- 3. 2. Files and Directories -

- INDEX -

- 3. 2. 1. Directory Structure -

The Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Linux distributions.

In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.

Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.

The most important Linux directories are the following:

/ – The Root Directory

Everything on your Linux system is located under the / directory, known as the root directory.

You can think of the / directory as being similar to the C:\ directory on Windows – but this isn't strictly true, as Linux doesn't have drive letters.

While another partition would be located at D:\ on Windows, this other partition would appear in another folder under / on Linux.

/bin – Essential User Binaries

The /bin directory contains the essential user binaries (programs) that must be present when the system is mounted in single-user mode.

Applications such as Firefox are stored in /usr/bin, while important system programs and utilities such as the bash shell are located in /bin.

The /usr directory may be stored on another partition – placing these files in the /bin directory ensures the system will have these important utilities even if no other file systems are mounted.

The /sbin directory is similar – it contains essential system administration binaries.

/boot – Static Boot Files

The /boot directory contains the files needed to boot the system – for example, the GRUB boot loader's files and your Linux kernels are stored here.

The boot loader's configuration files aren't located here, though – they're in /etc with the other configuration files.

/dev – Device Files

Linux exposes devices as files, and the /dev directory contains a number of special files that represent devices.

These are not actual files as we know them, but they appear as files – for example, /dev/sda represents the first SATA drive in the system.

If you wanted to partition it, you could start a partition editor (like GParted) and tell it to edit /dev/sda.

/etc – Configuration Files

The /etc directory contains configuration files, which can generally be edited by hand in a text editor.

Note that the /etc/ directory contains system-wide configuration files, but user-specific configuration files are located in each user's home directory.

/home – Home Folders

The /home directory contains a home folder for each user.

For example, if your user name is **bob**, you have a home folder located at **/home/bob** .

This home folder contains the user's data files and user-specific configuration files.

Each user only has write access to their own home folder and must obtain elevated permissions (become the root user) to modify other files on the system.

/lib – Essential Shared Libraries

The /lib directory contains libraries needed by the essential binaries in the /bin and /sbin folder.

Libraries needed by the binaries in the /usr/bin folder are located in /usr/lib.

/lost+found – Recovered Files

Each Linux file system has a lost+found directory.

If the file system crashes, a file system check will be performed at next boot.

Any corrupted files found will be placed in the lost+found directory, so you can attempt to recover as much data as possible.

/media – Removable Media

The /media directory contains subdirectories where removable media devices inserted into the computer are mounted.

For example, when you insert a CD into your Linux system, a directory will automatically be created inside the /media directory.

You can access the contents of the CD inside this directory.

/mnt – Temporary Mount Points

Historically speaking, the /mnt directory is where system administrators mounted temporary file systems while using them.

For example, if you're mounting a Windows partition to perform some file recovery operations, you might mount it at /mnt/windows.

However, you can mount other file systems anywhere on the system.

/opt – Optional Packages

The /opt directory contains subdirectories for optional software packages.

It's commonly used by proprietary software that doesn't obey the standard file system hierarchy – for example, a proprietary program might dump its files in /opt/application when you install it.

/proc – Kernel & Process Files

The /proc directory is similar to the /dev directory because it doesn't contain standard files.

It contains special files that represent system and process information.

/root – Root Home Directory

The /root directory is the home directory of the root user.

Instead of being located at **/home/root** , it's located at **/root** .

This is distinct from /, which is the system root directory.

/run – Application State Files

The /run directory is fairly new, and gives applications a standard place to store transient files they require like sockets and process IDs.

These files can't be stored in /tmp because files in /tmp may be deleted.

/sbin – System Administration Binaries

The /sbin directory is similar to the /bin directory.

It contains essential binaries that are generally intended to be run by the root user for system administration.

/srv – Service Data

The /srv directory contains "data for services provided by the system."

If you were using the Apache HTTP server to serve a website, you'd likely store your website's files in a directory inside the /srv directory.

/tmp – Temporary Files

Applications store temporary files in the /tmp directory.

These files are generally deleted whenever your system is restarted and may be deleted at any time by utilities such as tmpwatch.

