SHELL SCRIPTING

Ahmet Sefa Çetin

30.07.2024

i2i Systems

Overview

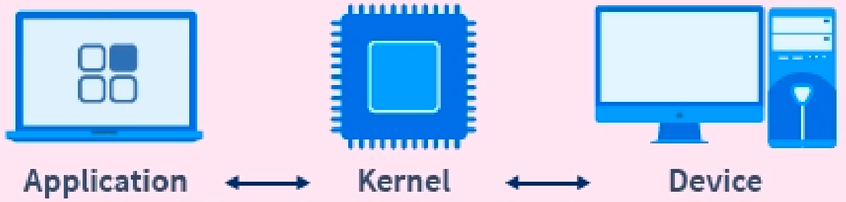
- What is Kernel?
- What is Shell?
- Command Line Shell
- Graphical Shells
- What is a Terminal?
- Shell Scripting
- Examples of Shell Scripting

WHAT IS KERNEL ?

The kernel is the core component of an operating system, managing hardware and system resources.

It acts as a bridge between applications and hardware, ensuring efficient communication and operation.

Key roles include resource management, process scheduling, and enforcing security.

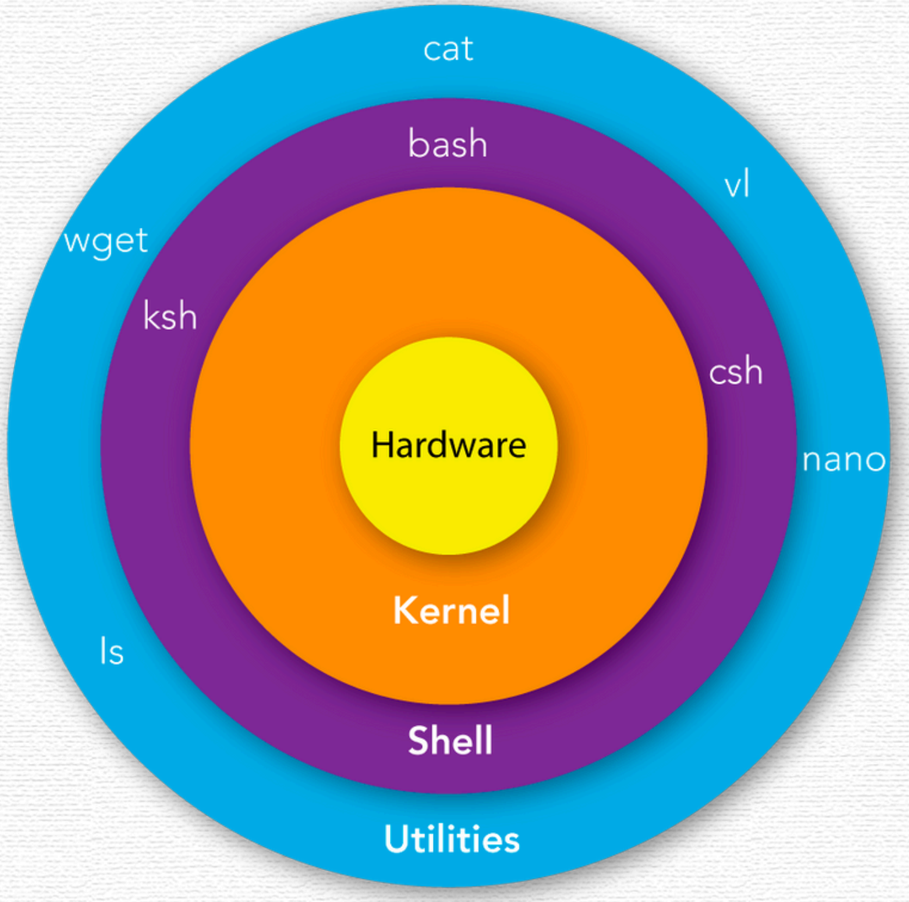


WHAT IS SHELL ?

A shell is a user program that provides an interface to access operating system services.

