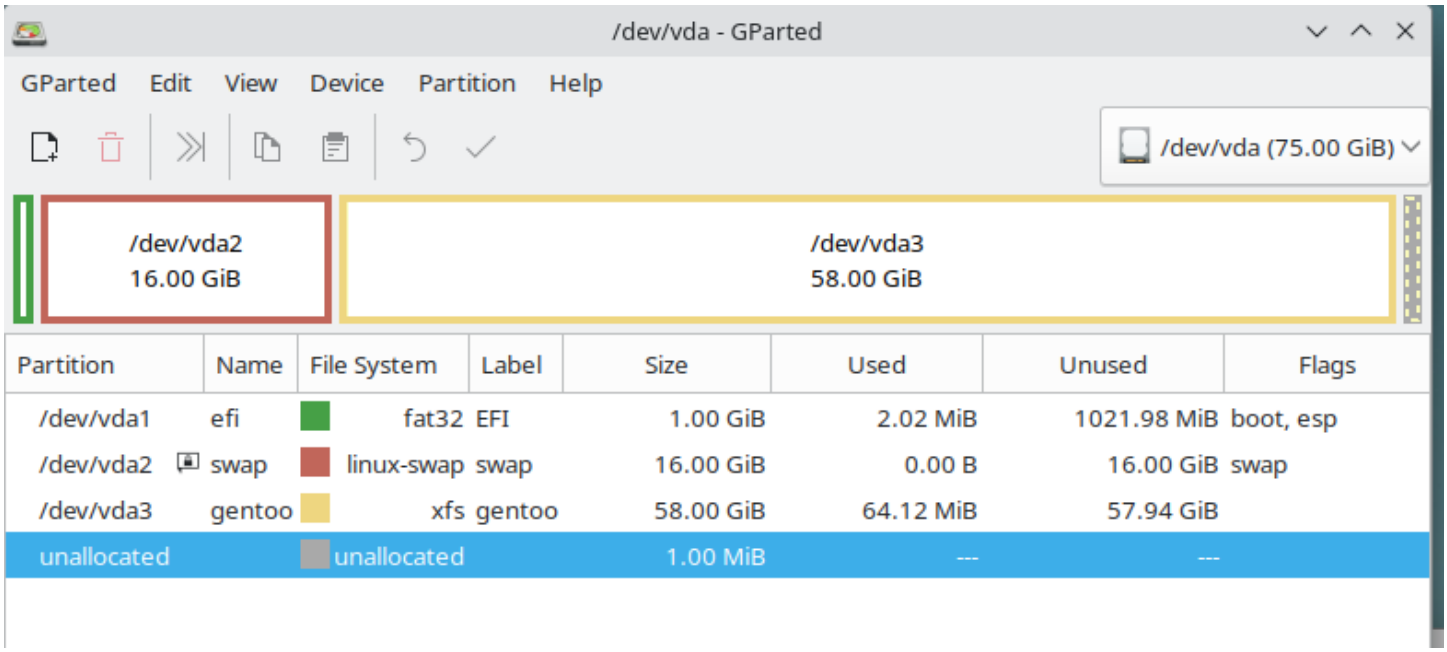
vmlinux-kernel-version is the core component of the OS responsible for managing hardware resources and providing essential services.

5. **Interaction with UEFI Firmware:**
    - Code to interact with UEFI firmware services, including accessing UEFI runtime services, managing UEFI variables, and initiating the boot process.

Error during formatting using Gparted
Unable to read the contents of this file system!
Because of this some operations may be unavailable.
The cause might be a missing software package.
The following list of software packages is required for fat32 file system support: dosfstools, mtools.

emerge --sync
emerge --ask sys-fs/dosfstools sys-fs/mtools

Installing Gentoo Linux in Normal UEFI mode using Q35




```
Boot flag in GPT is equivalent to active parition in MBR.
ESP: EFI System Partition
```

GUID Partition Table (GPT) and UEFI boot.

/dev/vda1	**fat32**	1 GiB	**EFI System Partition details.**
	xfs		MBR DOS/legacy BIOS boot partition details.
/dev/vda2	**linux-swap**	**RAM size * 2**	**Swap partition details.**
/dev/vda3	**xfs**	**Remainder of the disk**	**Root partition details.**

```
livecd /home/gentoo # mkdir --parents /mnt/gentoo

livecd /home/gentoo # mount /dev/vda3 /mnt/gentoo/ && cd /mnt/gentoo

livecd /mnt/gentoo # wget
https://distfiles.gentoo.org/releases/amd64/autobuilds/20231224T164659Z/stage3-
amd64-desktop-systemd-mergedusr-20231224T164659Z.tar.xz
--2023-12-27 15:49:21--
https://distfiles.gentoo.org/releases/amd64/autobuilds/20231224T164659Z/stage3-
amd64-desktop-systemd-mergedusr-20231224T164659Z.tar.xz
Resolving distfiles.gentoo.org... 89.187.177.16, 156.146.36.23,
2a02:6ea0:c400::11, ...
Connecting to distfiles.gentoo.org|89.187.177.16|:443... connected.
HTTP request sent, awaiting response... 200 OK
```

```
Length: 744754168 (710M) [application/x-xz]
Saving to: ‘stage3-amd64-desktop-systemd-mergedusr-20231224T164659Z.tar.xz’

stage3-amd64-desktop-systemd-mergedusr-20
100%[=============================================================================
===========>] 710.25M  3.56MB/s    in 3m 55s

2023-12-27 15:53:16 (3.02 MB/s) -
‘stage3-amd64-desktop-systemd-mergedusr-20231224T164659Z.tar.xz’ saved
[744754168/744754168]

livecd /mnt/gentoo # tar xpvf stage3-*.tar.xz --xattrs-include='*.*'
--numeric-owner


livecd /mnt/gentoo # nano /mnt/gentoo/etc/portage/make.conf

MAKEOPTS="-j6 -l4"
```