/usr – User Binaries & Read-Only Data

The /usr directory contains applications and files used by users, as opposed to applications and files used by the system.

For example, non-essential applications are located inside the /usr/bin directory instead of the /bin directory and non-essential system administration binaries are located in the /usr/sbin directory instead of the /sbin directory.

Libraries for each are located inside the /usr/lib directory.

/var – Variable Data Files

The /var directory is the writable counterpart to the /usr directory, which must be read-only in normal operation.

Log files and everything else that would normally be written to /usr during normal operation are written to the /var directory.

For example, you'll find log files in /var/log.

- 3. 2. 2. Absolute and Relative Paths -

An absolute path starts from the directory root (/) and goes up to the actual object (file or directory).

It contains the names of all directories that come in the middle of the directory root and the actual object.

In this, name of the parent directory is written in the left.

To write the absolute path of this directory, we have to start writing the path from the directory root.

The directory root is written as / (forward slash).

An absolute path does not depends on the current directory.

For example, if we want to change to the directory named "test" using an absolute path we should write the following command:

cd /home/alumno/test

A relative path starts from the current directory and goes up to the actual object.

A relative path depends on the current directory.

When we change the directory, the relative path also changes.

Just like the absolute path, the name of the parent directory is written in the left side.

Unlike the absolute path, all slashes in the relative path represent the directory separator.

For example, if we are located in the **"home"** directory and we want to change to the directory named **"test"** using a relative path we should write the following command:

cd alumno/test

- 3. 2. 3. Standard Input, Output, Error, Pipelining and Redirections -

Input / Output of shell commands

Many of the basic Linux shell commands work in a similar way.

Every command that you start from the shell gets three *channels* assigned:

- Standard Input - STDIN (channel 0):
  Where your command draws the input from. If you don't specify anything special this will be your keyboard input.
- Standard Out - STDOUT (channel 1):
  Where your command's output is sent to. If you don't specify anything special the output is displayed in your shell (screen).
- Standard Error - STDERR (channel 2):
  If anything wrong happens the command will send error message here. By default the output is also displayed in your shell (screen).

The most basic command that just passes everything through from STDIN to STDOUT is the 'cat' command.

Just open a shell and type 'cat' and press Enter.

Nothing seems to happen.

But actually 'cat' is waiting for input.

Type something like “hello world”.

Every time you press ‘Enter’ after a line ‘cat’ will output your input.

So you will get an echo of everything you type.

To let ‘cat’ know that you are done with the input send it an ‘end-of-file’ (EOF) signal by pressing Ctrl-D on an empty line.

Pipelining

A more interesting application of the STDIN/STDOUT is to chain commands together.

The output of the first command becomes the input of the second command usind the "pipe" symbol: **|**

For example:

ls | more

is going to show you the results of **"ls"** but page by page because of pipelining **"ls"** with **"more"**.

Redirections

If you want to redirect the Standard Input, use this symbol: **<**

For example:

sort < file_list.txt

In the example above, we used the **sort** command to process the contents of **file_list.txt** and the results are output on the screen (Standard Output).

If you want to redirect the Standard Output, use this symbol: **>**

For example:

ls > file1.txt

is going to copy the results of **"ls"** inside a file named **"file1.txt"**.

In this case, if the file exists, the content is erased and susbtituted by the redirection.

If the file exists and you want to append the redirection to the current information, you should do:

ls >> file1.txt

If you want to redirect the Standard Error, use this symbol: **2>**

For example:

sort < /etc/passwd > file1.txt 2> error1.txt

is going to sort the **/etc/passwd** file, place the results in a file called **"file1.txt"**, and trap any errors in a file called **"error1.txt"**.

- 3. 2. 4. Commands: Files and Directories -

Goes up one level of the directory tree and changes into the parent directory

cd ..

