



ALX LESSON 0x00. Shell, basics

Shell - Bash



TABLE OF CONTENTS

01

Overview
topics

02

Learning
Objectives

03

Quiz
questions

04

hands on lab
practice



01

OVERVIEW topics

We Done Them!

*Green will be explained

*purple done

Ubuntu Shell Commands

Topics

What does RTFM mean?

What is a Shebang

What is the Shell

What is the difference between a terminal and a shell

What is the shell prompt

How to use the history (the basics)

What do the commands or built-ins cd, pwd, ls do

How to navigate the filesystem

What are the . and .. directories

What is the working directory, how to print it and how to change it

What is the root directory

What is the home directory, and how to go there

What is the difference between them

What are the characteristics of hidden files and how to list them

What does the command cd - do

What do the commands ls, less, file do

We Done Them!

*Green will be explained

*purple done

Ubuntu Shell Commands

Topics

Understand the ls long format and how to display it

What does the ln command do

What do you find in the most common/important directories

What is a symbolic link

What is a hard link

What is the difference between a hard link and a symbolic link

Manipulating Files

What do the commands cp, mv, rm, mkdir do

What are wildcards and how do they work

How to use wildcards

We Done Them!

*Green will be explained

*purple done

Ubuntu Shell Commands

Topics

What do type, which, help, man commands do

What are the different kinds of commands

What is an alias

When do you use the command help instead of man

Common shortcuts for Bash

What does LTS mean?

How to read a man page

What are man page sections

What are the section numbers for User commands,

System calls and Library functions

We Done Them!

*Green will be explained

*purple done

Ubuntu Shell Commands

Topics

cd

ls

pwd

less

file

ln

cp

mv

rm

mkdir

type

which

help

man

chown

chmod

Watch First!



ALX LESSON
Ubuntu Shell
Commands

Shell - Bash

Watch Second!



ALX LESSON
Ubuntu Shell
Commands P2

Shell - Bash

OVERVIEW Topics

Ubuntu Shell Commands

Topics

- file
- ln
- type
- which
- help
- Man

Slides On Telegram

https://t.me/alx_2023

Ubuntu Shell Commands

Topics



@ALX_2023



02

Learning Objectives

What does RTFM mean?

RTFM is an acronym that stands for "Read The F***ing Manual."

It is often used as a derogatory and sarcastic remark to someone who asks a question that could have easily been answered by reading the documentation or manual that came with a product or software.

It can also be used as a humorous way to remind someone to refer to the instructions before asking for help. However, it is generally considered impolite and rude to use this acronym in a professional or polite context.

What is a Shebang

a shebang (also known as a hashbang or a pound-bang) is the first line in a script file that starts with the characters "#!" (hash followed by exclamation mark). It is used to specify the interpreter that should be used to execute the script.

For example, a shebang line for a Bash script might look like this:

```
#!/bin/bash
```

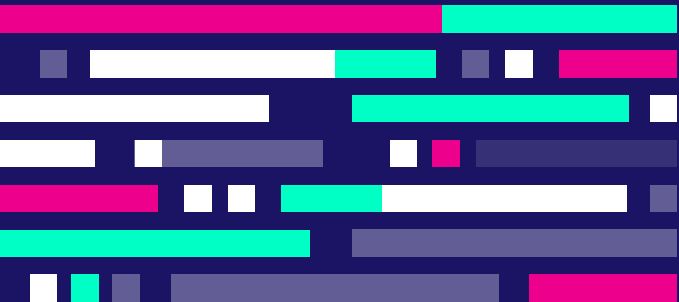
This shebang line tells the operating system to use the Bash interpreter to execute the script.



What is shell

In computing, a shell is a program that provides an interface for users to interact with the operating system's services and functions. The shell acts as a command-line interpreter that allows users to execute commands and run scripts.

In Unix-like operating systems, such as Linux, there are several different shell programs available, including **Bash**, **Zsh**, and **Fish**. Each shell has its own syntax and set of features, but they all provide a similar interface for interacting with the operating system.



