D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

School of Computer Science, Engineering and Applications(SCSEA)

Academic Year: 2024-25 ***B.Tech S. Y. (Sem IV)***

Subject : System Software (CSE2103)

Name of the Student: Shravan polshettiwar **PRN: 20230802162**

Title of Practical : **Advanced Shell Scripting Concepts**

AIM – Commands line shell scripts programming**.**

- **Topics:**
  - Command-line arguments ($1, $2, shift)
  - Using functions in shell scripts
  - Error handling and debugging scripts

Theory:

PERFORM OPERATION: –

1. The following command line arguments are used to make shell scripts program

- Awk
  Purpose: awk is a text-processing tool used to manipulate and extract specific fields from text files.

```
ubuntu@ubuntu:~$ cat file.txt
ID Name Salary Country
1 Arjun 200000 India
2 Ram   30000  Germany
3 Sham  40000  India
4 Raju  50000  Russia
ubuntu@ubuntu:~$ awk '{print $2}' file.txt
Name
Arjun
Ram
Sham
Raju
```

The awk '{print $2}' file.txt command extracts and prints the second column from file.txt.

Explanation:

- awk '{print $2}' file.txt → Prints the second field (column) from each line.

- The file has columns: **ID, Name, Salary, Country**.
- $2 corresponds to the **Name** column, so it prints the names only.

- Sed

  Purpose: sed (Stream Editor) is used to perform basic text transformations and editing like search and replace, delete, insert, etc.

```
ubuntu@ubuntu:~$ cat file.txt
ID Name Salary Country
1 Arjun 200000 India
2 Ram   30000  Germany
3 Sham  40000  India
4 Raju  50000  Russia
ubuntu@ubuntu:~$ sed "s/Arjun/Pol/g" file.txt
ID Name Salary Country
1 Pol 200000 India
2 Ram   30000  Germany
3 Sham  40000  India
4 Raju  50000  Russia
```

  sed Command:

  - The sed "s/Arjun/Pol/g" file.txt command replaces all occurrences of "Arjun" with "Pol" in file.txt (output shown but doesn't modify the file unless -i is used).

- Find

  Purpose: find is used to search for files and directories in a directory hierarchy based on conditions likename, size, date, etc

```
ubuntu@ubuntu:~$ find /home -name "file.txt"
/home/ubuntu/file.txt
find: ‘/home/installer’: Permission denied
```

  find Command:

  - The find /home -name "file.txt" command searches for a file named "file.txt" within the /home directory.

- Cut

  Purpose: cut is used to extract sections or columns from each line of input text files.

```
ubuntu@ubuntu:~$ cat file2.txt
Arjun
Ram
Sham
Raju
Parth

ubuntu@ubuntu:~$ cut -c1 file2.txt
A
R
S
R
P
```

cut Command:

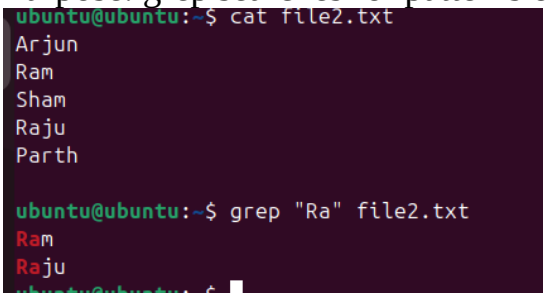
- cut -c1 file2.txt extracts and prints only the first character of each line from file2.txt

- Xargs

Purpose: xargs reads data from standard input and converts it into arguments for other commands.

- Grep

Purpose: grep searches for patterns or strings in files or input.

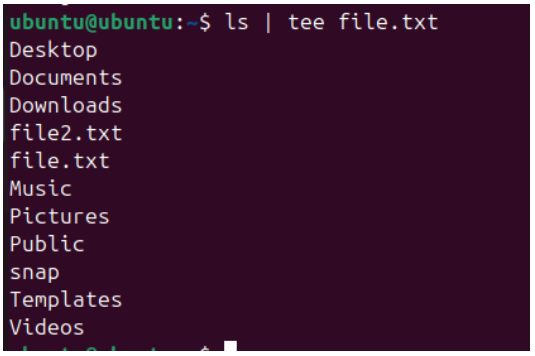


Grep Command:

- grep "Ra" file2.txt searches for lines containing "Ra" in file2.txt and displays matching lines: "Ram" and "Raju".

- Tee

Purpose: tee reads input and writes it to both the terminal and a file.



Tee Command:

- ls | tee file.txt lists directory contents and simultaneously writes the output to file.txt.

- Sort

sort is used to sort lines of text files in alphabetical or numerical order.

```
ubuntu@ubuntu:~$ sort file.txt
1 Arjun 200000 India
2 Ram   30000  Germany
3 Sham  40000  India
4 Raju  50000  Russia
ID Name Salary Country
```

Sort Command:

- sort file.txt sorts the contents of file.txt alphabetically, displaying the sorted data.

- Wc

```
ubuntu@ubuntu:~$ cat file.txt
ID Name Salary Country
1 Arjun 200000 India
2 Ram   30000  Germany
3 Sham  40000  India
4 Raju  50000  Russia
ubuntu@ubuntu:~$ wc -l file.txt
5 file.txt
```

wc (Word Count) Command:

- wc -l file.txt counts and displays the number of lines in file.txt, which is 5.

