Learning Objectives - Storage Concepts

Learners will be able to...

- Explain the difference between File, Block and Object storage
- Explain the difference between soft and hard links
- Elaborate on the different types of metadata stored for files
- Access the interface for Filesystem in Userspace (FUSE)

Introduction to Storage Concepts

In this assignment we will cover three different storage concepts - File Storage, Block Storage and Object Storage.

File storage is the way you access files on a personal computer, individual items are stored in directories or folders. You access your data via the hierarchical system.

Block Storage breaks data up into blocks, each block of data has a unique identifier and each block can be in a different location. Accessing Block storage is faster because you do not have to trace down a file tree. Block Storage is also file system independent and the data can be spread across devices running different file systems.

Object storage, also referred to as object based storage, is the newest technology of these three. With Object storage the application does not need to know where the data resides. Object data is retrieved through Application Programmer APIs (API) using the unique identifier associated with the object.

Object Storage is the cheapest of these three types of storage and most cost effective for high volumes of data. File storage works best for low volumes of data.

Block Storage

Block storage breaks data up in blocks and stores each block with a unique identifier.

The block device refers to the particular piece of hardware that is used to store data. Examples of block devices are: hard drives, solid state drives, flash drives, Storage Area Networks (SAN) and cloud storage.

The advantages of block storage is that it allows data to be spread over multiple environments, storage is not tied to file system compatibility. Block storage data retrieval is also faster than file storage data retrieval because there can be multiple paths to a block.

A disadvantage of block storage is that there is no metadata and therefore not as suitable for unstructured data. Search capabilities are limited with this storage format.

Underneath the file system you access via hierarchical addressing in the terminal window on the left is a Block storage system.

The `lsblk` command

The `lsblk` command can be used to find information about all available block devices.

```
lsblk
```

In the case of the virtual environment you are working in, you mostly see “loop” in this list. These are loopback devices, which are virtual devices and mean that all your data is stored on a different system. The **NVME** block devices you see indicate that the block devices are part of Amazon Web Services, the service that hosts Codio data.

The `blkid` command

The `blkid` can determine the type of content that a block device holds, and also the attributes from the content metadata.

Learn more about the `blkid` command:

```
man blkid
```

File Storage

File Storage, sometimes referred to as hierarchical storage, is when files are stored in folders or directories. Files are referenced by the directory/folder path you have to traverse to get to them. A **File Storage** system is the most common system for personal computers.

As we mentioned earlier in this course, on graphical systems such as the MacOS or Windows the term **folder** is used, on command line operating systems, such as Linux, the file containers are called **directories**.

We have already covered the Linux file system quite a bit in this course but as a reminder, try the `ls -l -a -p` command in the terminal. You will see a listing of the contents of the directory with extended information and including any hidden files.

```
ls -l -a -p
```

Soft and Hard links

A hard link associates a file name with a location on a disk.

Hard links can only be created for regular files, not for directories or special files. They cannot span multiple file systems.

Create a file with some content, when we use the `cat` command it will print the contents of that file to the console:

```
echo "Hello world" > myfile.txt
cat myfile.txt
```

Create a hard link to myfile.txt using the `ln` command

```
ln myfile.txt test/myfile.txt
cat test/myfile.txt
```

Modify the file using the path to the directory

```
echo "I added a line" >> test/myfile.txt
cat myfile.txt
cat test/myfile.txt
```

They are the same file, you can use either path for editing and you will be editing the same file.

The same is true for changing permissions on one of the file names, it will change them for both

```
chmod o+w test/myfile.txt
ls -l test/myfile.txt
ls -l myfile.txt
```

A soft link is a special file that points to an existing file

Soft links are also called symbolic links and they can span multiple file systems.

The `ln` command with the `-s` argument will create a symbolic link

```
ln -s myfile.txt myfilesym.txt
ls -l
```

You can see that the symbolic link looks different but if you try out a command using the link, it works the same

```
echo "I added another line" >> myfilesym.txt
cat myfilesym.txt
```

The soft links and hard links seemingly work the same. The one drawback with soft links is that if the original file is deleted the soft link is broken and you are left with a dangling link.

Create a “dangling” link

```
rm myfile.txt
cat myfilesym.txt
```

File metadata

On Linux the file metadata contains the following information:
- File type (data file, directory or symbolic link)
- Name
- Timestamps (creation date/time, last access date/time and modification date/time)
- Location in the file system (file tree)
- Size in bytes
- Physical location (block storage addresses)
- Ownership
- Access permissions

The `stat` command

We'll create a new file called `test.txt` and run `stat` on it. We'll also run it on a command we use all the time, `ls`.

```
touch test.txt
stat test.txt
stat /bin/ls
```

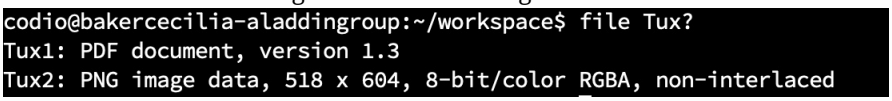
You'll notice something in the output called **inode**. The **inode** number refers to the data structure that stores all the file information other than the name. The number is assigned when a file is created.

The `file` command

The `file` command tells you more about the file type.

```
file Tux?
```

You should see something similar to the image below.



challenge

Try these variations

Guess what the file type will be before you run the command
* A text file
* Something in the /bin directory
* A folder

Object Storage

Object Storage is the newest of the three storage methods we are exploring in this assignment, the technology was introduced in the late 1990s.

In the Object Storage model, each object includes the data, metadata and a unique identifier. This method is particularly well suited for storing large amounts of unstructured data that is not being continuously modified (static data) such as videos, photos and songs.

Object Storage allows for a variable amount of metadata and as such allows for custom application or user specific information. Object data is retrieved through Application Programmer APIs (API) using the unique identifier associated with the object. The application does not need to know where the data is actually stored and the operating system does not manage the data.

Typically Object-based storage uses TCP/IP as its transport, and devices communicate using HTTP and REST APIs.

FUSE

What is FUSE

FUSE (Filesystem in Userspace) allows users to create their own file systems without editing Linux kernel code. These file systems may be mounted by users who do not have root privileges. You can use FUSE to extend the features of the Linux file system without the need to learn Linux file system internals or kernel module programming.

The **libfuse** library offers two Application Programmer Interfaces (APIs), the high level API allows you to access files using file names and paths and the low level API uses inodes.

The libraries and utilities you need to develop your own file system are available here.

More information about FUSE.