What is the difference between a terminal and a shell

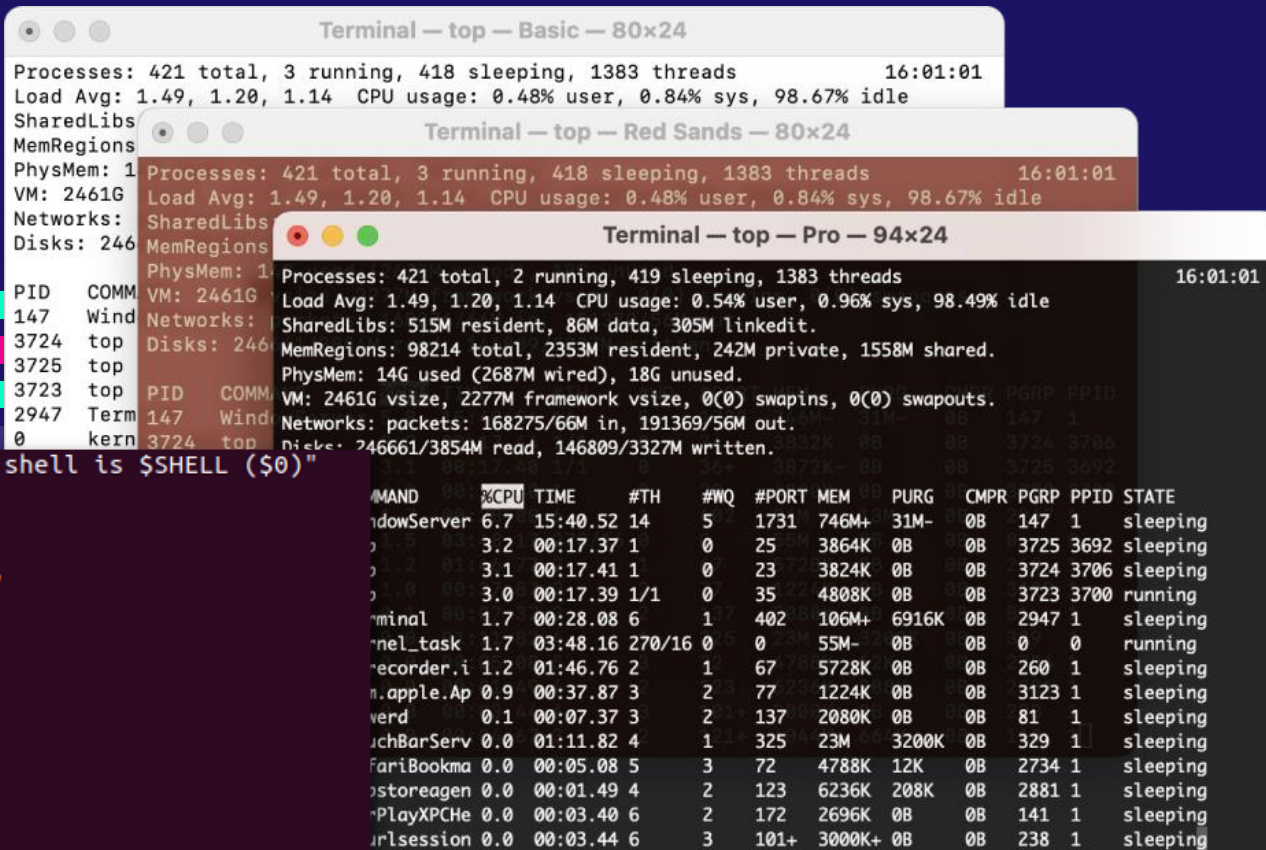
In computing, a terminal and a shell are two different concepts that are often used interchangeably, but they refer to different things.

A terminal is a program that provides a graphical or text-based interface for users to interact with the operating system's command-line interface (CLI). The terminal provides a window where users can enter commands, view output, and interact with the operating system using a keyboard and/or mouse. Examples of terminal programs include [GNOME Terminal](#), [Konsole](#), and [iTerm2](#).

