Operating System Lab

August 5, 2025

SHELL

Select lines with exactly two characters, and Select lines with minimum two characters:

```
grep "^..$" filename.txt
grep "^.." filename.txt
```

Explanation:

SHELL

Symbol	Explanation
..	Each dot represents any single character.
ˆ	Match to the start of the line.
.	'.' means any single character (except newline).
$	Match to the end of the line.

Table: Explanation of Regular Expression Symbols

SHELL

Select lines starting with uppercase letter, Select lines ending with uppercase letter

```
grep "^[A-Z]" filename.txt
grep "[A-Z]$" filename.txt
```

Explanation: Matches lines that begin and end with any uppercase letter (A-Z).

SHELL

Select lines ending with a period:

```
grep "\.$" filename.txt
```

Explanation:

```
The backslash \. escapes the dot so it matches a literal
period at the end of the line.
This finds lines that contain a dot (.) character.
Without \, . means "any character".
```

SHELL

Select lines with one or more blank spaces:

```
grep " " filename.txt
```

Explanation: Matches lines that contain at least one space character.

SHELL

Select lines with digits and write to new file:

```
grep "[0-9]" filename.txt > digits.txt
```

Explanation: Searches for any digit in a line. Redirects the matched lines to ‘digits.txt‘.

SHELL

```
grep ":ICT:" studentinformation.txt | wc -l
```

Purpose: Count the number of students from the `ICT` department.

- `grep ":ICT:" studentinformation.txt`
  - Searches for lines that contain `:ICT:` in the file.
  - Colons ensure accurate matching of department field (avoids partial matches).
- | (Pipe)
  - Passes output of `grep` to the next command.
- `wc -l`
  - Counts the number of matching lines (students from ICT).

SHELL

Sample Data:

```
20251234:Anu:ICT:IT:A:78:90:86
20251235:Nithya:CSE:IT:A:65:72:81
20251236:Rahul:ICT:ECE:B:88:77:91
```

Command Output:

- Matching Lines:

  ```
  20251234:Anu:ICT:IT:A:78:90:86
  20251236:Rahul:ICT:ECE:B:88:77:91
  ```

- Count = **2**

SHELL

Command: Replace "IT" with "Information Technology"

```
sed 's/:IT:/:Information Technology:/g'
studentinformation.txt > ITStudents.txt
```

Explanation:

- `s/old/new/g` — substitutes all occurrences of a pattern.
- `:IT:` — exact match ensures only the "Branch" field is replaced.
- `>` — redirects the output to a new file named `ITStudents.txt`.

SHELL

Command:Display Average Marks of Student 1234

```
grep "^20251234:" studentinformation.txt |
awk -F ":" '{avg=($6+$7+$8)/3; print
"Average marks of", $2, "is", avg}'
```

Explanation:

```
        grep "^20251234:" | matches lines starting with
        registration number 20251234.
```

- `awk -F ":"` — splits line by colon.
- Calculates average using fields 6, 7, and 8 (marks).

SHELL

Convert Title Row to Uppercase

```
head -1 studentinformation.txt | tr 'a-z' 'A-Z'
```

Explanation:

- `head -1` — gets the first line (title row).
- `tr 'a-z' 'A-Z'` — converts lowercase letters to uppercase.
- Command output: "20251234:ANU:ICT:IT:A:78:90:86"

SHELL

Command:

```
grep "MIT" *
grep "MIT" * | sed 's/MIT/Manipal Institute of Technology/g'
```

Explanation:

- `grep "MIT" *` – Lists all lines in all files that contain the word "MIT".
- `sed 's/MIT/Manipal Institute of Technology/g'` – Replaces all occurrences of "MIT" with its full form in those lines.

SHELL

Command:

```
wc *[0-9]*
```

Explanation:

- `*[0-9]*` – Matches filenames that contain at least one digit.
- `wc` – Prints number of lines, words, and characters in each file.

SHELL

Commands:

```
wc studentinformation.txt &
wc studentinformation.txt &
ps aux | grep wc
pkill wc
```

Explanation:

- `&` – Runs each `wc` command in background.
- `ps aux | grep wc` – Lists all `wc` processes.
- `pkill wc` – Terminates all `wc` processes.

What is Shell Scripting?