It interprets and executes user commands, converting them into a format the kernel can understand.

Shells can be command-line interfaces (CLI) or graphical user interfaces (GUI).

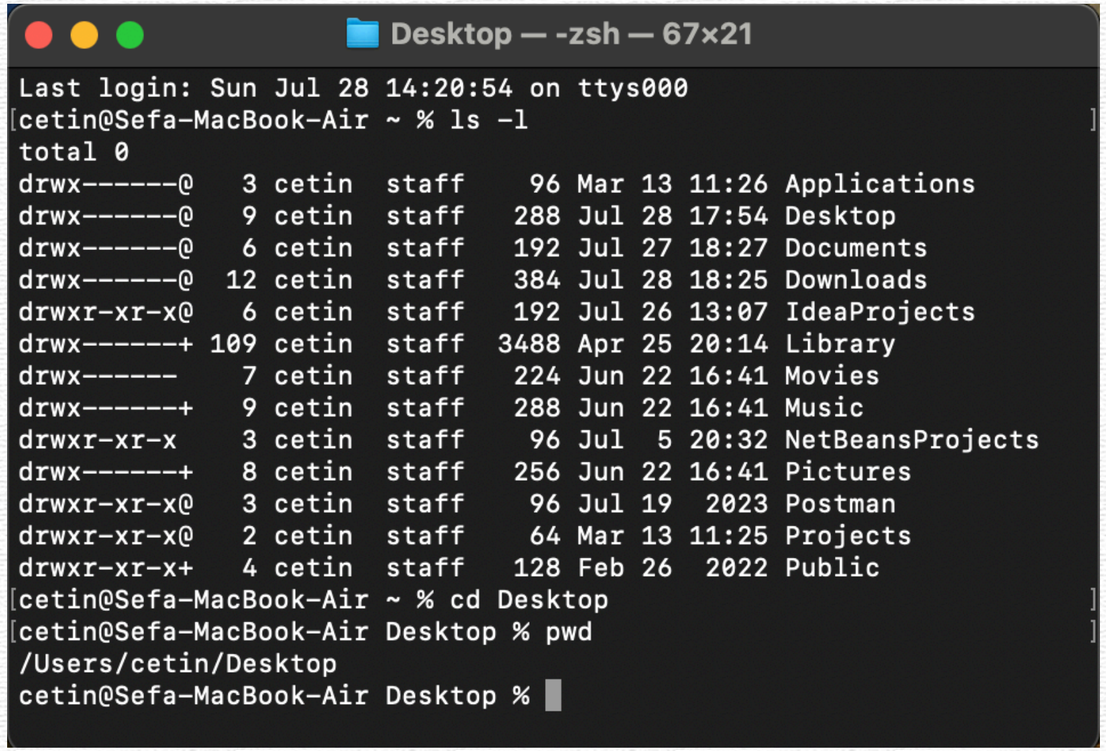


COMMAND LINE SHELL

Accessed via Terminal in Linux/macOS or Command Prompt in Windows.

Users type commands (e.g., cat, ls) that are executed and displayed.

Powerful for automation: commands can be stored in batch files (Windows) or shell scripts (Linux/macOS).