A shell, on the other hand, is a program that interprets and executes commands entered by the user in the terminal. The shell provides a command-line interface for users to interact with the operating system and its services. The shell reads commands entered by the user and executes them, providing feedback and output to the user. Examples of shell programs include [Bash](#), [Zsh](#), and [Fish](#).

What is the difference between a terminal and a shell

```
vivek@nixcraft-asus:~$ echo "My current shell is $SHELL ($0)"
My current shell is /bin/bash (bash)
vivek@nixcraft-asus:~$
vivek@nixcraft-asus:~$ ksh
$ echo "My current shell is $SHELL ($0)"
My current shell is /bin/bash (ksh)
$ echo $SHELL
/bin/bash
$ tcsh
nixcraft-asus:~>
nixcraft-asus:~> echo $SHELL
/bin/bash
nixcraft-asus:~> echo $0
tcsh
nixcraft-asus:~> exit
exit
```




The image shows three overlapping terminal windows. The top window, titled 'Terminal — top — Basic — 80x24', displays system statistics: 421 total processes (3 running, 418 sleeping, 1383 threads), load averages of 1.49, 1.20, and 1.14, and CPU usage of 0.48% user, 0.84% sys, and 98.67% idle. The middle window, titled 'Terminal — top — Red Sands — 80x24', shows identical system statistics. The bottom window, titled 'Terminal — top — Pro — 94x24', displays the same system statistics followed by a detailed process list table.

COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPR	PGRP	PPID	STATE
WindowServer	6.7	15:40.52	14	5	1731	746M+	31M-	0B	147	1	sleeping
...
...	3.2	00:17.37	1	0	25	3864K	0B	0B	3725	3692	sleeping
...	3.1	00:17.41	1	0	23	3824K	0B	0B	3724	3706	sleeping
...	3.0	00:17.39	1/1	0	35	4808K	0B	0B	3723	3700	running
terminal	1.7	00:28.08	6	1	402	106M+	6916K	0B	2947	1	sleeping
gnome_task	1.7	03:48.16	270/16	0	0	55M-	0B	0B	0	0	running
recorder.i	1.2	01:46.76	2	1	67	5728K	0B	0B	260	1	sleeping
gnapple.Ap	0.9	00:37.87	3	2	77	1224K	0B	0B	3123	1	sleeping
verd	0.1	00:07.37	3	2	137	2080K	0B	0B	81	1	sleeping
uchBarServ	0.0	01:11.82	4	1	325	23M	3200K	0B	329	1	sleeping
FariBookma	0.0	00:05.08	5	3	72	4788K	12K	0B	2734	1	sleeping
ostoreagen	0.0	00:01.49	4	2	123	6236K	208K	0B	2881	1	sleeping
PlayXPCh	0.0	00:03.40	6	2	172	2696K	0B	0B	141	1	sleeping
urlsession	0.0	00:03.44	6	3	101+	3000K+	0B	0B	238	1	sleeping

What is the shell prompt

the shell prompt is the text that appears in the terminal or console to indicate that the shell is ready to accept commands from the user. The prompt typically includes the current working directory, the username, and the name of the shell being used.



`username@hostname:~/Documents$` - This prompt includes the username, hostname, current working directory, and the `$` symbol to indicate that the shell is ready to accept commands.

`bash-5.1$` - This prompt includes the name of the shell being used and the `$` symbol to indicate that the shell is ready to accept commands.

`>>` - This prompt is a minimalistic prompt that includes only the `>>` symbol to indicate that the shell is ready to accept commands.

How to use the history (the basics)

the history command is used to display a list of commands that have been executed in the current shell session.

`history`

`!3`

This will execute the third command in the history.

`history | grep command`

This will display a list of all commands that contain the specified text.

`!!`

This will execute the last command in the history.

`history -c`

This will clear the entire history for the current shell session.

What are the . and .. directories

The . directory refers to the current directory, which is the directory that you are currently in.

For example, if you are in the `/home/user/Documents` directory, the . directory refers to the `/home/user/Documents` directory itself.