2. Write a function in a shell script that calculates the factorial of a given number.

```
#!/bin/bash


factorial(){
        num=$1
        fact=1

        for ((i=1; i<=num; i++)); do
                fact=$((fact*i))
        done

        echo "Factorial of $num  is $fact"
}

read -p "Enter your number:" n
factorial $n
```

```
ubuntu@ubuntu:~$ nano factorial.sh
ubuntu@ubuntu:~$ chmod +x factorial.sh
ubuntu@ubuntu:~$ ./factorial.sh
Enter your number:5
Factorial of 5  is 120
ubuntu@ubuntu:~$
```

Explanation:

1. **Script (factorial.sh)**:
   - Defines a function factorial() that:
     - Takes an input number ($1).
     - Initializes fact=1.
     - Uses a for loop to multiply fact with each number up to num.
     - Prints the factorial result.
   - Prompts the user to enter a number.

- Calls the factorial function with the entered number.

2. **Execution Steps**:
   - nano factorial.sh: Opens the script in the nano editor.
   - chmod +x factorial.sh: Makes the script executable.
   - ./factorial.sh: Runs the script.
   - User inputs **5**, and the output **120** is displayed (since **5! = 5×4×3×2×1**).

3. Write a function in a shell script that calculates student grade

```
#!/bin/bash
calculate_grade(){
        marks=$1
        if [ $marks -ge 90 ]; then
                echo "Grade A+"
        elif [ $marks -ge 80 ]; then
                echo "Grade A"
        elif [ $marks -ge 70 ]; then
                echo "Grade B"
        elif [ $marks -ge 60 ]; then
                echo "Grade C"
        elif [ $marks -ge 50 ]; then
                echo "Grade D"
        else
                echo "Grade F Fail"
        fi
}

read -p "Enter your marks:" marks
calculate_grade $marks
```

```
ubuntu@ubuntu:~$ nano grade.sh
ubuntu@ubuntu:~$ chmod +x grade.sh
ubuntu@ubuntu:~$ ./grade.sh
Enter your marks:85
Grade A
ubuntu@ubuntu:~$ ./grade.sh
Enter your marks:95
Grade A+
```

1. **Function Definition**:

- The script defines a function calculate_grade() that takes one argument ($1), which represents the marks.

2. **Conditional Checks (if-elif-else)**:
   - If marks are **90 or above**, it prints **"Grade A+"**.
   - If marks are **80 or above**, it prints **"Grade A"**.
   - If marks are **70 or above**, it prints **"Grade B"**.
   - If marks are **60 or above**, it prints **"Grade C"**.
   - If marks are **50 or above**, it prints **"Grade D"**.
   - Otherwise, it prints **"Grade F (Fail)"**.
3. **User Input Handling**:
   - The script uses read -p "Enter your marks:" marks to take input from the user.
4. **Function Call**:

- The script calls calculate grade $marks to determine and display the grade.

5. **Execution Steps** (shown in the terminal commands):
   - **Create the script**: nano grade.sh.
   - **Make it executable**: chmod +x grade.sh.
   - **Run the script**: ./grade.sh.
6. **Example Output**:

- User enters **85**, and the output is "**Grade A**".

4. Write a shell script that calculates the total size of all files in a directory

```
#!/bin/bash

read -p "Enter your path:" dir

if [ -d "$dir" ]; then
        total_size=$(du -sh "$dir" | awk '{print $1}')
        echo "Total size of all files $dir is : $total_size"
else
        echo "Directory does not exist!"
fi
```

```
ubuntu@ubuntu:~$ nano size.sh
ubuntu@ubuntu:~$ chmod +x size.sh
ubuntu@ubuntu:~$ ./size.sh
Enter your path:/home/ubuntu/Documents
Total size of all files /home/ubuntu/Documents is : 0
```

1. **Shebang**: #!/bin/bash (Executes in Bash).
2. **User Input**: Reads directory path.
3. **Directory Check**: Verifies if the path exists.
4. **Size Calculation**: Uses du -sh "$dir" | awk '{print $1}' to get total size.
5. **Output**: Displays total file size or error message.
6. **Execution**:
   - nano size.sh (Create script).
   - chmod +x size.sh (Make executable).
   - ./size.sh (Run script).

5. Error Handling & Debugging

```
#!/bin/bash

file="file.txt"

if [ -f "$file" ]; then
        echo "File exists"
else
        echo "Error: $file does not exist"
fi
```

```
ubuntu@ubuntu:~$ ./error.sh
Error: file.txt does not exist
```

```
#!/bin/bash

set -x

echo "Hello world"
ls "$home/nonexistent_dir"
```

```
ubuntu@ubuntu:~$ ./debug.sh
+ echo 'Hello world'
Hello world
```

1. **File Existence Check**:
   - Uses [-f "$file"] to check if file.txt exists.
   - Prints "File exists" or "Error: file.txt does not exist".
2. **Execution Output**:
   - If file.txt is missing, it prints an error message.
3. **Debugging with set -x**:
   - Enables debugging mode to show executed commands.
   - Helps identify issues in the script.
4. **Execution Output for Debugging**:
   - Displays each command before execution.
   - Example: + echo 'Hello world' shows the command before running.

CONCLUSION - In this practical, we delved into advanced shell scripting concepts by developing various command-line utilities and functions. The key topics covered included:

- **Factorial Calculation** – Showcased the use of recursive function calls in shell scripting.
- **Student Grade Calculation** – Applied conditional statements to determine grades based on input marks.
- **Total File Size Calculation** – Leveraged the du command to compute the total size of files within a directory.
- **Error Handling & Debugging** – Employed debugging techniques such as set -x and set -e, along with error trapping using trap, to ensure robust script execution.

Additionally, we utilized powerful commands like awk, sed, find, grep, cut, xargs, and wc to enhance the efficiency and functionality of our scripts. This hands-on experience reinforced our knowledge of shell scripting, automation, and system operations, which are crucial skills for system administrators and software developers.

Course TA (Mr. Rajkumar)