Preparing the disks

From Gentoo Wiki

Contents

Introduction to block devices

Block devices

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

To begin, let's look at block devices. SCSI and Serial ATA drives are both labeled under device handles such as: **`/dev/sda`**, **`/dev/sdb`**, **`/dev/sdc`**, etc. On more modern machines, PCI Express based NVMe solid state disks have device handles such as **`/dev/nvme0n1`**, **`/dev/nvme0n2`**, etc.

The following table will help readers determine where to find a certain type of block device on the system:

Type of device	Default device handle	Editorial notes and considerations
IDE, SATA, SAS, SCSI, or USB flash	**`/dev/sda`**	Found on hardware from roughly 2007 until the present, this device handle is perhaps the most commonly used in Linux. These types of devices can be connected via the SATA bus (https://en.wikipedia.org/wiki/Serial_ATA), SCSI (https://en.wikipedia.org/wiki/SCSI), USB (https://en.wikipedia.org/wiki/USB) bus as block storage. As example, the first partition on the first SATA device is called **`/dev/sda1`**.
NVM Express (NVMe)	**`/dev/nvme0n1`**	The latest in solid state technology, NVMe (https://en.wikipedia.org/wiki/NVM_Express) drives are connected to the PCI Express bus and have the fastest transfer block speeds on the market. Systems from around 2014 and newer may have support for NVMe hardware. The first partition on the first NVMe device is called **`/dev/nvme0n1p1`**.
MMC, eMMC, and SD	**`/dev/mmcblk0`**	embedded MMC (https://en.wikipedia.org/wiki/MultiMediaCard#eMMC) devices, SD cards, and other types of memory cards can be useful for data storage. That said, many systems may not permit booting from these types of devices. It is suggested to **not** use these devices for active Linux installations; rather consider using them to transfer files, which is their typical design intention. Alternatively this storage type could be useful for short-term file backups or snapshots.

The block devices above represent an abstract interface to the disk. User programs can use these block devices to interact with the disk without worrying about whether the drives are SATA, SCSI, or something else. The program can simply address the storage on the disk as a bunch of contiguous, randomly-accessible 4096-byte (4K) blocks.

Partition tables

Although it is theoretically possible to use a raw, unpartitioned disk to house a Linux system (when creating a btrfs RAID for example), this is almost never done in practice. Instead, disk block devices are split up into smaller, more manageable block devices. On **`amd64`** systems, these are called partitions. There are currently two standard partitioning technologies in use: MBR (sometimes also called DOS disklabel) and GPT; these are tied to the two boot process types: legacy BIOS boot and UEFI.

GUID Partition Table (GPT)

The *GUID Partition Table (GPT)* setup (also called GPT disklabel) uses 64-bit identifiers for the partitions. The location in which it stores the partition information is much bigger than the 512 bytes of the MBR partition table (DOS disklabel), which means there is practically no limit on the number of partitions for a GPT disk. Also, the maximum partition size is much larger (almost 8 ZiB -- yes, zebibytes).

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 DOS disklabel.

GPT also takes advantage of checksumming and redundancy. It carries CRC32 checksums to detect errors in the header and partition tables and has a backup GPT at the end of the disk. This backup table can be used to recover damage of the primary GPT near the beginning of the disk.

Important

There are a few caveats regarding GPT:

- Using GPT on a BIOS-based computer works, but the user won't be able to dual-boot with a Microsoft Windows operating system, since Microsoft Windows refuses to boot from a GPT partition when in BIOS mode.
- Some buggy (old) motherboard firmware configured to boot in BIOS/CSM/legacy mode might also have problems with booting from GPT labeled disks.

Master boot record (MBR) or DOS boot sector

The *Master boot record (https://en.wikipedia.org/wiki/Master_boot_record)* boot sector (also called DOS boot sector, DOS disklabel, and - more recently, in contrast to GPT/UEFI setups - legacy BIOS boot) was first introduced in 1983 with PC DOS 2.x. MBR 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**).

In order to support more partitions, one of the primary partitions in the MBR can be marked as an *extended* partition. This partition can then contain additional logical partitions (partitions within a partition).