The .. directory refers to the parent directory, which is the directory that contains the current directory.

For example, if you are in the `/home/user/Documents` directory, the .. directory refers to the `/home/user` directory.

How do you use options and arguments with commands

Identify the command you want to use, along with any options or arguments that you want to pass to the command. Options are usually specified with a single dash (-) or double dash (--), followed by one or more letters or words. Arguments are additional information that the command needs to perform its task.



```
ls -l /home/user/Documents
```

If the command has multiple arguments, you can separate them with spaces. For example:

```
cp file1.txt file2.txt /home/user/Documents
```

If an argument or option contains spaces or special characters, you can enclose it in quotes to prevent the shell from interpreting it as separate arguments or options. For example:

```
grep "search term" file.txt
```

What does the ln command do

The ln command in Linux is used to create links between files or directories. There are two types of links that can be created:

Hard links: A hard link is a file that points to the same **inode** as another file. Hard links are useful when you want to have multiple names for the same file, but without creating a copy of the file itself. Hard links can be used to save disk space, since only one copy of the file is actually stored on disk, but it can be accessed using different names. Hard links can also be used to create a backup copy of a file, since changes made to the original file will be reflected in all of its hard links.

ln source_file hard_link

This command creates a hard link named **hard_link** that points to the **source_file**. If you make changes to the **source_file**, the hard link will reflect those changes.

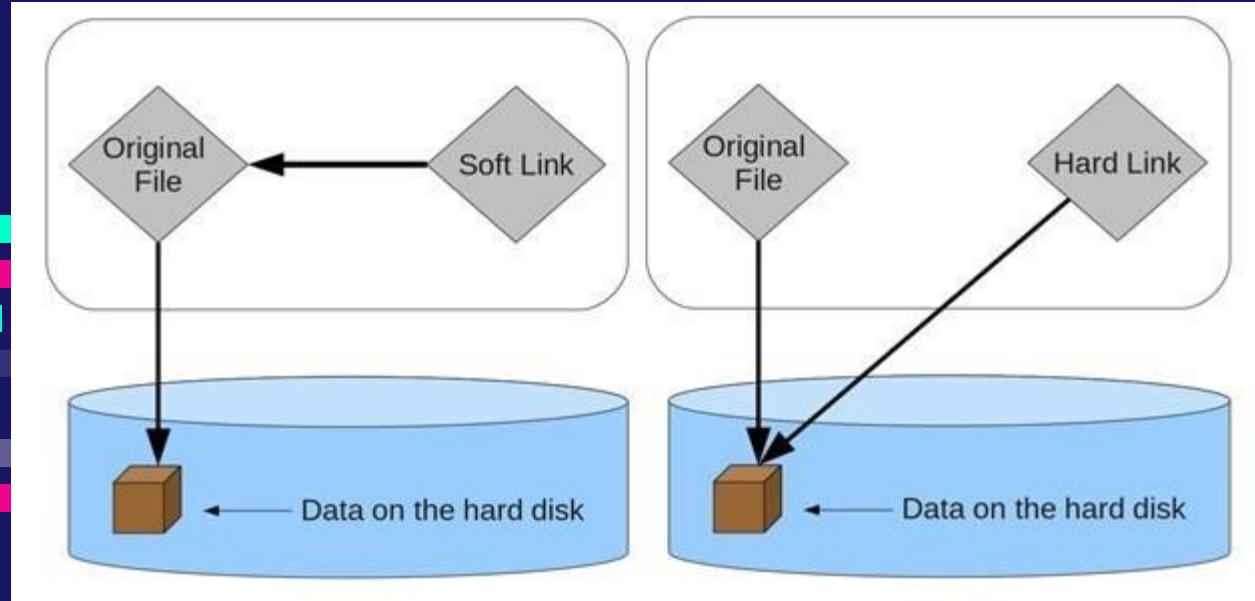
What does the ln command do

Symbolic links: A symbolic link is a special type of file that points to another file or directory. Symbolic links are useful when you want to create a shortcut to a file or directory, or when you want to reference a file or directory that has a long or complicated path. Symbolic links can be used across different file systems or partitions, since they point to a path rather than an inode. Symbolic links can also be used to create relative links, which can be useful when you want to reference files or directories relative to the location of the symbolic link.

ln -s source_file symbolic_link

This command creates a symbolic link named **symbolic_link** that points to the **source_file**. If you make changes to the **source_file**, the symbolic link will reflect those changes.