Installing Base System

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

livecd /mnt/gentoo # cat /mnt/gentoo/etc/portage/make.conf
# These settings were set by the catalyst build script that automatically
# built this stage.
# Please consult /usr/share/portage/config/make.conf.example for a more
# detailed example.
COMMON_FLAGS="-O2 -pipe"
CFLAGS="${COMMON_FLAGS}"
CXXFLAGS="${COMMON_FLAGS}"
FCFLAGS="${COMMON_FLAGS}"
FFLAGS="${COMMON_FLAGS}"

# NOTE: This stage was built with the bindist Use flag enabled

# This sets the language of build output to English.
# Please keep this setting intact when reporting bugs.
LC_MESSAGES=C.utf8

MAKEOPTS="-j6 -l4"

GENTOO_MIRRORS="https://mirror.clarkson.edu/gentoo/ \
   http://mirror.clarkson.edu/gentoo/ \
   rsync://mirror.clarkson.edu/gentoo/ \
   http://www.gtlib.gatech.edu/pub/gentoo \
   rsync://rsync.gtlib.gatech.edu/gentoo \
```

```
   https://mirrors.mit.edu/gentoo-distfiles/ \
   http://mirrors.mit.edu/gentoo-distfiles/ \
   rsync://mirrors.mit.edu/gentoo-distfiles/ \
   https://gentoo.osuosl.org/ \
   http://gentoo.osuosl.org/ \
   https://mirrors.rit.edu/gentoo/ \
   http://mirrors.rit.edu/gentoo/ \
   ftp://mirrors.rit.edu/gentoo/ \
   rsync://mirrors.rit.edu/gentoo/ \
   https://mirror.servaxnet.com/gentoo/ \
   http://mirror.servaxnet.com/gentoo/ \
   http://gentoo-mirror.flux.utah.edu/"

livecd /mnt/gentoo # mkdir --parents /mnt/gentoo/etc/portage/repos.conf


livecd /mnt/gentoo # cp /mnt/gentoo/usr/share/portage/config/repos.conf
/mnt/gentoo/etc/portage/repos.conf/gentoo.conf


livecd /mnt/gentoo # cat /mnt/gentoo/etc/portage/repos.conf/gentoo.conf
[DEFAULT]
main-repo = gentoo

[gentoo]
location = /var/db/repos/gentoo
sync-type = rsync
sync-uri = rsync://rsync.gentoo.org/gentoo-portage
auto-sync = yes
sync-rsync-verify-jobs = 1
sync-rsync-verify-metamanifest = yes
sync-rsync-verify-max-age = 3
sync-openpgp-key-path = /usr/share/openpgp-keys/gentoo-release.asc
sync-openpgp-keyserver = hkps://keys.gentoo.org
sync-openpgp-key-refresh-retry-count = 40
sync-openpgp-key-refresh-retry-overall-timeout = 1200
sync-openpgp-key-refresh-retry-delay-exp-base = 2
sync-openpgp-key-refresh-retry-delay-max = 60
sync-openpgp-key-refresh-retry-delay-mult = 4
sync-webrsync-verify-signature = yes


livecd /mnt/gentoo # cp --dereference /etc/resolv.conf /mnt/gentoo/etc/
livecd /mnt/gentoo # mount --types proc /proc /mnt/gentoo/proc
livecd /mnt/gentoo # mount --rbind /sys /mnt/gentoo/sys
livecd /mnt/gentoo # mount --make-rslave /mnt/gentoo/sys
livecd /mnt/gentoo # mount --rbind /dev /mnt/gentoo/dev
livecd /mnt/gentoo # mount --make-rslave /mnt/gentoo/dev
livecd /mnt/gentoo # mount --bind /run /mnt/gentoo/run
```

```
livecd /mnt/gentoo # mount --make-slave /mnt/gentoo/run
livecd /mnt/gentoo # chroot /mnt/gentoo /bin/bash
livecd / # source /etc/profile
livecd / # export PS1="(chroot) ${PS1}"
(chroot) livecd / #
```

```
(chroot) livecd / # mkdir /boot/efi
(chroot) livecd / # mount /dev/vda1 /boot/efi

(chroot) livecd / # emerge-webrsync
(chroot) livecd / # emerge --sync (optional)
```

```
(chroot) livecd / # eselect profile list
Available profile symlink targets:
  [1]   default/linux/amd64/17.1 (stable)
  [2]   default/linux/amd64/17.1/selinux (stable)
  [3]   default/linux/amd64/17.1/hardened (stable)
  [4]   default/linux/amd64/17.1/hardened/selinux (stable)
  [5]   default/linux/amd64/17.1/desktop (stable)
  [6]   default/linux/amd64/17.1/desktop/gnome (stable)
  [7]   default/linux/amd64/17.1/desktop/gnome/systemd (stable)
  [8]   default/linux/amd64/17.1/desktop/gnome/systemd/merged-usr (stable)
  [9]   default/linux/amd64/17.1/desktop/plasma (stable)
  [10]  default/linux/amd64/17.1/desktop/plasma/systemd (stable)
  [11]  default/linux/amd64/17.1/desktop/plasma/systemd/merged-usr (stable)
  [12]  default/linux/amd64/17.1/desktop/systemd (stable)
  [13]  default/linux/amd64/17.1/desktop/systemd/merged-usr (stable) *
  [14]  default/linux/amd64/17.1/developer (exp)
  [15]  default/linux/amd64/17.1/no-multilib (stable)
  [16]  default/linux/amd64/17.1/no-multilib/hardened (stable)
  [17]  default/linux/amd64/17.1/no-multilib/hardened/selinux (stable)
  [18]  default/linux/amd64/17.1/no-multilib/systemd (dev)
  [19]  default/linux/amd64/17.1/no-multilib/systemd/merged-usr (dev)
  [20]  default/linux/amd64/17.1/no-multilib/systemd/selinux (exp)
  [21]  default/linux/amd64/17.1/no-multilib/systemd/selinux/merged-usr (exp)
  [22]  default/linux/amd64/17.1/systemd (stable)
  [23]  default/linux/amd64/17.1/systemd/merged-usr (stable)
  [24]  default/linux/amd64/17.1/systemd/selinux (exp)
  [25]  default/linux/amd64/17.1/systemd/selinux/merged-usr (exp)
  [26]  default/linux/amd64/17.1/clang (exp)
  [27]  default/linux/amd64/17.1/systemd/clang (exp)
  [28]  default/linux/amd64/17.1/systemd/clang/merged-usr (exp)
  [29]  default/linux/amd64/17.0/x32 (dev)
  [30]  default/linux/amd64/17.0/x32/systemd (exp)
  [31]  default/linux/amd64/17.0/x32/systemd/merged-usr (exp)
  [32]  default/linux/amd64/17.0/musl (dev)
  [33]  default/linux/amd64/17.0/musl/clang (exp)
```