Important

Although still supported by most motherboard manufacturers, MBR boot sectors and their associated partitioning limitations are considered legacy. Unless working with hardware that is pre-2010, it best to partition a disk with GUID Partition Table (https://en.wikipedia.org/wiki/GUID_Partition_Table). Readers who must proceed with setup type should knowingly acknowledge the following information:

- Most post-2010 motherboards consider using MBR boot sectors a legacy (supported, but not ideal) boot mode.
- Due to using 32-bit identifiers, partition tables in the MBR cannot address storage space that is larger than 2 TiBs in size.
- Unless an extended partition is created, MBR supports a maximum of four partitions.
- This setup does not provide a backup boot sector, so if something overwrites the partition table, all partition information will be lost.

That said, MBR and legacy BIOS boot may still used in virtualized cloud environments such as AWS.

The Handbook authors suggest using GPT whenever possible for Gentoo installations.

Advanced storage

The official Gentoo boot media provides support for Logical Volume Manager (LVM) (/wiki/LVM). LVM can combine *physical volumes* such as partitions or disks into *volume groups*. Volume groups are more flexible than partitions and can be used to define RAID groups or caches on fast SSDs for slow HDs. Although usage is not covered in the handbook, LVM is fully supported in Gentoo.

Default partitioning scheme

Throughout the remainder of the handbook, we will discuss and explain two cases:

1. UEFI firmware with GUID Partition Table (GPT) disk.
2. MBR DOS/legacy BIOS firmware with a MBR partition table disk.

While it is possible to mix and match boot types with certain motherboard firmware, mixing goes beyond the intention of the handbook. As previously stated, it is strongly recommended for installations on modern hardware to use UEFI boot with a GPT disklabel disk.

The following partitioning scheme will be used as a simple example layout.

Important

The first row of the following table contains exclusive information for ***either*** a GPT disklabel ***or*** a MBR DOS/legacy BIOS disklabel. When in doubt, proceed with GPT, since **amd64** machines manufactured after the year 2010 generally support UEFI firmware and GPT boot sector.

Partition	Filesystem	Size	Description
/dev/sda1	fat32	1 GiB	EFI System Partition details.
	xfs		MBR DOS/legacy BIOS boot partition details.
/dev/sda2	linux-swap	RAM size * 2	Swap partition details.
/dev/sda3	xfs	Remainder of the disk	Root partition details.

If this suffices as information, the advanced reader can directly skip ahead to the actual partitioning.

Both **`fdisk`** and **`parted`** are partitioning utilities included within the official Gentoo live image environments. **`fdisk`** is well known, stable, and handles both MBR and GPT disks. **`parted`** was one of the first Linux block device management utilities to support GPT partitions. It can be used as an alternative to **`fdisk`** if the reader prefers, however the handbook will only provide instructions for **`fdisk`**, since it is commonly available on most Linux environments.

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

Designing a partition scheme

How many partitions and how big?

The design of disk partition layout is highly dependent on the demands of the system and the file system(s) applied to the device. If there are lots of users, then it is advised to have **`/home`** on a separate partition which will increase security and make backups and other types of maintenance easier. If Gentoo is being installed to perform as a mail server, then **`/var`** should be a separate partition as all mails are stored inside the **`/var`** directory. Game servers may have a separate **`/opt`** partition since most gaming server software is installed therein. The reason for these recommendations is similar to the **`/home`** directory: security, backups, and maintenance.

In most situations on Gentoo, **`/usr`** and **`/var`** should be kept relatively large in size. **`/usr`** hosts the majority of applications available on the system and the Linux kernel sources (under **`/usr/src`**). By default, **`/var`** hosts the Gentoo ebuild repository (located at **`/var/db/repos/gentoo`**) which, depending on the file system, generally consumes around 650 MiB of disk space. This space estimate *excludes* the **`/var/cache/distfiles`** and **`/var/cache/binpkgs`** directories, which will gradually fill with source files and (optionally) binary packages respectively as they are added to the system.

How many partitions and how big very much depends on considering the trade-offs and choosing the best option for the circumstance. 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 realized 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 certain disadvantages as well:

- If not configured properly, the system might have lots of free space on one partition and little free space on another.
- A separate partition for **`/usr/`** may require the administrator to boot with an initramfs to mount the partition before other boot scripts start. Since the generation and maintenance of an initramfs is beyond the scope of this handbook, **we recommend that newcomers do not use a separate partition for `/usr/`**.
- There is also a 15-partition limit for SCSI and SATA unless the disk uses GPT labels.

Note
Installations that intend to use systemd as the service and init system must have the **`/usr`** directory available at boot, either as part of the root filesystem or mounted via an initramfs.