What does the ln command do



What does the ln command do

Hard links cannot be created for directories or across different file systems.

Symbolic links can be created for directories, and can span across different file systems.

When you delete a file or directory that has **hard links**, the file or directory is not actually deleted until all hard links to it are removed.

When you delete a file or directory that has **symbolic links**, the links themselves become invalid but the original file or directory is not deleted.

What does the ln command do

```
ls -l filename
```

This will display the file or directory's detailed information, including the number of hard links in the second column

To display the inode number and the hard links of a file or directory, you can use the following command:

```
ls -li filename
```

```
find / -inum 12345
```

This command will search the entire file system (/) for all files with inode number 12345

Use to unlink any of both links

```
unlink <name of link>
```

What do you find in the most common/important directories

`/` (root directory): This is the top-level directory in the file system hierarchy. It contains all other directories and files in the system. Only the root user has write access to this directory.

`/bin`: This directory contains essential binary files that are necessary for the system to function, such as the `ls` and `cp` commands.

`/boot`: This directory contains the boot loader files, kernel files, and initial ramdisk files used during the boot process.

`/dev`: This directory contains device files that represent physical and logical devices on the system, such as hard drives, USB drives, and serial ports.

What do you find in the most common/important directories

`/etc`: This directory contains system configuration files, such as user accounts, network settings, and system-wide application settings.

`/home`: This directory contains user home directories, which store user-specific configuration files, documents, and other data.

`/lib`: This directory contains library files that are required by programs in `/bin` and `/sbin`.

`/mnt`: This directory is used as a mount point for temporary file systems, such as USB drives or network file systems.

`/opt`: This directory contains optional software packages that are installed separately from the system packages.

What is a symbolic link

What is a hard link

What is the difference between a hard link and a symbolic link



What are wildcards and how do they work

Asterisk (*): Matches any number of characters, including zero. For example, ".txt" matches any file with a .txt extension, while "file*" matches any file that begins with the word "file".

Question mark (?): Matches exactly one character. For example, "f?le" matches "file" and "fyle", but not "files".

Bracket expressions ([]): Matches any one character within the brackets. For example, "[abc]" matches "a", "b", or "c".

How to use wildcards

ls: List files in a directory that match a pattern. For example, "`ls *.txt`" lists all files with a .txt extension in the current directory.

cp: Copy files that match a pattern. For example, "`cp *.txt /path/to/destination`" copies all files with a .txt extension in the current directory to the specified destination.

rm: Remove files that match a pattern. For example, "`rm file*`" removes all files that begin with the word "file".

mv: Move files that match a pattern. For example, "`mv *.txt /path/to/destination`" moves all files with a .txt extension in the current directory to the specified destination.

What do type, which, help, man commands do

type: Shows the type of a command or program, such as whether it is a shell built-in or an executable file. For example, "type ls" shows that "ls" is an executable file.

which: Shows the location of a command or program. For example, "which ls" shows the location of the "ls" command.

help: Provides built-in help for a command. For example, "help cd" provides information on how to use the "cd" command within the shell.

man: Displays the manual page for a command or program. For example, "man ls" displays the manual page for the "ls" command, which provides detailed information on how to use the command and what options it supports.

What are the different kinds of commands

Built-in commands: These are commands that are part of the shell itself and are executed by the shell without creating a new process. Examples include `cd`, `echo`, and `pwd`.

Shell scripts: These are scripts written in a scripting language such as Bash, and can be executed as commands in the terminal. Shell scripts are a series of commands that are executed sequentially.

Executable files: These are standalone executable files that can be run as commands in the terminal. They can be compiled binaries or interpreted scripts.