```
[34]  default/linux/amd64/17.0/musl/hardened (exp)
[35]  default/linux/amd64/17.0/musl/hardened/selinux (exp)
```

Everyt time we change profie or do something, the good practice we need to follow is first "emerge --sync" to synchronize portege tree with the latest versions available from Gentoo repository, and then "emerge -auDN" upgrade dependencies uisng deep scanning considering newly updated USE flags.

```
(chroot) livecd / # emerge --ask --verbose --update --deep --newuse @world
```

```
(chroot) livecd / # emerge --info | grep ^USE
```

USE="X a52 aac acl acpi alsa amd64 bluetooth branding bzip2 cairo cdda cdr cli crypt cups dbus dri dts dvd dvdr encode exif flac fortran gdbm gif gpm gtk gui iconv icu ipv6 jpeg lcms libnotify libtirpc mad mng mp3 mp4 mpeg multilib ncurses nls nptl ogg opengl openmp pam pango pcre pdf png policykit ppds qt5 readline sdl seccomp sound spell ssl startup-notification svg systemd test-rust tiff truetype udev udisks unicode upower usb vorbis vulkan wxwidgets x264 xattr xcb xft xml xv xvid zlib" ABI_X86="64" ADA_TARGET="gnat_2021" APACHE2_MODULES="authn_core authz_core socache_shmcb unixd actions alias auth_basic authn_anon authn_dbm authn_file authz_dbm authz_groupfile authz_host authz_owner authz_user autoindex cache cgi cgid dav dav_fs dav_lock deflate dir env expires ext_filter file_cache filter headers include info log_config logio mime mime_magic negotiation rewrite setenvif speling status unique_id userdir usertrack vhost_alias" CALLIGRA_FEATURES="karbon sheets words" COLLECTD_PLUGINS="df interface irq load memory rrdtool swap syslog" CPU_FLAGS_X86="mmx mmxext sse sse2" ELIBC="glibc" GPSD_PROTOCOLS="ashtech aivdm earthmate evermore fv18 garmin garmintxt gpsclock greis isync itrax mtk3301 ntrip navcom oceanserver oncore rtcm104v2 rtcm104v3 sirf skytraq superstar2 tsip tripmate tnt ublox" INPUT_DEVICES="libinput" KERNEL="linux" LCD_DEVICES="bayrad cfontz glk hd44780 lb216 lcdm001 mtxorb text" LUA_SINGLE_TARGET="lua5-1" LUA_TARGETS="lua5-1" OFFICE_IMPLEMENTATION="libreoffice" PHP_TARGETS="php8-1" POSTGRES_TARGETS="postgres15" PYTHON_SINGLE_TARGET="python3_11" PYTHON_TARGETS="python3_11" RUBY_TARGETS="ruby31" VIDEO_CARDS="amdgpu fbdev intel nouveau radeon radeonsi vesa dummy" XTABLES_ADDONS="quota2 psd pknock lscan length2 ipv4options ipp2p iface geoip fuzzy condition tarpit sysrq proto logmark ipmark dhcpmac delude chaos account"

The purpose of the `cpuid2cpuflags` package is to help optimize software compilation for your specific CPU architecture. After installing it, you can run the `cpuid2cpuflags` command, and it will analyze your CPU and suggest optimization flags that you can add to your `make.conf` file.

```
(chroot) livecd / # emerge --ask app-portage/cpuid2cpuflags
(chroot) livecd / # cpuid2cpuflags
(chroot) livecd / # echo "*/* $(cpuid2cpuflags)" >
/etc/portage/package.use/00cpu-flags
```

```
(chroot) livecd / # nano /etc/portage/make.conf
```

```
  GNU nano 7.2
/etc/portage/make.conf
# These settings were set by the catalyst build script that automatically
# built this stage.
# Please consult /usr/share/portage/config/make.conf.example for a more
# detailed example.
COMMON_FLAGS="-O2 -pipe"
CFLAGS="${COMMON_FLAGS}"
CXXFLAGS="${COMMON_FLAGS}"
FCFLAGS="${COMMON_FLAGS}"
FFLAGS="${COMMON_FLAGS}"

# NOTE: This stage was built with the bindist Use flag enabled

# This sets the language of build output to English.
# Please keep this setting intact when reporting bugs.
LC_MESSAGES=C.utf8
MAKEOPTS="-j6 -l4"

GENTOO_MIRRORS="https://mirror.clarkson.edu/gentoo/ \
   http://mirror.clarkson.edu/gentoo/ \
   rsync://mirror.clarkson.edu/gentoo/ \
   http://www.gtlib.gatech.edu/pub/gentoo \
   rsync://rsync.gtlib.gatech.edu/gentoo \
   https://mirrors.mit.edu/gentoo-distfiles/ \
   http://mirrors.mit.edu/gentoo-distfiles/ \
   rsync://mirrors.mit.edu/gentoo-distfiles/ \
   https://gentoo.osuosl.org/ \
   http://gentoo.osuosl.org/ \
   https://mirrors.rit.edu/gentoo/ \
   http://mirrors.rit.edu/gentoo/ \
   ftp://mirrors.rit.edu/gentoo/ \
   rsync://mirrors.rit.edu/gentoo/ \
   https://mirror.servaxnet.com/gentoo/ \
   http://mirror.servaxnet.com/gentoo/ \
   http://gentoo-mirror.flux.utah.edu/"

VIDEO_CARDS="virtio"


(chroot) livecd / # emerge --ask --update --deep --newuse @world

(chroot) livecd / # date 122720432023.34
Wed Dec 27 20:43:34 -00 2023

or use systemsettings5 and then set time and zone from there

livecd /home/gentoo # date
Sun Dec 31 09:10:14 EST 2023
```

Setting time zone

Since my system (i.e., live USB) is booted with openRC and not with systemd, I can't use timedatectl systemd based command to set timezone.