What about swap space?

Recommendations for swap space size

RAM size	Suspend support?	Hibernation support?
2 GB or less	2 * RAM	3 * RAM
2 to 8 GB	RAM amount	2 * RAM
8 to 64 GB	8 GB minimum, 16 maximum	1.5 * RAM
64 GB or greater	8 GB minimum	Hibernation not recommended!

There is no perfect value for swap space size. The purpose of the space is to provide disk storage to the kernel when internal dynamic 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), which will free memory in RAM for the current task. Of course, if the pages swapped to disk are suddenly needed, they will need to be put back in memory (page-in) which will take considerably longer than reading from RAM (as disks are very slow compared to internal memory).

When a system is not going to run memory intensive applications or has lots of RAM available, then it probably does not need much swap space. However do note in case of hibernation that swap space is used to store *the entire contents of memory* (likely on desktop and laptop systems rather than on server systems). If the system requires support for hibernation, then swap space larger than or equal to the amount of memory is necessary.

As a general rule for RAM amounts less than 4 GB, the swap space size is recommended to be twice the internal memory (RAM). For systems with multiple hard disks, it is wise to create one swap partition on each disk so that they can be utilized for parallel read/write operations. The faster a disk can swap, the faster the system will run when data in swap space must be accessed. When choosing between rotational and solid state disks, it is better for performance to put swap on the solid state hardware.

It is worth noting that swap *files* can be used as an alternative to swap *partitions*; this is mostly helpful for systems with very limited disk space.

What is the EFI System Partition (ESP)?

When installing Gentoo on a system that uses UEFI to boot the operating system (instead of BIOS) it is essential that an EFI System Partition (ESP) is created. The instructions below contain the necessary pointers to correctly handle this operation. **The EFI system partition is not required when booting in BIOS/Legacy mode.**

The ESP *must* be a FAT variant (sometimes shown as *vfat* on Linux systems). The official UEFI specification (http://www.uefi.org/sites/default/files/resources/UEFI%202_5.pdf) denotes FAT12, 16, or 32 filesystems will be recognized by the UEFI firmware, although FAT32 is recommended for the ESP. After partitioning, format the ESP accordingly:

```
root # mkfs.fat -F 32 /dev/sda1
```

Important

If the ESP is not formatted with a FAT variant, the system's UEFI firmware will not find the bootloader (or Linux kernel) and will most likely be unable to boot 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. It only needed when a disk is formatted with a GPT disklabel, but the system's firmware will be booting via GRUB2 in legacy BIOS/MBR DOS boot mode. **It is *not required* when booting in EFI/UEFI mode, and also *not required* when using a MBR/Legacy DOS disklabel.** A *BIOS boot partition* will not be used in this guide.

Partitioning the disk with GPT for UEFI

The following parts explain how to create an example partition layout for a single GPT disk device which will conform to the UEFI Specification and Discoverable Partitions Specification (DPS). DPS is a specification provided as part of the Linux Userspace API (UAPI) Group Specification and is recommended, but entirely optional. The specifications are implemented using the **fdisk** utility, which is part of the sys-apps/util-linux (https://packages.gentoo.org/packages/sys-apps/util-linux) package.

The table provides a recommended defaults for a trivial Gentoo installation. Additional partitions can be added according to personal preference or system design goals.

Device path (sysfs)	**Mount point**	File system	**DPS UUID (*Type-UUID*)**	**Description**
/dev/sda1	**/efi**	vfat	c12a7328-f81f-11d2-ba4b-00a0c93ec93b	EFI system partition (ESP) details.
/dev/sda2	N/A.	swap	0657fd6d-a4ab-43c4-84e5-0933c84b4f4f	Swap partition details.
/dev/sda3	**/**	xfs	4f68bce3-e8cd-4db1-96e7-fbcaf984b709	Root partition details.

