Partitions and Filesystems

==========

Partitions

What is a partition?

Partitions allow you to divide a single physical disk into multiple, isolated sections. Each partition can be managed independently

Partition types in Linux

`MBR` - Master Boot Record

The Master Boot Record (MBR) is a special type of boot sector at the very beginning of a disk. The MBR contains important information about the disk's partitions and the filesystem, as well as executable code necessary to boot an operating system. MBR permits for up to 4 partitions on a storage device and also has limitations in the size of disks it can partition, as well as the size of partitions that can be created.

> TLDR; MBR can partition a drive into 4 partitions. Not ideal for large drives. Old standard that's being phased out in favour of GPT.

`GPT` - GUID Partition Table

The GUID Partition Table (GPT) is a modern standard for the layout of partition tables on a physical storage device. GPT is part of the Unified Extensible Firmware Interface (UEFI) standard, which is designed to replace the older BIOS firmware interface used by PCs.

Benefits of GPT include:

Larger Disk and Partition Support - GPT allows for a virtually unlimited number of partitions. GPT can support disks larger than 2 terabytes (TB), up to 9.4 zettabytes (ZB).

Redundancy - GPT stores a primary partition table at the beginning of the disk and a backup partition table at the end of the disk.

>

> TLDR; GPT is more modern, handles larger disks & partitions, and also has redundancy features.

Configuration & management of partitions

---

fdisk

fdisk (format disk) is a command-line utility that can be used for making changes to MBR disk partitions.

> fdisk is primarily designed for use with MBR partitioned disks. Check the disk partitioning your disk is using before making changes (fdisk -l /dev/devicename). If your disk is GPT, see here.

View current partitions and partitioning standard:

```
fdisk -l /dev/devicename
```

To enter into the fdisk utility to manage partitions:

```
fdisk /dev/devicename
```

We then have the following options available:

`n`	Create a new partition
`d`	Delete a partition
`i`	print current boot record

> **Example;**
>
> Let's say we have a single drive, and want to create a 10GB partition. This is an MBR drive.
>
> Firstly, enter into the fdisk utility:

> `fdisk /dev/diskname`
>
> From there, the following options can be used to add a new partition with a size of 10GB.
>
> Command (m for help): n
> Partition type:
> p primary
> e extended
> Select (default p): p
> Partition number (4-128, default 4): #left at default value
> First sector (34-16777182, default 16775168): #left at default value (typically)
> Last sector, +/-sectors or +/-size{K,M,G,T,P} (16775168-16777182, default 16777182):
> +10G
>
> NOTE: fdisk shouldn't really be used for partitioning GPT drives.
> NOTE: Leaving the last sector blank will automatically cause the remainder of space on the drive to be allocated.

---

gdisk

gdisk is the GPT equivalent of fdisk, meaning that it's designed specifically for partitioning disks using GPT formatting.

> gdisk is primarily designed for use with GPT partitioned disks. Check the disk partitioning your disk is using before making changes (fdisk -l /dev/devicename). If your disk is MBR, see here.

View current partitions and partitioning standard:

```
gdisk -l /dev/devicename
```

To enter into the gdisk utility to manage partitions:

```
gdisk /dev/devicename
```

We then have the following options available:

`n`	Create a new partition
`d`	Delete a partition

`i`	print current boot record

> **Example;**
>
> Let's say we have a single drive, and want to create a 10GB partition. This is a GPT drive.
>
> Firstly, enter into the fdisk utility:
> `gdisk /dev/diskname`
>
> From there, the following options can be used to add a new partition with a size of 10GB.
>
> Command (? for help): n
> Partition number (5-128, default 5):
> First sector (34-2047, default = 34) or {+-}size{KMGTP}:
> Last sector (34-2047, default = 2047) or {+-}size{KMGTP}: +5G
> Current type is 8300 (Linux filesystem)
> Hex code or GUID (L to show codes, Enter = 8300):
>
> NOTE: fdisk shouldn't really be used for partitioning GPT drives.
> NOTE: Leaving the last sector blank will automatically cause the remainder of space on the drive to be allocated.

Parted

Whilst gdisk and fdisk are designed for use with specific partition formatting (GPT or MBR), the parted command can be used to manage partitions on both. Parted is also a mini-CLI tool.

> Parted is a more versatile tool, but is also much ore complex to use. Also may not be installed by default on some systems. I would personally advise sticking with fdisk and gdisk.

Enter the parted CLI:

```
parted /dev/devicename
```

We then have the following options:

`print`	print current partition configuration
`mkpart`	create new partition

==========

Filesystems

What is a filesystem?

a filesystem is a method and data structure that the operating system uses to manage files on a storage device or partition. It provides a way to organize, store, retrieve, and manage data.

Filesystem types in Linux

EXT (Extended Filesystem)

The current newest EXT version is EXT4. EXT standards (2-4) provide backward compatibility. This is to ensure that newer versions of the file system can work seamlessly with older versions. This compatibility provides several important benefits:

1. Data Migration and Upgrade
2. Mixed Environment Compatibility
3. Data Integrity and Recovery

XFS (Extense Filesystem)

Able to track a much higher number of small files. XFS is better for systems handling large files or volumes of data - XFS is able to perform better than EXT in this scenario.

BTRFS (B-tree File System)

A modern file system developed by Oracle Corporation for Linux. It is designed to address various shortcomings of traditional file systems like ext4 and to offer advanced features, scalability, and improved performance.

> BTRFS whilst being much more advanced in it's capabilities than the alternatives, also has it's own shortcomings. For example, additional ability means additional complexity, BTRFS is also known to be temperamental.