- A shell script is a text file containing a sequence of commands.
- Bash (Bourne Again SHell) is the most common shell in Linux.
- Used for automation, text processing, file handling.
- Script files usually end with `.sh` and run with: bash script.sh

Variables in Shell

- Assign without spaces: name=”Athira”
- Access with: echo *name*
- Always quote variables with spaces: `"$name"`.

Taking Input and Printing Output

- Read input from user: read -p "Enter your name: " name
- Print output: echo "Hello *name*"
- `-p` prints a prompt before reading.

For Loop in Shell

Syntax:

```
for variable in list
do
commands
done
```

Explanation:

- Iterates through each item in the `list`.
- Stores current item in `$variable`.
- Executes commands between `do` and `done`.

If–Else–If in Shell

Syntax:

```
if [ condition1 ]
then
    commands1
elif [ condition2 ]
then
    commands2
else
    commands3
fi
```

Explanation:

- `if` → First condition check.
- `elif` → Additional condition(s) if the first is false.
- `else` → Runs if none of the conditions are true.
- `fi` → Ends the conditional block.

Important Commands for Lab 3

- `grep` → search text in files.
- `find` → search files by name/type.
- `sed` → stream editor (replace text).
- `awk` → process columns in text.
- `bc` → calculator for floating-point math.
- `cp` → copy files.
- `mv` → rename/move files.

Check File Type

Command:

```
read -p "Enter file name: " file
if [ -d "$file" ]; then
    echo "$file is a directory"
elif [ -f "$file" ]; then
    echo "$file is a regular file"
else
    echo "$file does not exist"
fi
```

Explanation:

- `read -p` → Prompts the user and stores input in a variable.
- `$file` → The variable containing the entered file name.
- `-d` → Returns true if the path is a directory.
- `-f` → Returns true if the path is a regular file.
- `elif` → Else-if condition in shell script.
- `echo` → Prints the output to the screen.

List Files Containing Pattern

Command:

```
read -p "Enter folder path: " folder
read -p "Enter pattern: " pattern
grep -l "$pattern" "$folder"/*
```

Explanation:

- `grep` → Searches text in files.
- `-l` → Prints only the filenames with matches.
- `"$pattern"` → Pattern entered by the user.
- `"$folder"/*` → Searches all files in the given folder.

Replace File Extension Recursively

Command:

```
find . -type f -name "*.txt" -exec bash -c
'mv "$0" "${0%.txt}.text"' {} \;
```

Explanation:

- `find .` → Searches from current directory.
- `-type f` → Only files.
- `-name "*.txt"` → Matches files ending with ‘.txt‘.
- `-exec` → Executes a command on each file found.
- `mv` → Moves/renames the file.
- `$0%.txt.text` → Changes extension from ‘.txt‘ to ‘.text‘.

Calculate Gross Salary

Command:

```
read -p "Enter Basic Salary: " basic
read -p "Enter TA: " ta
gross=$(echo "$basic + $ta + (0.1 * $basic)" | bc)
echo "Gross Salary = $gross"
```

Explanation:

- `$()` → Command substitution (stores output in a variable).
- `bc` → Linux calculator for floating-point math.
- Gross Salary Formula: `Basic + TA + 10% of Basic`.

Copy Files with Given Extension

Command:

```
read -p "Enter extension: " ext
read -p "Enter destination folder: " dest
mkdir -p "$dest"
find . -maxdepth 1 -type f -name "*.$ext" -exec cp {}
"$dest" \;
```

Explanation:

- `mkdir -p` → Creates folder if it doesn't exist.
- `-maxdepth 1` → Checks only current folder, not subfolders.
- `cp` → Copies file to destination.

Replace 'ex:' with 'Example:'

Command:

```
for file in *; do
    [ -f "$file" ] && sed -i \
    's/^\(ex:\)/Example:/; s/\(\.\s*\)ex:/\1Example:/g'
    "$file"
done
```

Explanation:

- `for file in *` → Loops through all files.
- `-f` → True if path is a file.
- `sed -i` → Edit file in place.
- `êx:` → Match start of line.
- `˙*ex:` → Match after a period and spaces.

Delete Even-numbered Lines

Command:

```
sed -i 'n;d' filename.txt
```

Explanation:

- `n` $\rightarrow$ Read next line.
- `d` $\rightarrow$ Delete that line.
- This removes all even-numbered lines from the file.