```
(chroot) livecd / # ln -sf /usr/share/zoneinfo/America/New_York /etc/localtime
(chroot) livecd / # date --set="20231231 14:22:34" or date 123114222023.34


(chroot) livecd / # nano /etc/locale.gen
uncommet en_US both lines


(chroot) livecd / # locale-gen
* Generating 3 locales (this might take a while) with 8 jobs
*  (1/3) Generating en_US.ISO-8859-1 ...
[ ok ]
*  (3/3) Generating C.UTF-8 ...
[ ok ]
*  (2/3) Generating en_US.UTF-8 ...
[ ok ]
* Generation complete
* Adding locales to archive ...
[ ok ]
(chroot) livecd / # eselect locale list
Available targets for the LANG variable:
  [1]   C
  [2]   C.utf8
  [3]   POSIX
  [4]   en_US
  [5]   en_US.iso88591
  [6]   en_US.utf8
  [7]   C.UTF8 *
  [ ]   (free form)
(chroot) livecd / # eselect locale set 3
Setting LANG to POSIX ...
Run ". /etc/profile" to update the variable in your shell.
(chroot) livecd / # . /etc/profile
livecd / # env-update && source /etc/profile && export PS1="(chroot) ${PS1}"
>>> Regenerating /etc/ld.so.cache...
(chroot) livecd / #
```

Configuring Linux Kernel

```
(chroot) livecd / # emerge --ask sys-kernel/linux-firmware
Note: We must select license before above command
```

Using genkernel

```
(chroot) livecd / # emerge --ask sys-kernel/gentoo-sources
(chroot) livecd / # eselect kernel list
(chroot) livecd / # eselect kernel set 1
(chroot) livecd / # ls -l /usr/src/linux


(chroot) livecd / # emerge --ask genkernel

(chroot) livecd / # genkernel --mountboot --install all
* Gentoo Linux Genkernel; Version 4.3.6
* Using genkernel configuration from '/etc/genkernel.conf' ...
* Running with options: --mountboot --install all

* Working with Linux kernel 6.1.67-gentoo for x86_64
* Using kernel config file '/usr/share/genkernel/arch/x86_64/generated-config'
...
*
* Note: The version above is subject to change (depends on config and status of
kernel sources).

* kernel: >> Initializing ...
*         >> Running 'make mrproper' ...
*         >> Running 'make oldconfig' ...
*         >> Re-running 'make oldconfig' due to changed kernel options ...
*         >> Kernel version has changed (probably due to config change) since
genkernel start:
*            We are now building Linux kernel 6.1.67-gentoo-x86_64 for x86_64
...
*         >> Compiling 6.1.67-gentoo-x86_64 bzImage ...
*         >> Compiling 6.1.67-gentoo-x86_64 modules ...
*         >> Installing 6.1.67-gentoo-x86_64 modules (and stripping) ...
*         >> Generating module dependency data ...
*         >> Compiling out-of-tree module(s) ...
*         >> Saving config of successful build to
'/etc/kernels/kernel-config-6.1.67-gentoo-x86_64' ...

* initramfs: >> Initializing ...
*         >> Appending devices cpio data ...
*         >> Appending base_layout cpio data ...
*         >> Appending util-linux cpio data ...
*         >> Appending eudev cpio data ...
*         >> Appending auxiliary cpio data ...
*         >> Appending busybox cpio data ...
*         >> Appending modprobed cpio data ...
*         >> Appending modules cpio data ...
*         >> Deduping cpio ...
*         >> Pre-generating initramfs' /etc/ld.so.cache ...
```

```
*          >> Compressing cpio data (.xz) ...
xz: Reduced the number of threads from 6 to 3 to not exceed the memory usage
limit of 3998 MiB

* Kernel compiled successfully!
*
* --no-bootloader set; Skipping bootloader update ...
*
* Required kernel parameter:
*
*       root=/dev/$ROOT
*
* Where $ROOT is the device node for your root partition as the
* one specified in /etc/fstab

* If you require Genkernel's hardware detection features, you MUST
* tell your bootloader to use the provided initramfs file
'/boot/initramfs-6.1.67-gentoo-x86_64.img'.

* WARNING... WARNING... WARNING...
* Additional kernel parameters that *may* be required to boot properly:

* Do NOT report kernel bugs as genkernel bugs unless your bug
* is about the default genkernel configuration...
*
* Make sure you have the latest ~arch genkernel before reporting bugs.



