DIPLOMA WALLAH

(Your One Stop Hub For Diploma Resources)

OPERATING SYSTEM AND ADMINISTRATION

Complete Notes Based on Full Syllabus

- Diploma Engineering

4th Semester

www.diplomawallah.in

Notes prepared by **Sangam**

Unit-07

Sagar Sangam

Date
Page

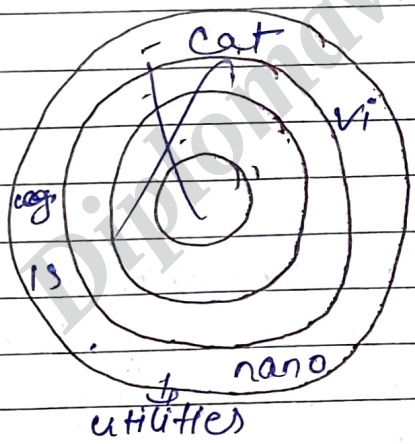
Shell programming

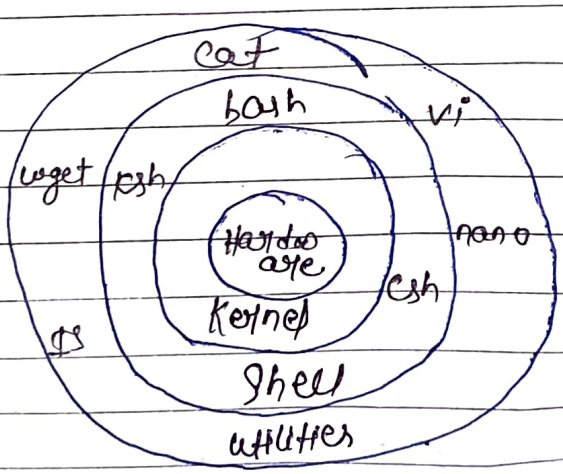
Shell

A Shell is a special user program that provides an interface for the user to use OS services.

Shell accepts human-readable commands from users and convert them into something which the Kernel can understand.

It is a command language interpreter that executes commands read from input devices such as keyboard or from files.





- Kernel :- controls hardware, manage resources, and run programs. User cannot access hardware
- Shell : Allow users to interact with Kernel utilities.

Types of Shell in Linux

1) The C Shell
denoted as csh
- Bill Joy created it at the university of California at Berkely.
It incorporated features such as aliases and command history.
Feature
- C like syntax
- Command aliases
- Command history
- Built in arithmetic & expression

use : Scripting and interactive command execution.
path : /bin/csh

A. C like Syntax
Command & Control structure (if, while, switch) are similar to C programming language.
- make scripting easier for programmers familiar with C.

```
#! /bin/csh
set x=5
if ($x > 0) then
    echo "x is positive."
endif
```

B. Command history
we can recall previous commands using the up-arrow.

4. ls -l
4. !! #Repeats last command (ls-l)

c. Aliases
- Shortcuts for long commands.
- Makes command execution faster

alias ll 'ls-l'
ll # Runs 'ls-l'

d. Built-in Arithmetic
- Support arithmetic operations directly in shell.

#!/bin/csh
set a=10
set b=20
set c='expr $a + $b'
echo "sum is $c"

Script example
#!/bin/csh

@ i=1
while ($i<=10)
if ($i%2==0) then
echo $i
endif
@ i=$i+1
end

- @ is used for arithmetic operation.
- while loop to iterate
- if statement to check condition.

The Bourne Shell
- Bourne Shell - created by Bourne (AT&T Bell Labs).
- path: /bin/sh
- prompt: $ (non-root), # (root)

Mainly used for scripting and basic shell operations.

Features
- A simple and lightweight
- Scripting support —

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

- Variables

```
#!/bin/sh
name = "Sagar"
echo "Hello, $name"
```

- Conditional Statement —
  - Support if, else, elif

```
#!/bin/sh
a=10
if [$a -gt 5]
then
    echo "a is greater than 5."
else
    echo "a is less or equal to 5."
fi
```

- loops

support for while, until loops.

print 1 to 5
```
#!/bin/sh
for i in 1 2 3 4 5
do
    echo $i
done
```

F. Functions
you can define and call functions inside shell scripts.
Ex
```
#!/bin/sh
greet ()
{
    echo "Hello, $1"
}
greet "ABC"
```

* TENEX C shell (tcsh)
→ Improved C shell
path: /bin/tcsh
Features:
- Command line editing
- Auto-completion of commands
- Friendly interactive shell

Ex-
```
alias ll 'ls -l'
ll
```

* Korn shell (Ksh)
- Created by David Korn (AT&T Bell Labs)
- path: /bin/ksh
prompt: $ (non-root), # root
Feature:
- Superset of Bourne shell
- function, array arithmetic operation.

Ex
```
#/bin/ksh
arr=(1 2 3 4 5)
for i in ${arr[@]}
do
  echo $i
done
```

* Z shell (Zsh)
* Fish shell (Friendly Interactive shell)

* Shell Scripts
- A shell script is a file containing a sequence of commands that the shell can execute.
- Instead of typing commands one by one, we can write a script and run it automatically.

Ex - Simple script to print "Hello World"

```
echo "Hello world"
```

Shebang / Hashbang

- Shebang (#!) is the first line in a shell script.
- It tells the OS which shell to use to execute the scripts.

