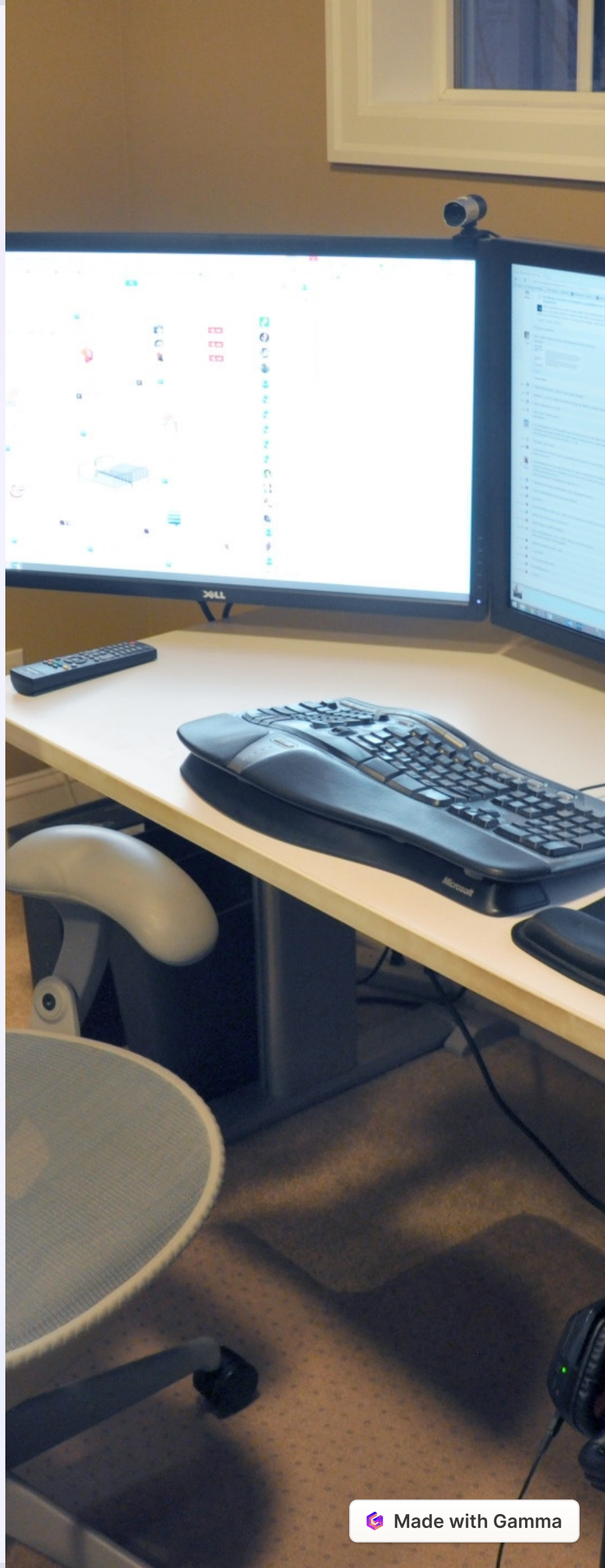
Welcome to Shell Scripting Basics

In this presentation, we will explore the fundamentals of shell scripting and learn how to solve common problems using shell scripts. Let's dive in!

by hemanth m



List All Files in the Current Directory

Problem 1: List All Files in the Current Directory is a handy shell script that helps you quickly obtain a list of all files in the current directory.

1 One-line Definition

Create a shell script called **list_files.sh** that, when executed, displays a list of all files in the current directory.