//--install all: Install the kernel, initramfs, and associated modules.
(chroot) livecd / # ls /boot/vmlinu* /boot/initramfs*
/boot/initramfs-6.1.67-gentoo-x86_64.img  /boot/vmlinuz-6.1.67-gentoo-x86_64
(chroot) livecd / # ls /lib/modules
6.1.67-gentoo-x86_64


(chroot) livecd / # emerge -auDN --ask @world
```

Configuring System

```
(chroot) livecd / # blkid
/dev/sr0: BLOCK_SIZE="2048" UUID="2023-12-11-04-35-01-00" LABEL="ISOIMAGE"
TYPE="iso9660" PTTYPE="PMBR"
/dev/loop0: TYPE="squashfs"
/dev/vda1: UUID="0489-FAB7" BLOCK_SIZE="512" TYPE="vfat" PARTLABEL="esp"
PARTUUID="c218b575-d365-47a0-9c
71-d16eef02e7d3"
```

```
/dev/vda2: UUID="642a2410-37fa-437f-aa91-0e34b620d08f" TYPE="swap"
PARTLABEL="swap" PARTUUID="ee1ee52d-d
eae-451d-af23-ab310932dff3"
/dev/vda3: UUID="6fe4884a-f76d-43b5-b66c-4bc133fa1b86" BLOCK_SIZE="512"
TYPE="xfs" PARTLABEL="root" PART
UUID="6f628aba-41c5-4eb4-8024-bbf52e535007"
```

```
(chroot) livecd / # cat /etc/fstab
# /etc/fstab: static file system information.
#
# See the manpage fstab(5) for more information.
#
# NOTE: The root filesystem should have a pass number of either 0 or 1.
#       All other filesystems should have a pass number of 0 or greater than 1.
#
# NOTE: Even though we list ext4 as the type here, it will work with ext2/ext3
#       filesystems.  This just tells the kernel to use the ext4 driver.
#
# NOTE: You can use full paths to devices like /dev/sda3, but it is often
#       more reliable to use filesystem labels or UUIDs. See your filesystem
#       documentation for details on setting a label. To obtain the UUID, use
#       the blkid(8) command.

# <fs>                  <mountpoint>    <type>          <opts>          <dump>
<pass>

#LABEL=boot             /boot           ext4            defaults        1 2
#UUID=58e72203-57d1-4497-81ad-97655bd56494               /               xfs
defaults        0
1
#LABEL=swap             none            swap            sw              0 0
#/dev/cdrom             /mnt/cdrom      auto            noauto,ro       0 0


/dev/vda1   /efi        vfat    defaults    0 2
/dev/vda2   none         swap    sw                  0 0
/dev/vda3   /            xfs    defaults,noatime              0 1

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

```
(chroot) livecd / # echo GentooLinux > /etc/hostname

(chroot) livecd / # emerge --ask net-misc/dhcpcd
```

Although I am on openRC init system during my gentoo installation because I am running from live USB stick. But I chose profile that is sytemd based. So, my system after it is done installation is systemd based.

```
(chroot) livecd / # ls /etc/systemd
coredump.conf  logind.conf  networkd.conf  pstore.conf    sleep.conf
system.conf     user
journald.conf  network      oomd.conf      resolved.conf  system
timesyncd.conf  user.conf
(chroot) livecd / # ls /etc/init.d
cups-browsed  dbus    gpm        iptables           pciparm     pydoc-3.12  sshd
udev-settle
cupsd         dhcpcd  ip6tables  kmod-static-nodes  pydoc-3.11  rsyncd      udev
udev-trigger
```

```
(chroot) livecd / # systemctl enable dhcpcd
Created symlink /etc/systemd/system/multi-user.target.wants/dhcpcd.service ->
/usr/lib/systemd/system/dhcpcd.service.
```

```
(chroot) livecd / # passwd

You can now choose the new password or passphrase.

A valid password should be a mix of upper and lower case letters, digits, and
other characters.  You can use a password containing at least 7 characters
from all of these classes, or a password containing at least 8 characters
from just 3 of these 4 classes.
An upper case letter that begins the password and a digit that ends it do not
count towards the number of character classes used.

A passphrase should be of at least 3 words, 11 to 72 characters long, and
contain enough different characters.

Alternatively, if no one else can see your terminal now, you can pick this as
your password: "Life3Lad$entire".

Enter new password:
Re-type new password:
passwd: password updated successfully
```

```
(chroot) livecd / # systemd-machine-id-setup
Initializing machine ID from random generator.
(chroot) livecd / # systemd-firstboot --prompt
```

```
Welcome to your new installation of Gentoo Linux!

Please configure your system!

-- Press any key to proceed --
> Please enter system keymap name or number (empty to skip, "list" to list
options):
No data entered, skipping.


(chroot) livecd / # systemctl preset-all --preset-mode=enable-only
Created symlink /etc/systemd/system/multi-user.target.wants/machines.target ->
/usr/lib/systemd/system/machines.target.
Created symlink
/etc/systemd/system/sysinit.target.wants/systemd-network-generator.service ->
/usr/lib/systemd/system/systemd-network-g
enerator.service.
Created symlink
/etc/systemd/system/sockets.target.wants/systemd-journald-audit.socket ->
/usr/lib/systemd/system/systemd-journald-audi
t.socket.
Created symlink
/etc/systemd/system/systemd-journald.service.wants/systemd-journald-audit.socke
t -> /usr/lib/systemd/system/systemd-jou
rnald-audit.socket.
Created symlink /etc/systemd/system/sockets.target.wants/systemd-userdbd.socket
-> /usr/lib/systemd/system/systemd-userdbd.socket.
Created symlink
/etc/systemd/system/network-online.target.wants/systemd-networkd-wait-online.se
rvice -> /usr/lib/systemd/system/systemd
-networkd-wait-online.service.
Created symlink /etc/systemd/system/dbus-org.freedesktop.network1.service ->
/usr/lib/systemd/system/systemd-networkd.service.
Created symlink
/etc/systemd/system/multi-user.target.wants/systemd-networkd.service ->
/usr/lib/systemd/system/systemd-networkd.servic
e.
Created symlink
/etc/systemd/system/sockets.target.wants/systemd-networkd.socket ->
/usr/lib/systemd/system/systemd-networkd.socket.
Created symlink /etc/systemd/system/ctrl-alt-del.target ->
/usr/lib/systemd/system/reboot.target.
Created symlink /etc/systemd/system/sysinit.target.wants/systemd-pstore.service
-> /usr/lib/systemd/system/systemd-pstore.service.
Created symlink /etc/systemd/system/dbus-org.freedesktop.timesync1.service ->
/usr/lib/systemd/system/systemd-timesyncd.service.
```

```
Created symlink
/etc/systemd/system/sysinit.target.wants/systemd-timesyncd.service ->
/usr/lib/systemd/system/systemd-timesyncd.service
.
Created symlink /etc/systemd/system/dbus-org.freedesktop.resolve1.service ->
/usr/lib/systemd/system/systemd-resolved.service.
Created symlink
/etc/systemd/system/sysinit.target.wants/systemd-resolved.service ->
/usr/lib/systemd/system/systemd-resolved.service.


(chroot) livecd / # emerge --ask sys-apps/mlocate

(chroot) livecd / # systemctl enable sshd
Created symlink /etc/systemd/system/multi-user.target.wants/sshd.service ->
/usr/lib/systemd/system/sshd.service.


To enable serial console support, run:

(chroot) livecd / # systemctl enable getty@tty1.service

(chroot) livecd / # emerge --ask net-misc/chrony


(chroot) livecd / # systemctl enable chronyd.service
Created symlink /etc/systemd/system/multi-user.target.wants/chronyd.service ->
/usr/lib/systemd/system/chronyd.service.
```

```
(chroot) livecd / # emerge --ask net-misc/dhcpcd

(chroot) livecd / # emerge --ask net-dialup/ppp

(chroot) livecd / # emerge --ask net-wireless/iw net-wireless/wpa_supplicant


(chroot) livecd / # echo 'GRUB_PLATFORMS="efi-64"' >> /etc/portage/make.conf
(chroot) livecd / # emerge --ask sys-boot/grub

(chroot) livecd / # emerge --ask --update --newuse --verbose sys-boot/grub
```

```
(chroot) livecd / # grub-install --target=x86_64-efi --efi-directory=/boot/efi
--bootloader-id=Gentoo
Installing for x86_64-efi platform.
Installation finished. No error reported.
```

