Here are some basic projects for a beginner in Bash scripting using Windows OS:

1. Hello World Program: Write a script that prints "Hello, World!" to the console.
2. File Management: Write a script that creates a new directory, copies a file from one location to another, and deletes the original file.
3. System Information: Write a script that displays system information like CPU usage, memory usage, and disk space.
4. Backup Script: Write a script that backs up a directory by compressing it and copying it to another location.
5. File Renaming: Write a script that renames all the files in a directory with a particular extension to a new name.
6. Simple Calculator: Write a script that performs basic arithmetic operations like addition, subtraction, multiplication, and division.
7. Text Processing: Write a script that reads a text file and performs some processing, like finding and replacing a word or counting the occurrences of a word.
8. Password Generator: Write a script that generates a random password and saves it to a file.
9. Automated Task: Write a script that automates a repetitive task like running a backup, checking for updates, or sending an email.
10. Command-Line Tool: Write a script that can be used as a command-line tool to perform a specific task, like converting file formats, searching for files, or downloading a file from the internet.

Remember to start with simple scripts and gradually increase the complexity as you gain more experience. Good luck!

**Project 1 – Hello world**

Great! Let's get started with Project 1: Hello World Program.

In Bash scripting, we can use the echo command to print a message to the console. Open a text editor like Notepad or Notepad++ and type the following code:

#!/bin/bash

echo "Hello, World!"

Save the file with a .sh extension (e.g. hello.sh).

Now open the command prompt and navigate to the directory where you saved the hello.sh file. Then type the following command to run the script:

bash hello.sh

This should print the message "Hello, World!" to the console.

Congratulations, you have successfully created your first Bash script!

**Project 2: File Management**

Great! Let's move on to Project 2: File Management.

In this project, we will write a Bash script that creates a new directory, copies a file from one location to another, and deletes the original file.

Here's the code for the script:

#!/bin/bash

# create a new directory

mkdir mydir

# Copy a file from one location to another

cp /path/to/myfile.txt mydir/

# delete the original file

rm /path/to/myfile.txt

Replace /path/to/myfile.txt with the actual path to the file you want to copy.

Save the script with a .sh extension (e.g. file\_management.sh) and navigate to the directory where you saved it.

Then, run the script with the following command:

bash file\_management.sh

This should create a new directory called mydir, copy the specified file to the new directory, and delete the original file.

Congratulations, you have successfully created a Bash script for file management!

**Project 3: System Information.**

In this project, we will write a Bash script that displays system information like CPU usage, memory usage, and disk space.

Here's the code for the script:

#!/bin/bash

# Display CPU usage

echo "CPU Usage:"

top -bn1 | grep load | awk '{printf "%.2f%%\n", $(NF-2)}'

# Display memory usage

echo "Memory Usage:"

free | awk 'NR==2{printf "%.2f%%\n", $3\*100/$2}'

# Display disk space

echo "Disk Space:"

df -h | awk '$NF=="/"{printf "%s\n", $5}'

Save the script with a .sh extension (e.g. system\_info.sh) and navigate to the directory where you saved it.

Then, run the script with the following command:

bash system\_info.sh

This should display the CPU usage, memory usage, and disk space information.

Congratulations, you have successfully created a Bash script to display system information!

**Project 4 – Backup Scripting**

In this project, we will create a Bash script that creates a backup of a specified directory by compressing its contents into a tarball (.tar.gz file). Here's the code for the script:

#!/bin/bash

# Get the current date and time

current\_date=$(date +"%Y-%m-%d\_%H-%M-%S")

# Prompt the user for the directory to back up

echo "Enter the directory to back up:"

read backup\_directory

# Check if the specified directory exists

if [ ! -d "$backup\_directory" ]; then

echo "Error: Directory does not exist."

exit 1

fi

# Create a tarball of the specified directory

tar -czf "${backup\_directory}\_${current\_date}.tar.gz" "$backup\_directory"

echo "Backup complete!"

Save the script with a .sh extension (e.g. backup\_script.sh) and navigate to the directory where you want to create the backup.

Then, run the script with the following command:

bash backup\_script.sh

The script will prompt you to enter the directory to back up. After you enter the directory name, the script will create a tarball of the directory's contents and name it with the directory name and current date and time.