2 Code

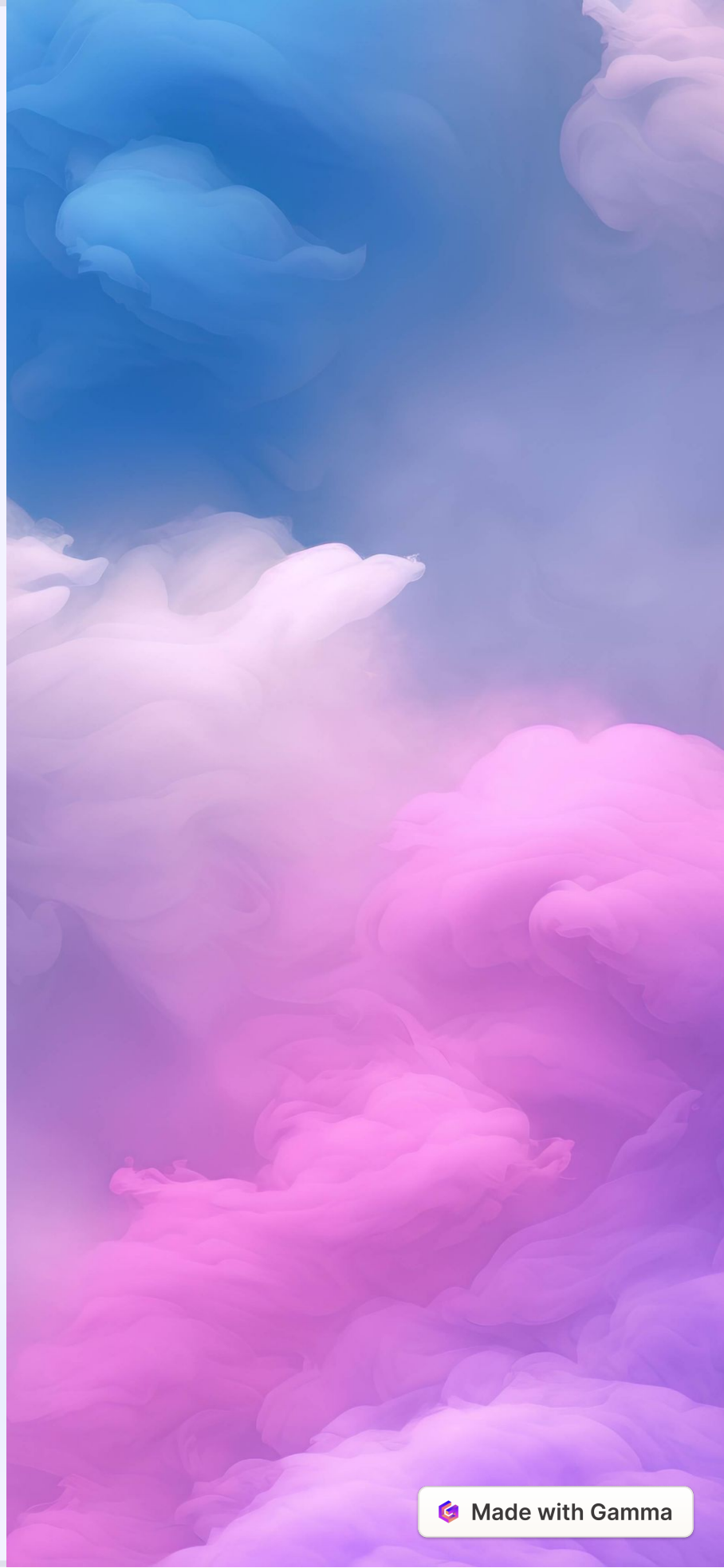
```
#!/bin/bash
# List all files in the current directory for file in *;
 do if [ -f "$file" ]; then echo "File: $file" elif [ -d "$file" ];
then echo "Directory: $file" fi done
```

3 Sample Input

(No user input required for this script)

4 Expected Output

```
File: file1.txt Directory: subdirectory File: file2.txt
```



Check If Arguments are Files or Directories

Problem 2: Check If Arguments are Files or Directories is a useful shell script that helps you determine whether an argument passed to the script is a file, a directory, or if it doesn't exist.

1 One-line Definition

Create a shell script called **check_files.sh** that takes any number of file or directory names as arguments and reports whether each argument is a file, a directory, or if it doesn't exist.

2 Code

```
#!/bin/bash
# Check if no arguments were
provided
if [ $# -eq 0 ];
then
echo "No arguments provided."
exit 1 fi
# Loop through all provided
arguments for arg in "$@";
 do
 if [ -e "$arg" ];
 then
 if [ -f "$arg" ]; then echo "$arg is a
file."
elif [ -d "$arg" ]; then echo "$arg is a
directory." else echo "$arg exists
but is neither a file nor a directory."
fi else echo "$arg does not exist." fi
done
```

3 Sample Input

(You can provide file and directory names as arguments)

4 Expected Output

file1.txt is a file. directory1 is a directory. non-existent-file does not exist.

Improve Your Shell Scripting Skills

Efficiency

Learn techniques to optimize your shell scripts for efficiency and faster execution.

Error Handling

Discover strategies to handle errors and exceptions gracefully in your shell scripts.

Advanced Techniques

Explore advanced topics like regular expressions, command substitution, and pipelines.

Shell Scripting Best Practices

1 Modularity

Break down your scripts into reusable functions for better organization and maintainability.

2 Comments

Document your code with clear and concise comments to improve readability and understanding.

3 Testing

Implement robust testing methodologies to ensure the reliability and correctness of your scripts.