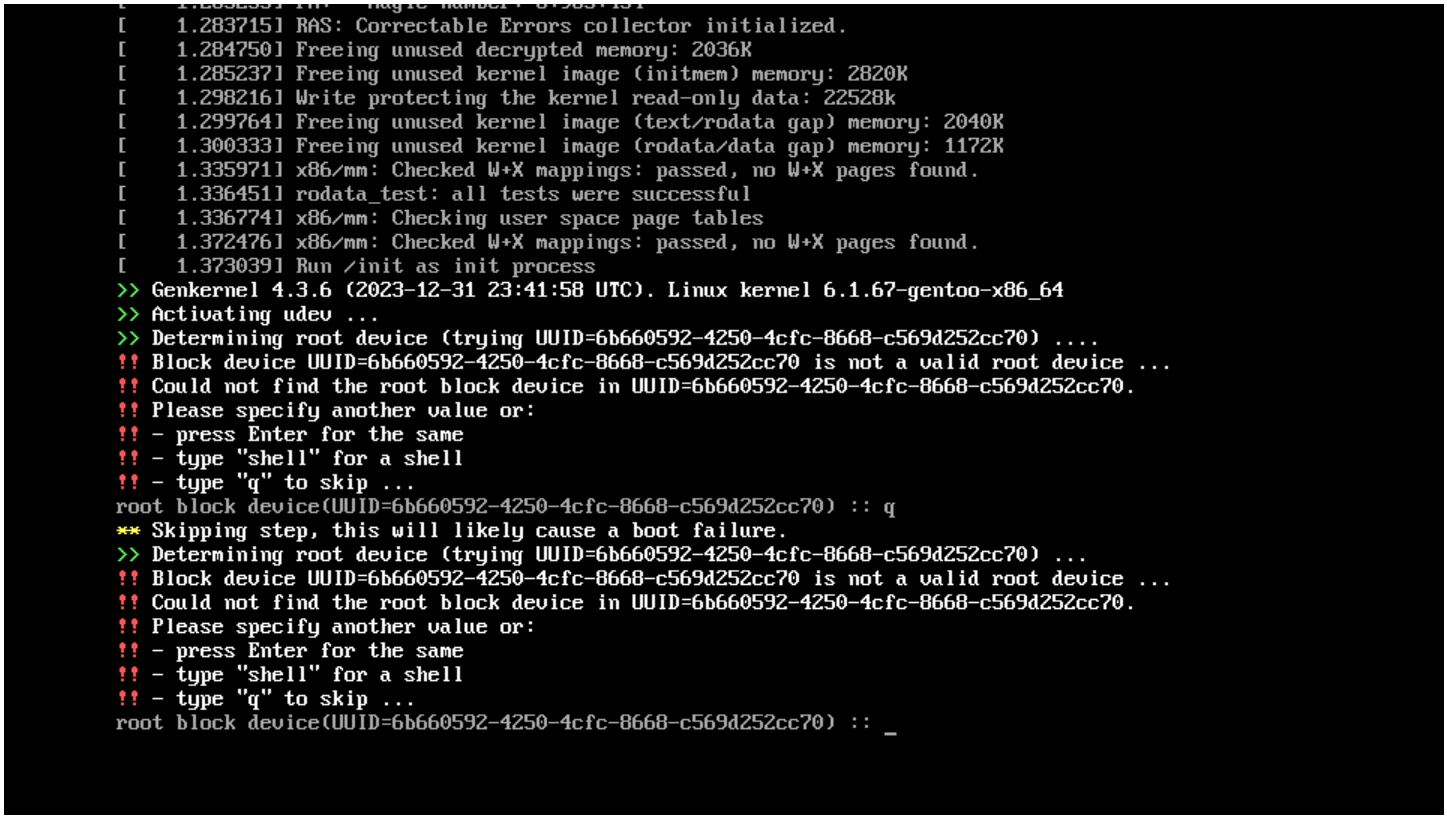
```
(chroot) livecd / # grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.1.67-gentoo-x86_64
Found initrd image: /boot/initramfs-6.1.67-gentoo-x86_64.img
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

Exit

```
(chroot) livecd / # exit
exit
livecd /mnt/gentoo # cd
livecd ~ # umount -l /mnt/gentoo/dev{/shm,/pts,}
livecd ~ # umount -R /mnt/gentoo

livecd /home/gentoo # reboot
```

Boot failure



To fix this I ran into live gentoo USB again.

