Session - 15

Shell Scripting – Part 4: Arrays, Functions, Logging, and Error Handling

This part introduces **arrays**, **advanced functions**, and techniques like **logging**, **error handling**, and **debugging** in shell scripts.

🔹 1. Arrays in Shell

Shell arrays let you store multiple values in one variable.

Declare and Access:

```
fruits=("apple" "banana" "cherry")

echo "${fruits[0]}"   # apple
echo "${fruits[1]}"   # banana
```

Loop Through Array:

```
for fruit in "${fruits[@]}"; do
  echo "$fruit"
done
```

- **2. Functions with Arguments and Return**

Define and Call:

```
greet() {
  echo "Hello, $1"
}

greet "shining"
```

Returning Status Code:

```
add() {
  result=$(( $1 + $2 ))
  return $result
add 5 3
echo "Sum: $?"
```

- **3. Logging in Shell Script**

Simple Logger:

```
logfile="mylog.txt"
log() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" >> "$logfile"
}
log "Script started"
```

- **4. Error Handling**

Check If Command Succeeds:

```
if cp source.txt backup/; then
  echo "Copy successful"
else
  echo "Copy failed"
fi
```

Use Exit Codes:

```
#!/bin/bash

[ -f myfile.txt ] || { echo "File not found"; exit 1; }
```

- **5. Set Script to Exit on Error Automatically**

```
set -e  # Exit if any command fails
```

You can also use:

```
set -u  # Treat unset variables as error
set -x  # Debug: print commands as they run
```

- **6. Trap and Cleanup**

To perform actions on exit or error:

```
cleanup() {
```

```
  echo "Cleaning up..."
  rm -f temp.txt
}

trap cleanup EXIT
```

- **7. Mini Project – Script Logger**

```
!/bin/bash
logfile="status.log"

log() {
  echo "$(date): $1" >> "$logfile"
}

log "Script started"
if ls /some/path; then
  log "Directory listed successfully"
else
  log "Failed to list directory"
fi
```

✅ Summary of Part 4

Feature	Use
Arrays	arr=(a b c), ${arr[1]}
Functions	myfunc() { ... }
Logging	Use date and >> log.txt
Error handling	- if, exit, `
Debugging	set -e, set -x