Bash Scripting Exercises & Solutions

Exercise 1: Create and Use Local Variables

Write a script `vars.sh` that:
1. Defines a variable `NAME` with your name.
2. Defines a variable `AGE` with your age.
3. Prints both variables.

Solution:
#!/bin/bash
NAME="Francis"
AGE=25
echo "My name is $NAME and I am $AGE years old."

Run:
chmod +x vars.sh
./vars.sh

Expected Output:
My name is Francis and I am 25 years old.

Exercise 2: Modify Environment Variables

Run the following command in your terminal:
echo $HOME

Modify your `$PATH` to include a new directory `~/scripts`:
export PATH=$PATH:~/scripts

Verify:
echo $PATH

Exercise 3: Create and Manipulate Files

1. Create a directory `test_dir`.
2. Inside `test_dir`, create `file1.txt`, `file2.txt`, `file3.txt`.
3. Copy `file1.txt` to `backup_file.txt`.
4. Rename `file2.txt` to `renamed_file.txt`.

5. Delete `file3.txt`.

Solution:
mkdir test_dir
cd test_dir
touch file1.txt file2.txt file3.txt
cp file1.txt backup_file.txt
mv file2.txt renamed_file.txt
rm file3.txt
ls -l

Exercise 4: Modify File Permissions

1. Create `script.sh`.
2. Make it executable for the owner only.

Solution:
touch script.sh
chmod 700 script.sh
ls -l script.sh

Expected Output:
-rwx------ 1 user user 0 script.sh

Exercise 5: Redirect Output to a File

Run:
ls -l > directory_list.txt

View file:
cat directory_list.txt

Exercise 6: Append to a File

Add the current date to `log.txt`:
date >> log.txt

Run twice and view:
cat log.txt

Exercise 7: Count Number of Files in a Directory

Use:
ls | wc -l

Exercise 8: Find the Largest File in a Directory

Use:
ls -l | sort -k5 -nr | head -1

Exercise 9: Search for a Word in a File

Create `sample.txt` with content:

This is a test file.
Bash scripting is fun.
Regular expressions are powerful.

Search for "Bash":
grep "Bash" sample.txt

Expected Output:
Bash scripting is fun.

Exercise 10: Count the Occurrences of a Word

Count "is" in `sample.txt`:
grep -o "is" sample.txt | wc -l

Expected Output:
2

Exercise 11: Find Lines That Start with a Letter

Find lines starting with "B":
grep "^B" sample.txt

Expected Output:
Bash scripting is fun.

Exercise 12: Find Valid Emails

Create `emails.txt`:
alice@example.com

bob_at_example.com
charlie@mtu.ie

Find valid emails:
grep -E "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}" emails.txt

Expected Output:
alice@example.com
charlie@mtu.ie

Exercise 13: Find Phone Numbers

Find numbers in `phones.txt`:
grep -E "^[0-9]{3}-[0-9]{3}-[0-9]{4}$" phones.txt

Expected Output:
123-456-7890

Exercise 14: Find Empty Lines

Find empty lines in `sample.txt`:
grep "^$" sample.txt

Exercise 15: Display Non-Matching Lines

Show lines **not** containing "test":
grep -v "test" sample.txt

Expected Output:
Bash scripting is fun.
Regular expressions are powerful.

Challenge: Log Analysis Script

Create `system.log`:
[INFO] System started.
[ERROR] Disk space low.
[WARNING] CPU temperature high.
[INFO] User logged in.
[ERROR] Network failure detected.

Write `log_analysis.sh` to:
1. Count `ERROR` messages.
2. Count `WARNING` messages.
3. Display all `INFO` messages.

Solution:
```
#!/bin/bash
echo "ERROR Count: $(grep -c 'ERROR' system.log)"
echo "WARNING Count: $(grep -c 'WARNING' system.log)"
echo "INFO Messages:"
grep 'INFO' system.log
```

Run:
```
chmod +x log_analysis.sh
./log_analysis.sh
```

Expected Output:
```
ERROR Count: 2
WARNING Count: 1
INFO Messages:
[INFO] System started.
[INFO] User logged in.
```