```
livecd /home/gentoo # mkdir --parents /mnt/gentoo
livecd /home/gentoo # mount /dev/vda3 /mnt/gentoo

livecd /home/gentoo # mount --types proc /proc /mnt/gentoo/proc
livecd /home/gentoo # mount --rbind /sys /mnt/gentoo/sys
livecd /home/gentoo # mount --make-rslave /mnt/gentoo/sys
livecd /home/gentoo # mount --rbind /dev /mnt/gentoo/dev
livecd /home/gentoo # mount --make-rslave /mnt/gentoo/dev
livecd /home/gentoo # mount --bind /run /mnt/gentoo/run
livecd /home/gentoo # mount --make-slave /mnt/gentoo/run

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

(chroot) livecd / # emerge --sync
(chroot) livecd / # emerge --ask sys-fs/dosfstools sys-fs/mtools

(chroot) livecd / # blkid
/dev/vda2: LABEL="swap" UUID="ce36b4b8-135a-4584-bdad-cfeb8eaab61c" TYPE="swap"
PARTLABEL="swap" PARTUUID="723
4df26-21d2-42d3-ba0a-62159c009b59"
```

```
/dev/vda3: LABEL="root" UUID="6b660592-4250-4cfc-8668-c569d252cc70"
BLOCK_SIZE="512" TYPE="xfs" PARTLABEL="roo
t" PARTUUID="a0adbb3c-a30c-4866-a396-8d880ecfecaf"
/dev/vda1: UUID="EBAB-F955" BLOCK_SIZE="512" TYPE="vfat" PARTLABEL="efi"
PARTUUID="41e6b019-a246-4d1c-9d40-7fd
fd6c4ffbc"
/dev/sr0: BLOCK_SIZE="2048" UUID="2023-12-11-04-35-01-00" LABEL="ISOIMAGE"
TYPE="iso9660" PTTYPE="PMBR"
/dev/loop0: TYPE="squashfs"



(chroot) livecd / # cat /etc/fstab
/dev/vda1   /efi         ext4    defaults    0 2
/dev/vda2   none         swap    sw                  0 0
/dev/vda3   /            xfs    defaults,noatime              0 1

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


(chroot) livecd / # cat /etc/default/grub
# Copyright 1999-2020 Gentoo Authors
# Distributed under the terms of the GNU General Public License v2
#
# To populate all changes in this file you need to regenerate your
# grub configuration file afterwards:
#     'grub-mkconfig -o /boot/grub/grub.cfg'
#
# See the grub info page for documentation on possible variables and
# their associated values.

GRUB_DISTRIBUTOR="Gentoo"

# Default menu entry
#GRUB_DEFAULT=0

# Boot the default entry this many seconds after the menu is displayed
#GRUB_TIMEOUT=5
#GRUB_TIMEOUT_STYLE=menu

# Append parameters to the linux kernel command line
# GRUB_CMDLINE_LINUX="root=UUID=6b660592-4250-4cfc-8668-c569d252cc70"
GRUB_CMDLINE_LINUX="root=/dev/vda3"
#
# Examples:
#
# Boot with network interface renaming disabled
# GRUB_CMDLINE_LINUX="net.ifnames=0"
#
```

```
# Boot with systemd instead of sysvinit (openrc)
# GRUB_CMDLINE_LINUX="init=/usr/lib/systemd/systemd"

# Append parameters to the linux kernel command line for non-recovery entries
#GRUB_CMDLINE_LINUX_DEFAULT=""

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal.
# Note that you can use only modes which your graphic card supports via VBE.
# You can see them in real GRUB with the command `vbeinfo'.
#GRUB_GFXMODE=640x480

# Set to 'text' to force the Linux kernel to boot in normal text
# mode, 'keep' to preserve the graphics mode set using
# 'GRUB_GFXMODE', 'WIDTHxHEIGHT'['xDEPTH'] to set a particular
# graphics mode, or a sequence of these separated by commas or
# semicolons to try several modes in sequence.
#GRUB_GFXPAYLOAD_LINUX=

# Path to theme spec txt file.
# The starfield is by default provided with use truetype.
# NOTE: when enabling custom theme, ensure you have required font/etc.
#GRUB_THEME="/boot/grub/themes/starfield/theme.txt"

# Background image used on graphical terminal.
# Can be in various bitmap formats.
#GRUB_BACKGROUND="/boot/grub/mybackground.png"

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to kernel
#GRUB_DISABLE_LINUX_UUID=true

# Comment if you don't want GRUB to pass "root=PARTUUID=xxx" parameter to
kernel
GRUB_DISABLE_LINUX_PARTUUID=false

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY=true

# Uncomment to disable generation of the submenu and put all choices on
# the top-level menu.
# Besides the visual affect of no sub menu, this makes navigation of the
# menu easier for a user who can't see the screen.
#GRUB_DISABLE_SUBMENU=y

# Uncomment to play a tone when the main menu is displayed.
# This is useful, for example, to allow users who can't see the screen
# to know when they can make a choice on the menu.
```

```
#GRUB_INIT_TUNE="60 800 1"

