What is an Exit Code?

An **exit code** (**exit status**) is a numeric value returned by a command or script to indicate its execution status.

- A **successful** command returns `0`.
- A **failure** returns a **non-zero** value (1-255).

Checking Exit Codes

The special variable **`$?`** holds the exit status of the last executed command.

Example:

```
ls /nonexistent_directory
echo "Exit code: $?"
```

Output:

```
ls: cannot access '/nonexistent_directory': No such file or directory
Exit code: 2
```

`ls` failed, so `$?` contains `2`.

3. Common Exit Codes

Exit Code	Meaning
`0`	Success (no error)
`1`	General error (catch-all)
`2`	Misuse of shell builtins
`126`	Command invoked but cannot execute
`127`	Command not found
`128`	Invalid exit argument
`130`	Script terminated with `Ctrl+C` (SIGINT)
`137`	Process killed (SIGKILL)
`255`	Exit status out of range

4. Using `exit` in Scripts

You can manually set an exit code using `exit <code>`.

Example:

```
#!/bin/bash
echo "Script doing something"
exit 5
```

Checking the exit code:

```
./script.sh
echo $?
```

Using Exit Codes in Conditional Statements

Ex: Handling Errors

```
#!/bin/bash
mkdir /root/test_directory 2>/dev/null

if [ $? -ne 0 ]; then
    echo "Failed to create directory. Permission denied."
    exit 1
fi
echo "Directory created successfully."
```

`set -e` for Automatic Exit on Error

The `set -e` option makes a script **exit immediately** if any command fails.

Example:

```
#!/bin/bash
set -e
cp nonexistent_file /tmp
echo "This will not execute if the copy fails"
```

Output (if `nonexistent_file` does not exist):

```
cp: cannot stat 'nonexistent_file': No such file or directory
```

The script exits without executing the last `echo` command.

7. `trap` for Cleanup on Exit

You can use `trap` to execute commands before the script exits.

Example:

```
#!/bin/bash
trap 'echo "Cleaning up before exit"; rm -f temp.txt' EXIT

echo "Creating temp.txt..."
touch temp.txt

exit 0  # Script exits here, triggering the trap
```

Summary

- **Exit codes indicate success (`0`) or failure (`non-zero`).**
- **Use `$?` to check the last command's exit status.**
- **Use `exit <code>` to set a script's exit status.**
- **Use `set -e` to exit on failure automatically.**
- **Use `trap` to run cleanup commands before exiting.**