Goes to the $HOME directory

cd

Changes to the /dir1 directory

cd /dir1

Lists all files in a long listing (detailed) format

ls -al

Displays the present working directory

pwd

Displays the current user

whoami

Cleans the screen

clear

Repeats the text or the content of the variable

echo var

"sudo" originally stood for "superuser do" as the older versions of sudo were designed to run commands only as the superuser. However, the later versions added support for running commands not only as the superuser but also as other (restricted) users, and thus it is also commonly expanded as "substitute user do". Although the latter case reflects its current functionality more accurately, sudo is still often called "superuser do" since it is so often used for administrative tasks.

sudo

su, which stands for substitute user is used by a computer user to execute commands with the privileges of another user account. When executed it invokes a shell without changing the current working directory or the user environment.

su

Creates a directory

mkdir dir2

Removes a directory

rmdir dir3

Removes a file

rm file1

Removes the directory and its contents recursively

rm -r dir3

Forces removal of file without prompting for confirmation

rm -f file2

Forcefully removes directory recursively

rm -rf dir4

Copies file1 to file2

cp file1 file2

Copies source_directory recursively to destination

cp -r source_directory destination

Renames or moves file1 to file2. If file2 is an existing directory, move file1 into directory file2

mv file1 file2

Creates an empty file or update the access and modification times of file

touch file

Views the contents of file

cat file

Displays the contents of file page by page

more file

Displays the first 10 lines of file

head file

Displays the last 10 lines of file

tail file

Edits the contents of file from the Terminal

nano file

Edits the contents of file from the Terminal

vi file

Edits the contents of file from the GUI

gedit file

Searches for pattern in file

grep pattern file

Searches recursively for pattern in directory

grep -r pattern directory

Finds files in /home/john that start with "prefix"

find /home/john -name 'prefix*'

Finds files larger than 100MB in /home

find /home -size +100M

Sorts alphabetically

sort

Counts newlines, words, and bytes for file

wc file

Shows help of a particular command

man command

Displays the tree directory

sudo apt install tree

tree

- Exercises -

You are going to create a new Google Document inside the "3. Linux" folder of your Google Drive, named:

"3. 2. Files and Directories - Apellidos, Nombre"

being "Apellidos, Nombre" your Last Name and Name.

Share this Google Document with the teacher (jorge@iesdoctorbalmis.com) with "Edit" permissions.

Inside this Google Document you are going to copy and answer all the "Exercises" of this sub-unit.

1. Create the directory **/dam** using relative paths.

2. Create the directory **/dam/test** using absolute paths.

3. Create the directory **/dam/test/balmis** using relative paths.

4. Delete the directory **/dam/test/balmis** using relative paths.

5. Create the file **/dam/test/doc1.txt** .

6. Edit the content of the file **/dam/test/doc1.txt** and add 3 rows inside: "row1","row2" and "row3".

7. Copy the file **/dam/test/doc1.txt** as **/dam/test/doc2.txt** .

8. Move the file **/dam/test/doc2.txt** to the **/dam** directory.

9. View the contents of the file **/dam/doc2.txt** .

10. Search for the the text "row2" inside the file **/dam/doc2.txt** .

11. Search for the the text "row4" inside the file **/dam/doc2.txt** .

12. Edit the content of the file **/dam/doc2.txt** and add 4 rows inside: "row7", "row6","row5" and "row4".

13. Sort alphabetically the content of the file **/dam/doc2.txt** .

14. Copy the file **/dam/doc2.txt** as **/dam/doc3.txt** .

15. Find the files in **/dam** that start with **"doc"** .

16. Find the files in **/dam** that have the **".txt"** extension.

17. Show the contents of the first 3 rows of the the file **/dam/doc3.txt** .

18. Show the contents of the last 2 rows of the the file **/dam/doc3.txt** .

19. Display the present working directory.

20. Show help information for the **"grep"** command.

21. Copy the file **/dam/test/doc1.txt** to your home folder (normally **/home/alumno**).

22. Copy the file **/dam/doc2.txt** to your home folder (normally **/home/alumno**).

23. Search for the the text "row2" inside the file **/home/alumno/doc1.txt** and save the result in a new file named **/home/alumno/doc4.txt** .

24. Take the content of the file **/home/alumno/doc2.txt** , sort the content, and save the result in a new file named **/home/alumno/doc5.txt** .

25. Take the contents of the last 3 rows of the the file **/home/alumno/doc2.txt** , sort them, and show them on the screen.