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
```

Use the p key to display the disk's current partition configuration:

```
Command (m for help): p
```

```
Disk /dev/sda: 931.51 GiB, 1000204886016 bytes, 1953525168 sectors
Disk model: HGST HTS721010A9
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: 3E56EE74-0571-462B-A992-9872E3855D75

Device        Start        End    Sectors   Size Type
/dev/sda1      2048    2099199    2097152     1G EFI System
/dev/sda2   2099200   10487807    8388608     4G Linux swap
/dev/sda3  10487808 1953523711 1943035904 926.5G Linux root (x86-64)
```

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

Creating a new disklabel / removing all partitions

Pressing the `g` key will instantly remove all existing disk partitions and create a new GPT disklabel:

Command (m for help): g

```
Created a new GPT disklabel (GUID: 3E56EE74-0571-462B-A992-9872E3855D75).
```

Alternatively, to keep an existing GPT disklabel (see the output of `p` above), consider removing the existing partitions one by one from the disk. Press `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, press `q` immediately and hit `Enter` and the partition will not be deleted.

Repeatedly press `p` to print out a partition listing and then press `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: 931.51 GiB, 1000204886016 bytes, 1953525168 sectors
Disk model: HGST HTS721010A9
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: 3E56EE74-0571-462B-A992-9872E3855D75
```

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

Creating the EFI System Partition (ESP)

Note
A smaller ESP is possible but not recommended, especially given it may be shared with other OSes.

First create a small EFI system partition, which will also be mounted as **/efi**. Type `n` to create a new partition, followed by `1` to select the first partition. When prompted for the first sector, make sure it starts from 2048 (which may be needed for the boot loader) and hit `Enter`. When prompted for the last sector, type +1G to create a partition 1 gibibyte in size:

Command (m for help): n

```
Partition number (1-128, default 1): 1
First sector (2048-1953525134, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-1953525134, default 1953523711): +1G

Created a new partition 1 of type 'Linux filesystem' and of size 1 GiB.
Partition #1 contains a vfat signature.

Do you want to remove the signature? [Y]es/[N]o: Y
The signature will be removed by a write command.
```

Mark the partition as an EFI system partition:

Command (m for help): t

```
Selected partition 1
Partition type or alias (type L to list all): 1
Changed type of partition 'Linux filesystem' to 'EFI System'.
```

Creating the swap partition

Next, to create the swap partition, press `n` to create a new partition, then press `2` to create the second partition, **/dev/sda2**. When prompted for the first sector, hit `Enter`. When prompted for the last sector, type +4G (or any other size needed for the swap space) to create a partition 4 GiB in size.

Command (m for help): n

```
Partition number (2-128, default 2):
First sector (2099200-1953525134, default 2099200):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2099200-1953525134, default 1953523711): +4G

Created a new partition 2 of type 'Linux filesystem' and of size 4 GiB.
```

After this, press `t` to set the partition type, `2` to select the partition just created and then type in *19* to set the partition type to "Linux Swap".

Command (m for help): t

```
Partition number (1,2, default 2): 2
Partition type or alias (type L to list all): 19

Changed type of partition 'Linux filesystem' to 'Linux swap'.
```

Creating the root partition

Finally, to create the root partition, press `n` to create a new partition, and then press `3` to create the third partition: **`/dev/sda3`**. When prompted for the first sector, press `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.

Command (m for help): n

```
Partition number (3-128, default 3): 3
First sector (10487808-1953525134, default 10487808):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (10487808-1953525134, default 1953523711):

Created a new partition 3 of type 'Linux filesystem' and of size 926.5 GiB..
```

Note
Setting the root partition's type to "Linux root (x86-64)" is not required and the system will function normally if it is set to the "Linux filesystem" type. This filesystem type is only necessary for cases where a bootloader that supports it (i.e. systemd-boot) is used and a fstab file is not wanted.

After creating the root partition, press `t` to set the partition type, `3` to select the partition just created, and then type in *23* to set the partition type to "Linux Root (x86-64)".

Command(m for help): t

```
Partition number (1-3, default 3): 3
Partition type or alias (type L to list all): 23

Changed type of partition 'Linux filesystem' to 'Linux root (x86-64)'
```

After completing these steps, pressing `p` should display a partition table that looks similar to the following:

Command (m for help): p

```
Disk /dev/sda: 931.51 GiB, 1000204886016 bytes, 1953525168 sectors
Disk model: HGST HTS721010A9
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: 3E56EE74-0571-462B-A992-9872E3855D75

Device        Start        End    Sectors   Size Type
/dev/sda1      2048    2099199    2097152     1G EFI System
/dev/sda2   2099200   10487807    8388608     4G Linux swap
/dev/sda3  10487808 1953523711 1943035904 926.5G Linux root (x86-64)

Filesystem/RAID signature on partition 1 will be wiped.
```

Saving the partition layout

Press `w` to save the partition layout and exit the **`fdisk`** utility:

Command (m for help): w

```
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

With partitions now available, the next installation step is to fill them with filesystems.

Partitioning the disk with MBR for BIOS / legacy boot

The following table provides a recommended partition layout for a trivial MBR DOS / legacy BIOS boot installation. Additional partitions can be added according to personal preference or system design goals.

Device path (sysfs)	**Mount point**	File system	**DPS UUID (*PARTUUID*)**	**Description**
`/dev/sda1`	**`/boot`**	xfs	N/A	MBR DOS / legacy BIOS boot partition details.
`/dev/sda2`	N/A.	swap	0657fd6d-a4ab-43c4-84e5-0933c84b4f4f	Swap partition details.
`/dev/sda3`	**`/`**	xfs	4f68bce3-e8cd-4db1-96e7-fbcaf984b709	Root partition details.

Change the partition layout according to personal preference.

Viewing the current partition layout

Fire up **`fdisk`** against the disk (in our example, we use **`/dev/sda`**):

```
root # fdisk /dev/sda
```

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

```
Command (m for help): p
```

```
Disk /dev/sda: 931.51 GiB, 1000204886016 bytes, 1953525168 sectors
Disk model: HGST HTS721010A9
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: dos
Disk identifier: 0xf163b576

Device     Boot    Start        End    Sectors   Size Id Type
/dev/sda1  *        2048    2099199    2097152     1G 83 Linux
/dev/sda2        2099200   10487807    8388608     4G 82 Linux swap / Solaris
/dev/sda3       10487808 1953525167 1943037360 926.5G 83 Linux
```

This particular disk was until now configured to house two Linux filesystems (each with a corresponding partition listed as "Linux") as well as a swap partition (listed as "Linux swap"), using a GPT table.

Creating a new disklabel / removing all partitions

Pressing `o` will instantly remove all existing disk partitions and create a new MBR disklabel (also named DOS disklabel):

```
Command (m for help): o
```

```
Created a new DOS disklabel with disk identifier 0xf163b576.
The device contains 'gpt' signature and it will be removed by a write command. See fdisk(8) man page and --
wipe option for more details.
```

Alternatively, to keep an existing DOS disklabel (see the output of `p` above), consider removing the existing partitions one by one from the disk. Press `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 press `p` to print out a partition listing and then press `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: 931.51 GiB, 1000204886016 bytes, 1953525168 sectors
Disk model: HGST HTS721010A9
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: dos
Disk identifier: 0xf163b576
```

The disk is now ready to create new partitions.

Creating the boot partition

First, create a small partition which will be mounted as **/boot**. Press `n` to create a new partition, followed by `p` for a *primary* partition and `1` to select the first primary partition. When prompted for the first sector, make sure it starts from 2048 (which may be needed for the boot loader) and press `Enter`. When prompted for the last sector, type +1G to create a partition 1 GB in size:

Command (m for help): n

```
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-1953525167, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-1953525167, default 1953525167): +1G

Created a new partition 1 of type 'Linux' and of size 1 GiB.
```

Mark the partition as bootable by pressing the `a` key and pressing `Enter`:

Command (m for help): a

```
Selected partition 1
The bootable flag on partition 1 is enabled now.
```

Note: if more than one partition is available on the disk, then the partition to be flagged as bootable will have to be selected.

Creating the swap partition

Next, to create the swap partition, press `n` to create a new partition, then `p`, then type `2` to create the second primary partition, **/dev/sda2**. When prompted for the first sector, press `Enter`. When prompted for the last sector, type +4G (or any other size needed for the swap space) to create a partition 4GB in size.

Command (m for help): n

```
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): 2
First sector (2099200-1953525167, default 2099200):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2099200-1953525167, default 1953525167): +4G

Created a new partition 2 of type 'Linux' and of size 4 GiB.
```

After all this is done, press `t` to set the partition type, `2` to select the partition just created and then type in *82* to set the partition type to "Linux Swap".