What are the different kinds of commands

Alias commands: These are user-defined commands that are created by assigning a command string to a short name. Aliases can be created using the alias command.

External commands: These are commands that are not built into the shell and are stored in separate executable files. Examples include ls, cat, and grep.

System commands: These are commands that are related to the operating system and are used for system management tasks. Examples include systemctl, useradd, and mount.

What are the different kinds of commands

Built-in commands:

cd: changes the current working directory

echo: prints a message to the terminal

pwd: displays the current working directory

Shell scripts:

A script that creates a backup of a directory and its contents

A script that automates a series of commands to set up a development environment

Executable files:

/usr/bin/firefox: launches the Firefox web browser

/usr/bin/python3: launches the Python 3 interpreter

a file system onto a directory



What are the different kinds of commands

Alias commands:

alias ll='ls -la': creates an alias 'll' that runs 'ls -la'

External commands:

ls: lists the contents of a directory

cat: displays the contents of a file

grep: searches for a pattern in a file

System commands:

systemctl: controls the system's services and daemons

useradd: adds a new user to the system

mount: mounts a file system onto a directory



What is an alias

an alias is a way to create a short name or custom command that maps to a longer command or series of commands. Aliases are often used to make it easier to execute frequently used or complex commands.

```
alias newcommand='originalcommand'
```

For example, to create an alias for "ls -l --color=auto" as "ll", you would type:

```
alias ll='ls -l --color=auto'
```

When do you use the command help instead of man

When you type help followed by a command name, it displays a **brief** description of the command and its options. For example, if you type **help cd**, it will display information on how to use the cd command to change directories.

man ls

it will display the manual page for the **ls** command, which includes **detailed** information on how to use the command to list files and directories.

Common shortcuts for Bash

Ctrl + A: Move cursor to the beginning of the line.

Ctrl + E: Move cursor to the end of the line.

Ctrl + U: Cut (delete) the text from the cursor to the beginning of the line.

Ctrl + K: Cut (delete) the text from the cursor to the end of the line.

Ctrl + W: Cut (delete) the word before the cursor.

Ctrl + Y: Paste (yank) the previously cut text.

Ctrl + L: Clear the screen.

Ctrl + R: Search the command history.

Tab: Auto-complete a command or filename.

!!: Repeat the last command.

!\$: Reuse the last argument of the previous command.

!n: Repeat the nth command in the history.

Ctrl + C: Cancel the currently running command.

Ctrl + D: Exit the shell (logout if the shell is empty).

Ctrl + Z: Suspend the currently running command or process.

What does LTS mean?

LTS stands for "Long Term Support". In the context of software releases, an LTS version is a version that is supported for an extended period of time, typically several years. During this period, the software is maintained and receives regular updates and security patches, even after newer versions are released.

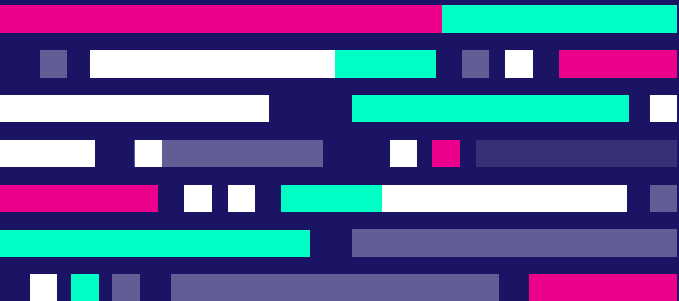
`lsb_release -a`: This command shows information about the current distribution, including the release number and whether it is an LTS release.

`ubuntu-support-status`: This command shows the status of support for packages installed on an Ubuntu system, including which packages are supported by the LTS release and which are not.

How to read a man page

Open the man page: To open a man page, type "man" followed by the name of the command or function you want to learn more about. For example, to open the man page for the "ls" command, you would type "man ls" in the terminal.

Navigate the man page: Once the man page is open, you can navigate through it using the arrow keys on your keyboard. Use the down arrow key to scroll down, and the up arrow key to scroll up.