(chroot) livecd / # mkdir /efi
(chroot) livecd / # mount /dev/vda1 /efi
```

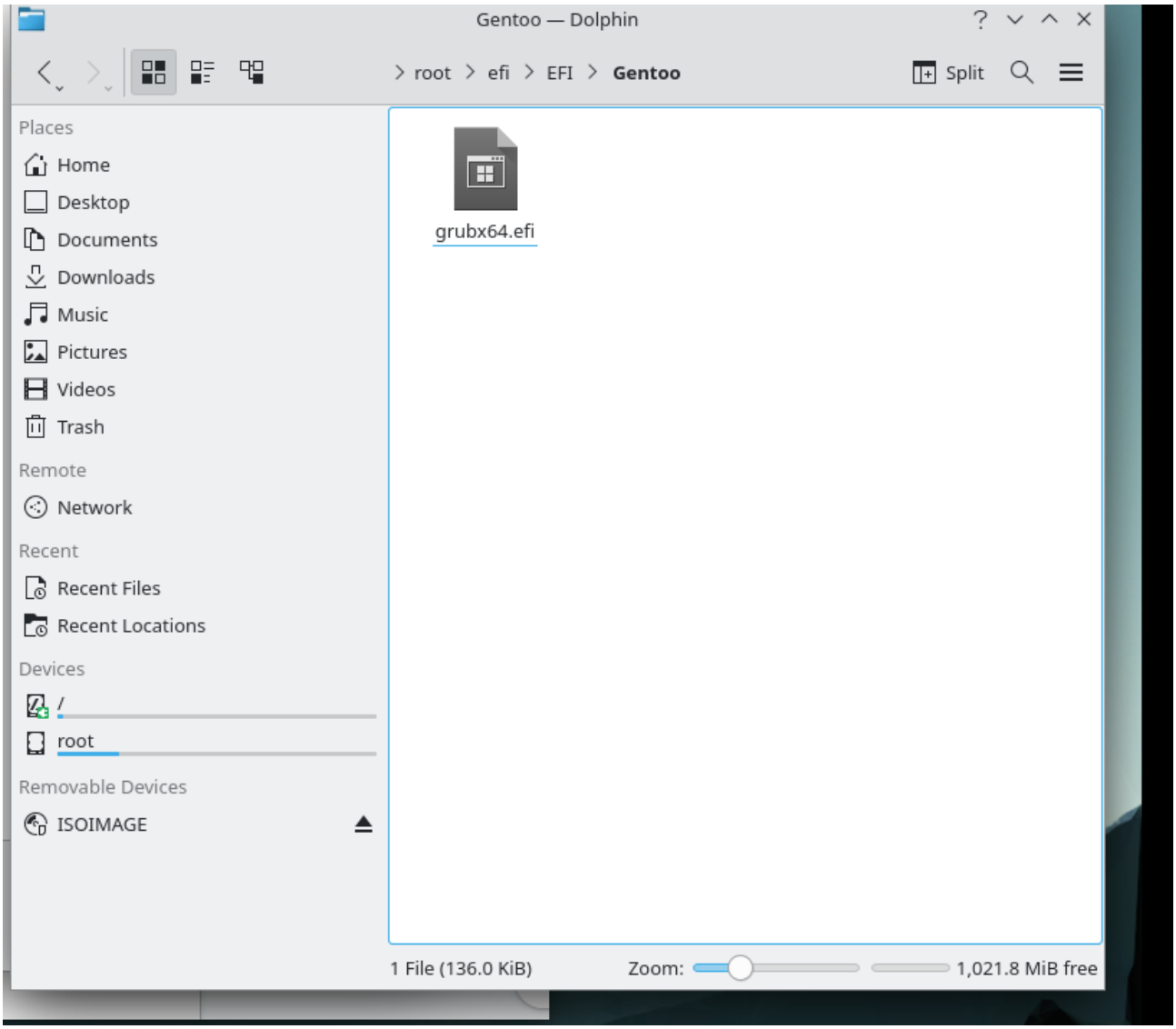
grub-install --target=x86_64-efi `--efi-directory=/efi` --bootloader-id=Gentoo --removable --recheck --no-rs-codes

```
(chroot) livecd / # grub-install --target=x86_64-efi --efi-directory=/efi --bootloader-id=Gentoo
Installing for x86_64-efi platform.
Installation finished. No error reported.



(chroot) livecd / # grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.1.67-gentoo-x86_64
Found initrd image: /boot/initramfs-6.1.67-gentoo-x86_64.img
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done

(chroot) livecd / # efibootmgr
BootCurrent: 0002
Timeout: 0 seconds
BootOrder: 0004,0002,0001,0000,0003
Boot0000* UiApp
FvVol(7cb8bdc9-f8eb-4f34-aaea-3ee4af6516a1)/FvFile(462caa21-7614-4503-836e-8ab6
f4662331)
Boot0001* UEFI Misc Device
PciRoot(0x0)/Pci(0x2,0x3)/Pci(0x0,0x0){auto_created_boot_option}
Boot0002* UEFI QEMU DVD-ROM QM00001
PciRoot(0x0)/Pci(0x1f,0x2)/Sata(0,65535,0){auto_created_boot_option}
Boot0003* EFI Internal Shell
FvVol(7cb8bdc9-f8eb-4f34-aaea-3ee4af6516a1)/FvFile(7c04a583-9e3e-4f1c-ad65-e052
68d0b4d1)
Boot0004* Gentoo
HD(1,GPT,cf80b80f-f61b-4816-bb3f-e1d6549f4201,0x800,0x200000)/File(\EFI\Gentoo\
grubx64.efi)
```



Exit

```
(chroot) livecd / # exit
exit
livecd /mnt/gentoo # cd
livecd ~ # umount -l /mnt/gentoo/dev{/shm,/pts,}
livecd ~ # umount -R /mnt/gentoo

livecd /home/gentoo # reboot
```

But still I get the same issue. While doing blkid from rescueshell I see no block devices. Then I booted from live USB again and chrooted and check if there is virtio devices drivers and notices no drivers installed.

```
ls /lib/modules/$(uname -r)/kernel/drivers/virtio/
```

there is nothing virtio directory.

```
Device Drivers -->
  [*] Block devices -->
    <*> Virtio block driver
  <*> Virtio driver for PCI/PCI Express
```

I now have exact answer:https://wiki.gentoo.org/wiki/User:Flow/Gentoo_as_KVM_guest

genkernel's virtio is not yet fully supported. We need to compile the source again.