Command (m for help): t

```
Partition number (1,2, default 2): 2
Hex code (type L to list all codes): 82

Changed type of partition 'Linux' to 'Linux swap / Solaris'.
```

Creating the root partition

Finally, to create the root partition, press `n` to create a new partition. Then press `p` and `3` to create the third primary partition, **/dev/sda3**. 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:

Command (m for help): n

```
Partition type
   p   primary (2 primary, 0 extended, 2 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (3,4, default 3): 3
First sector (10487808-1953525167, default 10487808):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (10487808-1953525167, default 1953525167):

Created a new partition 3 of type 'Linux' and of size 926.5 GiB.
```

After completing these steps, pressing `p` should display a partition table that looks similar to this:

Command (m for help): p

```
Disk /dev/sda: 931.51 GiB, 1000204886016 bytes, 1953525168 sectors
Disk model: HGST HTS721010A9
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: dos
Disk identifier: 0xf163b576

Device     Boot    Start        End    Sectors   Size Id Type
/dev/sda1  *        2048    2099199    2097152     1G 83 Linux
/dev/sda2        2099200   10487807    8388608     4G 82 Linux swap / Solaris
/dev/sda3       10487808 1953525167 1943037360 926.5G 83 Linux
```

Saving the partition layout

Press `w` to write the partition layout and exit **fdisk**:

Command (m for help): w

```
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Now it is time to put filesystems on the partitions.

Creating file systems

Warning
When using SSD or NVMe drive, it is wise to check for firmware upgrades. Some Intel SSDs in particular (600p and 6000p) require a firmware upgrade for possible data corruption (https://bugzilla.redhat.com/show_bug.cgi?id=1402533) induced by XFS I/O usage patterns. The problem is at the firmware level and not any fault of the XFS filesystem. The **smartctl** utility can help check the device model and firmware version.

Introduction

Now that the partitions have been 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 (/wiki/Handbook:AMD64/Installation/Disks#Applying_a_filesystem_to_a_partition). The others should read on to learn about the available filesystems...

Filesystems

Linux supports several dozen filesystems, although many of them are only wise to deploy for specific purposes. Only certain filesystems may be 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. **XFS is the recommended all-purpose, all-platform filesystem.** The below is a non-exhaustive list:

btrfs (/wiki/Btrfs)
Newer generation filesystem. Provides advanced features like snapshotting, self-healing through checksums, transparent compression, subvolumes, and integrated RAID. Kernels prior to 5.4.y are not guaranteed to be safe to use with btrfs in production because fixes for serious issues are only present in the more recent releases of the LTS kernel branches. RAID 5/6 and quota groups unsafe on all versions of btrfs.

ext4 (/wiki/Ext4)
Ext4 is a reliable, all-purpose all-platform filesystem, although it lacks modern features like reflinks.

f2fs (/wiki/F2fs)
The Flash-Friendly File System was originally created by Samsung for the use with NAND flash memory. It is a decent choice when installing Gentoo to microSD cards, USB drives, or other flash-based storage devices.

XFS (/wiki/XFS)
Filesystem with metadata journaling which comes with a robust feature-set and is optimized for scalability. It has been continuously upgraded to include modern features. The only downside is that XFS partitions cannot yet be shrunk, although this is being worked on. XFS notably supports reflinks and Copy on Write (CoW) which is particularly helpful on Gentoo systems because of the amount of compiles users complete. XFS is the recommended modern all-purpose all-platform filesystem. Requires a partition to be at least 300MB.

VFAT (/wiki/VFAT)
Also known as FAT32, is supported by Linux but does not support standard UNIX permission settings. It is mostly used for interoperability/interchange with other operating systems (Microsoft Windows or Apple's macOS) but is also a necessity for some system bootloader firmware (like UEFI). Users of UEFI systems will need an EFI System Partition (/wiki/EFI_System_Partition) formatted with VFAT in order to boot.

NTFS (/wiki/NTFS)
This "New Technology" filesystem is the flagship filesystem of Microsoft Windows since Windows NT 3.1. Similarly to VFAT, it does not store UNIX permission settings or extended attributes necessary for BSD or Linux to function properly, therefore it should not be used as a root filesystem for most cases. It should *only* be used for interoperability or data interchange with Microsoft Windows systems (note the emphasis on *only*).

More extensive information on filesystems can be found in the community maintained Filesystem article (/wiki/Filesystem).

Applying a filesystem to a partition

Note
Please make sure to emerge the relevant user space utilities package for the chosen filesystem before rebooting. There will be a reminder to do so near the end of the installation process.

To create a filesystem on a partition or volume, there are user space utilities available for each possible filesystem. Click the filesystem's name in the table below for additional information on each filesystem:

Filesystem	Creation command	Within the live environment?	Package
btrfs (/wiki/Btrfs)	**`mkfs.btrfs`**	Yes	`sys-fs/btrfs-progs` (https://packages.gentoo.org/packages/sys-fs/btrfs-progs)
ext4 (/wiki/Ext4)	**`mkfs.ext4`**	Yes	`sys-fs/e2fsprogs` (https://packages.gentoo.org/packages/sys-fs/e2fsprogs)
f2fs (/wiki/F2FS)	**`mkfs.f2fs`**	Yes	`sys-fs/f2fs-tools` (https://packages.gentoo.org/packages/sys-fs/f2fs-tools)
xfs (/wiki/XFS)	**`mkfs.xfs`**	Yes	`sys-fs/xfsprogs` (https://packages.gentoo.org/packages/sys-fs/xfsprogs)
vfat (/wiki/FAT)	**`mkfs.vfat`**	Yes	`sys-fs/dosfstools` (https://packages.gentoo.org/packages/sys-fs/dosfstools)
NTFS (/wiki/NTFS)	**`mkfs.ntfs`**	Yes	`sys-fs/ntfs3g` (https://packages.gentoo.org/packages/sys-fs/ntfs3g)

Important
The handbook recommends new partitions as part of the installation process, but it is important to note running any **`mkfs`** command will erase any data contained within the partition. When necessary, ensure any data that exists within is appropriately backed up before creating a new filesystem.

For instance, to have the root partition (**`/dev/sda3`**) as xfs as used in the example partition structure, the following commands would be used:

```
root # mkfs.xfs /dev/sda3
```

EFI system partition filesystem

The EFI system partition (**`/dev/sda1`**) must be formatted as FAT32:

```
root # mkfs.vfat -F 32 /dev/sda1
```

Legacy BIOS boot partition filesystem

Systems booting via legacy BIOS with a MBR/DOS disklabel can use any filesystem format supported by the bootloader.

For example, to format with XFS:

```
root # mkfs.xfs /dev/sda1
```

Small ext4 partitions

When using the ext4 filesystem on a small partition (less than 8 GiB), the filesystem should be created with the proper options to reserve enough inodes. This can specified using the `-T small` option:

```
root # mkfs.ext4 -T small /dev/<device>
```

Doing so will quadruple the number of inodes for a given filesystem, since its "bytes-per-inode" reduces from one every 16kB to one every 4kB.

Activating the swap partition

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

```
root # mkswap /dev/sda2
```

To activate the swap partition, use **`swapon`**:

```
root # swapon /dev/sda2
```

This 'activation' step is only necessary because the swap partition is newly created within the live environment. Once the system has been rebooted, as long as the swap partition is properly defined within fstab or other mount mechanism, swap space will activate automatically.

Mounting the root partition

Note
Installations which were previously started, but did not finish the installation process can resume the installation from this point in the handbook. Use this link as the permalink: Resumed installations start here (/wiki/Handbook:AMD64/Installation /Disks#Resumed_installations_start_here).

Certain live environments may be missing the suggested mount point for Gentoo's root partition (**/mnt/gentoo**), or mount points for additional partitions created in the partitioning section:

```
root # mkdir --parents /mnt/gentoo
```

Continue creating additional mount points necessary for any additional (custom) partition(s) created during previous steps by using the **mkdir** command.

With mount points created, it is time to make the partitions accessible via **mount** command.

Mount the root partition:

```
root # mount /dev/sda3 /mnt/gentoo
```

For EFI installs only, the ESP should be mounted under the root partition location:

```
root # mkdir --parents /mnt/gentoo/efi
```

Continue mounting additional (custom) partitions as necessary using the **mount** command.

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 pseudo-filesystems will be mounted. But first the Gentoo stage file (/wiki/Handbook:AMD64/Installation/Stage) must be extracted.

← Configuring the network (/wiki/Handbook:AMD64/Installation/Networking) | Home (/wiki/Handbook:AMD64) | Installing the stage file → (/wiki/Handbook:AMD6

Retrieved from "https://wiki.gentoo.org/index.php?title=Handbook:AMD64/Installation/Disks&oldid=212414 (https://wiki.gentoo.org /index.php?title=Handbook:AMD64/Installation/Disks&oldid=212414)"

- This page was last edited on 2 January 2015, at 00:03.
- Privacy policy (/wiki/Gentoo_Wiki:Privacy_policy)
- About Gentoo Wiki (/wiki/Gentoo_Wiki:About)
- Disclaimers (/wiki/Gentoo_Wiki:General_disclaimer)