ChatGPT

Shell Programming - Unit 4: In-Depth Concepts and Applications

---

1. Introduction to the Shell

A **shell** is a command-line interpreter that facilitates communication between the user and the operating system. It reads user input, interprets commands, and executes them by interacting with the system kernel.

---

2. Major Types of Shells

- **Bourne Shell (sh)** – An early and widely used shell.
- **Bourne Again Shell (bash)** – Default shell on most Linux distributions.
- **C Shell (csh)** – Syntax similar to the C programming language.
- **Korn Shell (ksh)** – Combines features from both Bourne and C shells.
- **Z Shell (zsh)** – Advanced shell with rich features and customization.

---

3. Identifying the Default Shell

To identify the shell currently in use:

```
echo $SHELL
```

To list all supported shells on your system:

```
cat /etc/shells
```

---

4. Shell Variables

Shell variables are placeholders used to store values temporarily for use in scripts or the command line.

```
name="Sahil"
echo $name
```

---

5. Reserved Keywords in Shell Scripting

Shell scripting utilizes predefined keywords such as: `if`, `then`, `else`, `fi`, `case`, `esac`, `for`, `while`, `do`, `done`, which help define the logic and control flow of the script.

---

6. Environment Variables

These are system-wide variables that provide configuration settings for the shell session:

```
echo $HOME  # Displays user home directory
echo $PATH  # Shows executable search paths
```

To define an environment variable:

```
export MYVAR="Test"
```

---

7. Writing a Shell Script

Shell scripts are plain text files containing a sequence of commands.

```
#!/bin/bash
echo "Hello, World!"
```

Save the file as `script.sh`, then:

```
chmod +x script.sh
./script.sh
```

---

8. Passing Parameters to a Shell Script

Shell scripts can accept arguments at runtime using positional parameters:

```
#!/bin/bash
echo "First parameter: $1"
echo "Second parameter: $2"
```

Execute as:

```
./script.sh arg1 arg2
```

---

9. Positional Parameters and the Shift Command

Positional parameters are accessed as `$1`, `$2`, etc. The `shift` command discards the current `$1` and reassigns the next parameter:

```
#!/bin/bash
echo $1
shift
echo $1
```

---

10. Loop Structures

For Loop

Used to iterate over a list:

```
for i in 1 2 3; do
echo $i
done
```

While Loop

Repeats a block of code while a condition is true:

```
count=1
while [ $count -le 5 ]; do
echo $count
count=$((count+1))
done
```

Until Loop

Executes until the condition becomes true:

```
count=1
until [ $count -gt 5 ]; do
```

```
echo $count
count=$((count+1))
done
```

---

11. Conditional Statements - If

```
if [ $1 -gt 10 ]; then
echo "Greater than 10"
else
echo "Less than or equal to 10"
fi
```

---

12. Case Statement

Simplifies multiple conditional branches:

```
read -p "Enter a number: " num
case $num in
1) echo "One";;
2) echo "Two";;
*) echo "Other number";;
esac
```

---

13. Scripting in Python within Linux

Python scripts can be invoked just like shell scripts and used for advanced logic.

```
# myscript.py
print("Hello from Python")
```

Run using:

```
python3 myscript.py
```

Practice Questions for Mastery:

1. Develop a script using a `for` loop to print numbers from 1 to 10.

2. Write a script to determine whether a number is even or odd.
3. Accept two command-line arguments and output their sum.
4. Write a script to check the existence of a file.
5. Implement a menu-driven calculator using a `case` statement.

Let me know if you'd like this rewritten version exported as a PDF or extended with more examples.