BTRFS offers features such as:
cross server partitions

SWAP

SWAP is technically a partition type, but it isn't focused on traditional data storage. Rather SWAP is allocated space on a storage device used by a Linux System that allows for the storage device to

be used as an alternative to RAM. See this page for more info.

Configuring filesystems

Once a disk has been partitioned, we can then look to create a filesystem on that disk. You can also create a filesystem on an unpartitioned disk, this will cause the entire disk to be formatted into the specified filesystem type.

Check existing filesystem types:

```
df -T
```

or

```
lsblk -f
```

Create a filesystem on a device/partition

There are lots of different filesystem types available in Linux. To simplify the process for configuring filesystems, there are symlinks to binaries added in most systems that can be used to specify filesystem types:

```
root@test:~# ls -l /usr/sbin/mk*
lrwxrwxrwx 1 root root      8 Mar 23  2022 /usr/sbin/mkdosfs -> mkfs.fat
-rwxr-xr-x 1 root root 133752 Jun  1  2022 /usr/sbin/mke2fs
-rwxr-xr-x 1 root root  14720 Apr  9 15:32 /usr/sbin/mkfs
-rwxr-xr-x 1 root root  22912 Apr  9 15:32 /usr/sbin/mkfs.bfs
-rwxr-xr-x 1 root root 482560 Feb 24  2022 /usr/sbin/mkfs.btrfs
-rwxr-xr-x 1 root root  35144 Apr  9 15:32 /usr/sbin/mkfs.cramfs
lrwxrwxrwx 1 root root      6 Jun  1  2022 /usr/sbin/mkfs.ext2 -> mke2fs
lrwxrwxrwx 1 root root      6 Jun  1  2022 /usr/sbin/mkfs.ext3 -> mke2fs
lrwxrwxrwx 1 root root      6 Jun  1  2022 /usr/sbin/mkfs.ext4 -> mke2fs
-rwxr-xr-x 1 root root  52048 Mar 23  2022 /usr/sbin/mkfs.fat
-rwxr-xr-x 1 root root  43408 Apr  9 15:32 /usr/sbin/mkfs.minix
lrwxrwxrwx 1 root root      8 Mar 23  2022 /usr/sbin/mkfs.msdos -> mkfs.fat
lrwxrwxrwx 1 root root      6 Nov  1  2022 /usr/sbin/mkfs.ntfs -> mkntfs
lrwxrwxrwx 1 root root      8 Mar 23  2022 /usr/sbin/mkfs.vfat -> mkfs.fat
-rwxr-xr-x 1 root root 391952 Feb  9  2022 /usr/sbin/mkfs.xfs
-rwxr-xr-x 1 root root  22704 Jan 10 13:54 /usr/sbin/mkhomedir_helper
-rwxr-xr-x 1 root root  12453 Jun 14  2023 /usr/sbin/mkinitramfs
```

```
-rwxr-xr-x 1 root root  14648 Jun  1  2022 /usr/sbin/mklost+found
-rwxr-xr-x 1 root root  72072 Nov  1  2022 /usr/sbin/mkntfs
-rwxr-xr-x 1 root root  47496 Apr  9 15:32 /usr/sbin/mkswap
```

You can then create a filesystem using the format shown above.

> Example;
> I have just added an additional disk to my server (/dev/sdb). I've then split that disk into 2 separate partitions (/dev/sdb1 & /dev/sdb2). I'd like to format /dev/sdb2 as an EXT4 filesystem:
>
> `mkfs.ext4 /dev/sdb2`

Alternatively, you can use the full command syntax to specify a desired filesystem type:

```
mkfs -t type /dev/devicename
```

Mounting filesystems & fstab

Syntax to mount a filesystem (not persistent):

```
mount /dev/devicename /mountpoint
```

> Using the above command will force the system to try and identify the filesystem type of the filesystem you're mounting. The preferred and safer method is to specify the type manually, as shown below:
>
> `mount -t ext4 /dev/devicename /mountpoint`

Syntax to mount a filesystem (persistent) -fstab

To persistently mount a filesystem, we need to add an entry to the `/etc/fstab` file instructing the system to do this.

```
vim /etc/fstab
```

Once editing, you can then add a line, following the format below:

/dev/devicename /mountpoint fstype defaults

>

> Example;
>
> I've added a new additional disk to my machine, I've partitioned the disk into 2 (/dev/sdb1 & /dev/sdb2) and I've created filesystems on both (EXT4 on /dev/sdb1, XFS on /dev/sdb2). The below lines could be added to /etc/fstab to persistently mount these filesystems:
>
> `/dev/sdb1 /mnt/mount1 ext4 defaults 0 0`
> `/dev/sdb2 /mnt/mount2 xfs defaults 0 0`
>
> Once the entries have been added, the `mount -a` command can be used to mount all entries in /etc/fstab.

You'll notice on the above fstab entries, there is 'defaults'. This specifies that the system default options should be used, however, there are lots of options available for us to set, some common examples I've listed below:

ro	read only
rw	read & write
async	write operations may be cached and written to disk at a later time. This can improve performance but may risk data integrity in case of unexpected shutdowns.
sync	writes have to be completed before the next write starts. Ensures better data integrity at the cost of performance.

You'll also notice the 2 numbers after defaults:

First number (Dump Field) (potential value 0 or 1)	determines whether disks will be backed up when the dump command is run. The dump command is deprecated so this value can safely be set to 0.
Second number (Pass Field) (potential value 0 or 1)	This field determines the order in which `fsck` checks filesystems during the boot process. • A value of `0` means that the filesystem should not be checked by `fsck` during system boot. • A value of `1` or higher indicates the order in which `fsck` should check the filesystem.

Revision #33
Created 2024-06-08 22:28:07 UTC by Daniel
Updated 2024-06-09 15:45:43 UTC by Daniel