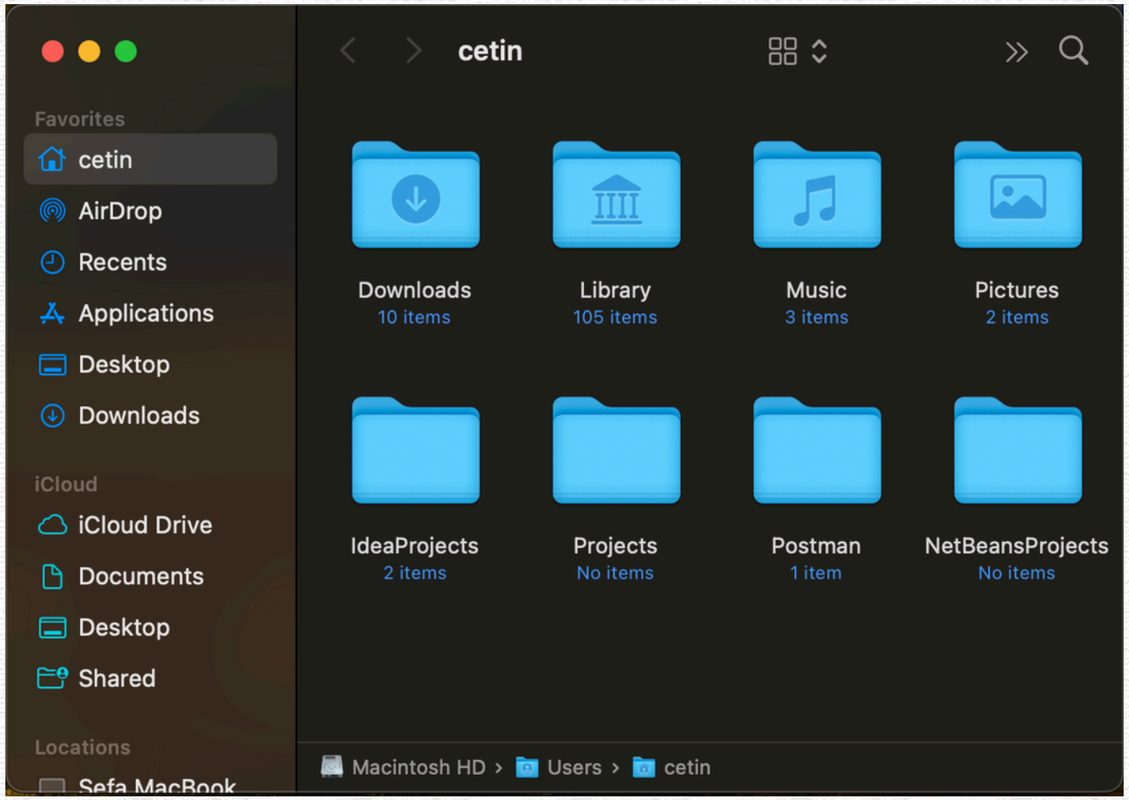
GRAPHICAL SHELL

Provides a graphical user interface (GUI) for interacting with programs.

Allows operations like opening, closing, moving, and resizing windows, and switching focus.

Users interact through visual elements, not commands.

Examples: Windows OS, Ubuntu OS.



TYPES OF SHELLS LINUX

BASH (Bourne Again SHell): The most widely used shell, default in Linux and macOS, also installable on Windows.

CSH (C SHell): Syntax and usage similar to the C programming language, with history and job control features.

KSH (Korn SHell): Basis for POSIX Shell standards, combining features of Bourne and C Shells, known for scripting.