What are man page sections

NAME: This section provides the name of the command or function and a brief description of what it does.

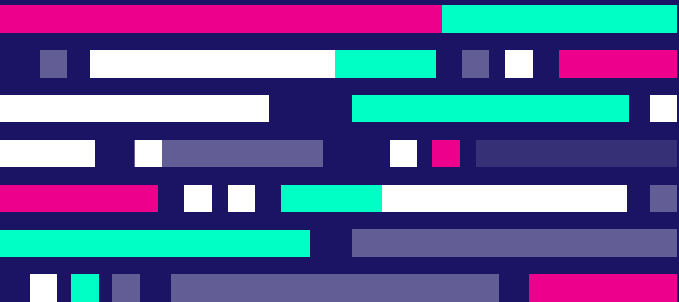
SYNOPSIS: This section provides a summary of the command syntax, including any options or arguments that can be used.

DESCRIPTION: This section provides a more detailed description of what the command or function does, along with examples of how it can be used.

OPTIONS: This section provides a detailed list of the available options that can be used with the command or function.

EXAMPLES: This section provides examples of how the command or function can be used in various scenarios.

SEE ALSO: This section provides links to related commands or functions that you may find useful.



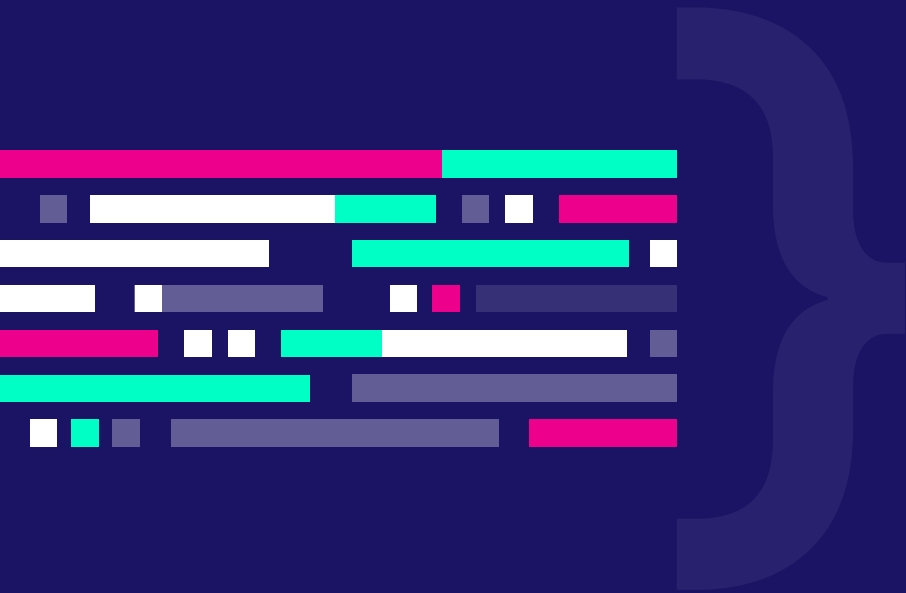
What are the section numbers for User commands, System calls and Library functions

User commands: This section contains the documentation for commands that can be run by users in a shell. For example, "ls" or "mkdir". Section 1 is for user commands.

System calls: This section contains the documentation for functions that can be called by user-space programs to interact with the kernel. These functions typically deal with low-level system resources such as memory or devices. Section 2 is for system calls.

Library functions: This section contains the documentation for the C library functions that are used to perform various tasks in C programming, such as string manipulation, memory allocation, and input/output. Section 3 is for library functions.

file



The `file` command is used to determine the file type of a file. It works by examining the contents of a file and making an educated guess about what type of file it is based on the patterns it finds.

The `file` command can be used with wildcards to determine the types of multiple files at once. For example:

```
$ file *.txt
file1.txt: ASCII text
file2.txt: UTF-8 Unicode text
file3.txt: empty
```



03

Quiz questions



04

Hands on lab Practice



Have a Question
Leave a Comment!



Share

To let the others know more



Subscribe

To stay updated with latest
videos



Thanks