Syntax:-

```
#!/path/to/shell
```

Example:

```
#!/bin/sh       #Bourne shell
#!/bin/bash     #Bash shell
#!/bin/csh      #C shell
#!/bin/ksh      #Korn shell
#!/bin/zsh      #Z shell
```

Explanation:

- # is a comment in shell.
- ! after # tells OS this is an interpreter path.
- Without shebang, the script might run in the default shell, which can cause errors.

Creating a Basic Shell Script

1. open a text editor and write commands.
2. Save the file with .sh extension
3. Add shebang at the top
4. Give execute permission
5. Run the script

Ex - Hello world script

```
#!/bin/bash
use Bash                    # shebang tells OS to
echo "Hello World."         # Print message
```

Decision making

- If statement

```
if [condition]
then
    #commands
fi
```

Ex- check positive number

```
#!/bin/bash
echo "Enter a number:"
read num
if [ $num -gt 0 ]
then
  echo "Number is positive"
fi
```

If-else

```
#!/bin/bash
echo "Enter a number:"
read num
if [ $num -gt 0 ]
then
  echo "Number is positive"
else
  echo "Number is zero or negative."
fi
```

if-elif-else

```
#!/bin/bash
echo "Enter a number:"
read num
if [$num -gt 0]; then
    echo "positive"
elif [$num -lt 0]; then
    echo "Negative"
else
    echo "zero"
fi
```

B. case Statement

use: multiple choices / pattern matching.

```
case $variable in
    pattern1)
    commands;;
    pattern2)
        commands;;
    *)
    default commands;;
esac
```

Eg

```
#!/bin/bash
echo "Enter a number (1-7):"
read day
case $day in
1) echo "Sunday";;
2) echo "Monday";;
*) Echo "Invalid input";;
esac.
```

* Iterative Scripts

for loop

```
for var in list
do
  commands
done
```

Ex- 1 to 5

```
#!/bin/bash
for i in 1 2 3 4 5
do
  echo "number $i"
done
```

* using range

```
for i in {1..5}
do
  echo $i
done
```

* while loop

```
while [condition]
do
  commands
done
```

Ex - print 1 to 5

```
#!/bin/bash
i=1
while [$i -le 5]
do
  echo $i
  i=$((i+1))
done
```

Practice ~~string operation~~
String operation & file operation

String
string = "Hello"
#!/bin/bash
name = "Sagar"
echo "Hello $name"

O/p: Hello Sagar

* File operations
use -e to check if file exits.
syntax:-
if [-e filename]; then
commands
fi

Ex #!/bin/bash
echo "Enter file name:"
read file
if [-e "$file"]; then
echo "file exits"
else
echo "file does not exits"
fi

Checking file type
-f → regular file
-d - directory
-r → readable
-w → writable
-x → executable

Automation of System Tasks

Writing scripts to automate common tasks.

Automation means performing repetitive tasks automatically without manual intervention.

Shell scripts are commonly used for automation tasks such as:

- Backups
- Cleaning temporary files
- Monitoring disk space
- Updating the system
- User account management

* Basic steps for Automation
  1. Identify repetitive tasks.
  2. Write a shell script to perform the task.
  3. Make the script executable.
     chmod +x script.sh

Eg: Backup Automation

```
#!/bin/bash
# Backup Documents folder
SOURCE="/home/user/Documents"
DEST="/home/user/Backup"
DATE=$(date +%Y-%m-%d)
mkdir -p $DEST
cp -r $SOURCE $DEST/Backup_$DATE
echo "Backup completed on $DATE"
```

* mkdir -p → creates folder if it doesn't exist
  cp -r → copies recursively.

Cleaning Temporary files
Delete old log files.

```
#!/bin/bash
LOG_DIR="/var/log/myapp"
find $LOG_DIR -type f -name "*.log" -mtime
+7 -exec rm {} \;
echo "Old log files deleted".
```

- Find → Searches files.
- -mtime +7 → files older than 7 days.
- rm → Deletes the files.

* Disk space monitoring.
Task: Alert if disk space exceeds 80%.

```
#!/bin/bash
USAGE=$(df / | tail -1 | awk '{print $5}'
| sed 's/%//')
if [ $usage -gt 80 ]; then
    echo "Warning: Disk usage is above 80%!"
fi
```

Explanation
- df / — Shows disk usage
- awk '{print $5}' → Extracts usage percentage.
- sed 's/%//' → Removes the % symbol.

1. User account creation Automation

```
#!/bin/bash
echo "Enter username: "
read user
sudo useradd $user
sudo passwd $user
echo "user $user created successfully".
```

Automating system updates

```
#!/bin/bash
sudo apt update
sudo apt upgrade -y
echo "System updated successfully"
```

- apt update - updates repository
- apt upgrade -y - Installs all updates automatically.

- Scheduling Automation using Cron
- Cron is used to run scripts automatically at scheduled intervals.

crontab -e

Ex- Run backup script daily at 2AM.

```
02 * * * /home/user/backup.sh
```

Task type	script / command eg.
Backup folder	cp -r /source/ backup
Clean old files	find /logs -mtime +7 -exec rm {} \;
Disk monitoring	'df'
User creation	useradd username; passwd username
System update	apt update && apt upgrade -y
Schedule task	cron (0 2 * * * /path/ script.sh)

Sangam

Sharing/selling not allowed