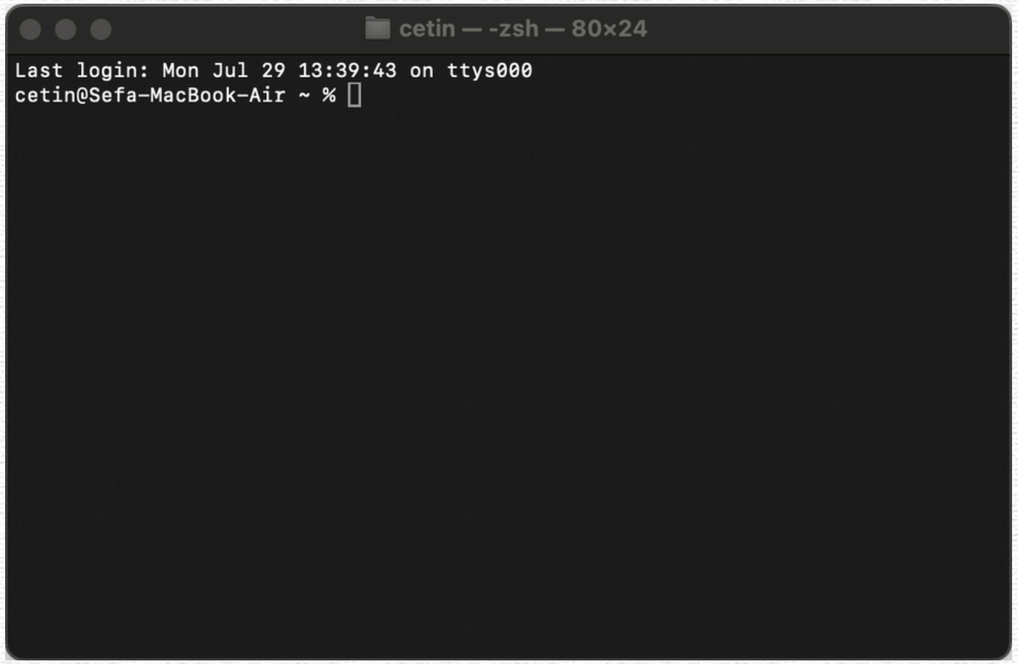
Each shell performs the same tasks but uses different commands and functions.

WHAT IS TERMINAL ?

A program that provides an interface to access the shell.

Allows users to enter commands and see the output in a text-based interface.

Used to execute scripts that automate complex tasks.



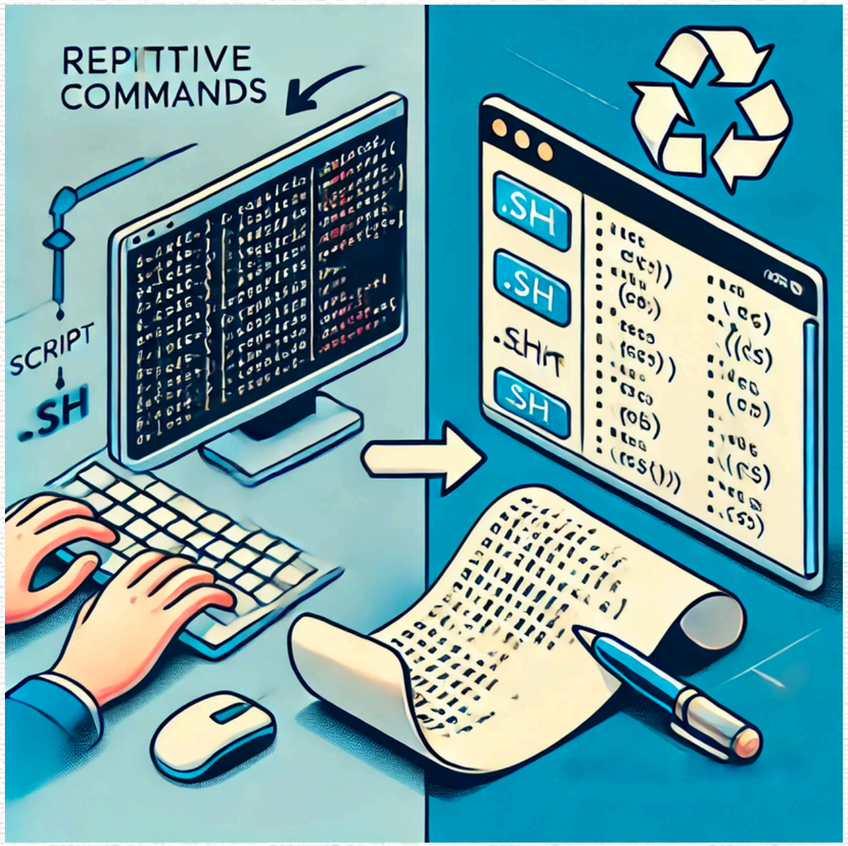
SHELL SCRIPTING

Shells are interactive, accepting and executing user commands.

To avoid repetitive routine tasks, commands can be written in files called Shell Scripts or Shell Programs.

