Unix Shell Programming - Day 2

By Dhandapani Yedappalli Krishnamurthi Sep 30, 2025

1️⃣ Local, Environment Variables & `export`

📘 Variables store values. Local vars exist only in script/session.

`export` makes them available to child processes.

```
#!/bin/bash
city="Chennai"          # Local variable
export state="TamilNadu" # Environment variable

echo "City: $city"
echo "State: $state"
```

🔍 `state` is accessible to subprocesses, but `city` is not.

2️⃣ Input/Output, `read` Command

```
#!/bin/bash
read -p "Enter your name: " name
echo "Welcome $name, have a great day!"
```

🔍 Reads input from user and prints greeting.

3️⃣ `expr`, `let`, `$((…))`

```
#!/bin/bash
a=10
b=5

echo "Sum: $(expr $a + $b)"
let c=a*b
echo "Product: $c"
echo "Power: $((a ** b))"
```

🔍 Arithmetic using different methods.

4️⃣ Command Substitution: `` `...` `` and `$(...)`

```
#!/bin/bash
today='date'
users=$(who | wc -l)
```

```
echo "Today is $today"
echo "Logged in users: $users"
```

🔍 Both forms capture command output into variables.

5️⃣ $0, $1, $#, $*

```
#!/bin/bash
echo "Script name: $0"
echo "First arg: $1"
echo "Total args: $#"
echo "All args: $*"
```

Run:

```
./script.sh Dhandapani Raj
```

👉 Output:

```
Script name: ./script.sh
First arg: Dhandapani
Total args: 2
All args: Dhandapani Raj
```

6️⃣ if, else, elif

```
#!/bin/bash
marks=75

if [ $marks -ge 90 ]; then
   echo "Grade: A"
elif [ $marks -ge 60 ]; then
   echo "Grade: B"
else
   echo "Grade: C"
fi
```

7️⃣ String, Numeric, File Checks

```
#!/bin/bash
name="Ravi"
age=20
file="students.txt"

[ "$name" = "Ravi" ] && echo "Name matches"
[ $age -gt 18 ] && echo "Adult"
[ -f $file ] && echo "File exists"
```

8️⃣ break & continue

```
#!/bin/bash
for num in {1..5}
do
   if [ $num -eq 3 ]; then
      continue  # skip 3
   fi
   if [ $num -eq 5 ]; then
      break  # stop loop
   fi
   echo "Number: $num"
done
```

9️⃣ Functions

```
#!/bin/bash
greet() {
   echo "Hello $1 from $2!"
}

greet "Anitha" "Hyderabad"
```

🔟 Arrays: Declaring, Iterating, Accessing

```bash
#!/bin/bash
names=("Arun" "Priya" "Lakshmi")

echo "First: ${names[0]}"

for n in "${names[@]}"; do
   echo "Student: $n"
done
```

1️⃣1️⃣ File Handling

```bash
#!/bin/bash
# Read file line by line
while read line; do
   echo "Line: $line"
done < students.txt

# Concat files
cat file1.txt file2.txt > combined.txt

# Touch file
touch newfile.txt

# File size & record count
ls -lh newfile.txt
```

```
wc -l newfile.txt
```

1️⃣2️⃣ Advanced awk & sed

```
# Print 2nd column (names) from CSV
awk -F, '{print $2}' employees.csv

# Replace Chennai with Bengaluru
sed -i 's/Chennai/Bengaluru/g' employees.csv
```

1️⃣3️⃣ head, tail, wc

```
head -5 students.txt   # First 5 lines
tail -3 students.txt   # Last 3 lines
wc -l students.txt     # Line count
wc -w students.txt     # Word count
```

1️⃣4️⃣ SFTP Basics

```
sftp user@server
sftp> put localfile.txt
sftp> get remotefile.txt
```

1️⃣5️⃣ Connect to DB from Shell Script

```
#!/bin/bash
psql -h localhost -U postgres -d AdventureWorks
-c "SELECT COUNT(*) FROM sales.salesorderheader;"
```

🔍 Executes SQL command directly from script.

1️⃣6️⃣ Recap & Best Practices

- Always use **meaningful variable names**
- Quote variables → `"$var"`
- Add error handling
- Test scripts with `set -x`

🎯 Real-Life Example: Salary Processing Script

```
#!/bin/bash
# salary.sh

while IFS=, read name basic hra da
do
   total=$((basic + hra + da))
   echo "Employee: $name | Salary: $total"
```

```
done < salaries.csv
```

👉 If `salaries.csv` contains:

```
Ravi,20000,5000,3000
Meena,25000,6000,4000
```

Output:

```
Employee: Ravi | Salary: 28000
Employee: Meena | Salary: 35000
```