Shell scripts, saved with a .sh extension (e.g., asd.sh), automate tasks.

Their syntax is similar to languages like Python or C/C++, making them easy to learn.



WHY DO WE NEED SHELL SCRIPTS ?

Avoid Repetitive Work: Automate tasks to save time and reduce errors.

Routine Backups: System admins use scripts for regular, automated backups.

System Monitoring: Automate the monitoring of system performance and status.

Extend Functionality: Add new features and capabilities to the shell environment.

SHEBANG (#!) IN SHELL SCRIPTS

The shebang (#!) at the beginning of a script indicates the interpreter that should be used to execute the script.

It tells the system which shell or interpreter should interpret the script's commands.

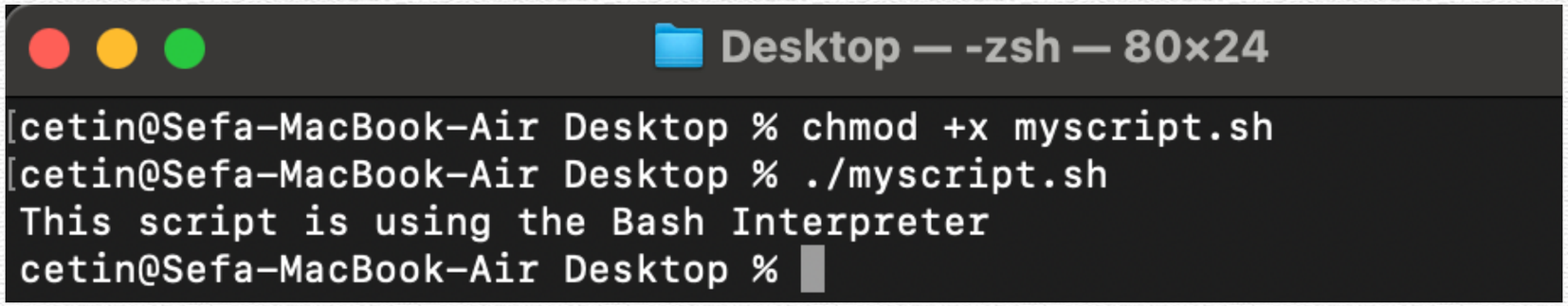
myscript.sh

```
#!/bin/bash

echo "This script is using the Bash Interpreter"
```

RUNNING A SHELL SCRIPT

Make sure the script file has executable permissions using the chmod command.

Execute the script using its filename.



EXAMPLE

A shell script that takes a user's name as input and greets them.

myscript.sh

```
#!/bin/bash
# Ask the user for their name
echo “What’s your name?”
read name
# Greet the user
echo “Hello, $name! Nice to meet you.”
```

Desktop — -zsh — 80×24

```
cetin@Sefa-MacBook-Air Desktop % ./myscript.sh
“What’s your name?”
sefa
“Hello, sefa! Nice to meet you.”
cetin@Sefa-MacBook-Air Desktop %
```

EXAMPLE

A script that checks if a file named “sefa.txt” exists in the current directory.

myscript.sh

```
#!/bin/bash
file="sefa.txt"
# Check if the file exists
if [ -e "$file" ]; then
echo "File exists: $file"
else
echo "File not found: $file"
fi
```

Desktop — -zsh — 80×24

```
cetin@Sefa-MacBook-Air Desktop % ./myscript.sh
File exists: sefa.txt
cetin@Sefa-MacBook-Air Desktop %
```

EXAMPLE

Command-line arguments are values provided to a script when it’s executed and can be accessed using special variables like $1, $2, etc.

myscript.sh

```
#!/bin/bash

echo "Script name: $0"
echo "First argument: $1"
echo "Second argument: $2"
```

Desktop — -zsh — 80×24

```
cetin@Sefa-MacBook-Air Desktop % ./myscript.sh ahmet sefa
Script name: ./myscript.sh
First argument: ahmet
Second argument: sefa
cetin@Sefa-MacBook-Air Desktop %
```

EXAMPLE

A shell script that calculates the sum of integers from 1 to N using a loop.

myscript.sh

```
#!/bin/bash

echo "Enter a number (N):"
read N
sum=0
for (( i=1; i<=$N; i++ )); do
sum=$((sum + i))
done
echo "Sum of integers from 1 to $N is: $sum"
```

Desktop — -zsh — 80×24

```
cetin@Sefa-MacBook-Air Desktop % ./myscript.sh
Enter a number (N):
6
Sum of integers from 1 to 6 is: 21
cetin@Sefa-MacBook-Air Desktop %
```

EXAMPLE

A function in a shell script that calculates the factorial of a given number.

myscript.sh

```
#!/bin/bash
# Define a function to calculate factorial
calculate_factorial() {
num=$1
fact=1
for ((i=1; i<=num; i++)); do
fact=$((fact * i))
done
echo $fact
}

# Prompt the user to enter a number
echo "Enter a number: "
read input_num

# Call the calculate_factorial function with the input number
factorial_result=$(calculate_factorial $input_num)

# Display the factorial result
echo "Factorial of $input_num is: $factorial_result"
```

Desktop — -zsh — 80×24

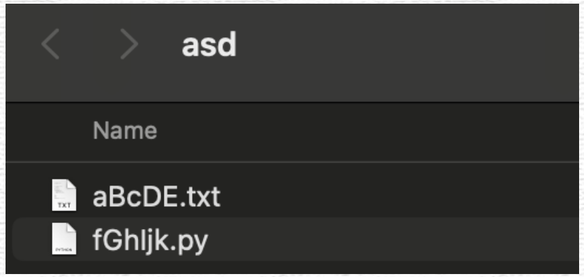
```
cetin@Sefa-MacBook-Air Desktop % ./myscript.sh
Enter a number:
4
Factorial of 4 is: 24
cetin@Sefa-MacBook-Air Desktop %
```

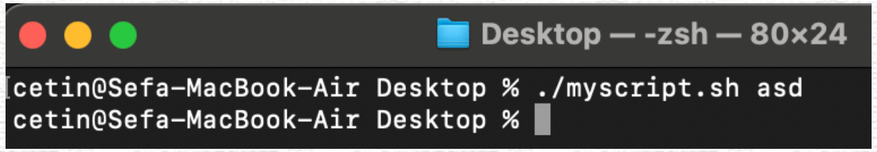
EXAMPLE

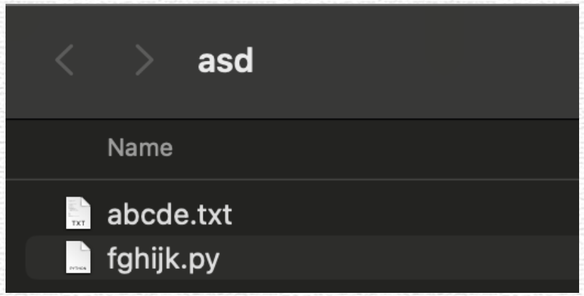
A shell script that converts all filenames in a directory to lowercase.

myscript.sh

```
directory="$1"
if [ -z "$directory" ]; then
echo "Usage: $0 <directory>"
exit 1
fi

if [ ! -d "$directory" ]; then
echo "Error: '$directory' is not a valid directory."
exit 1
fi

cd "$directory" || exit 1

for file in *; do
if [ -f "$file" ]; then
newName=$(echo "$file" | tr 'A-Z' 'a-z')
[ "$file" != "$newName" ] && mv "$file" "$newName"
fi
done
```







References

1. https://www.geeksforgeeks.org/introduction-linux-shell-shell-scripting/
2. https://www.geeksforgeeks.org/shell-script-examples/
3. https://www.tutorialspoint.com/unix/index.htm
4. https://chatgpt.com/

Thank you!

Do you have any questions ?

linkedin.com/in/ahmetsefacetin

ahmetsefacetin@outlook.com

i2i Systems