JAIN DEEMED-TO-BE UNIVERSITY | SCHOOL OF COMPUTER SCIENCE AND IT

DEPARTMENT OF CS and IT

Programme: MCA

LAB RECORD

Advanced Operating System Lab

(23MCAC103L)

NAME : Dhanraj Chhalani

USN: 24MCAR0195

SEMESTER : I

BRANCH & SEC : MCA - Sec - B

SUBMITTED TO: Dr. S.K.Manju Bargavi, Professor
School of Computer Science and IT

ACADEMIC YEAR : 2024-2025

Certificate

I hereby declare that the work which is prepared for the award of the Degree of Master of Computer Application, submitted in the Department of Master of Computer Application, Jain (Deemed- to-be University), Jayanagar 560069, Bengaluru, Karnataka, India is an authentic record of my activity work carried out under the supervision of Dr. Manju Bargavi S.K, Assistant Professor, School of CS and IT, Jain (Deemed-to-be University), Jayanagar 560069, Bengaluru, Karnataka, India.

Signature of the student
Dhanraj Chhalani

Signature of the faculty incharge
Dr. Manju Bargavi S.K

Dhanraj Chhalani

INDEX PAGE

Experiment 1

AIM: Steps to install VMware Workstation pro 16 on Windows

Download VMware Workstation: Go to the VMware website and download the latest version of VMware Workstation Pro.

Installation Steps

Run the Installer:

Locate the downloaded installer file (usually with a.exe extension) and double-click it to start the installation process.

Start the Installation:

You might be prompted by User Account Control (UAC) to allow the installer

to make changes to your device. Click yes to continue.

Dialog box will appear Welcome to VMware Workstation pro setup wizard. Click next to continue the setup.

Accept the License Agreement:

Read and accept the End User License Agreement (EULA) by selecting the appropriate checkbox, then click Next.

Select Installation Type:

Choose between a typical or custom installation. The typical installation is recommended for most users. Click Next to continue.

Choose Installation Directory:

If you selected custom installation, you can choose the destination folder where

VMware Workstation will be installed. By default, it is installed in C:\Program Files

(x86)\VMware\VMware Workstation. Click Next.

Add VMware Workstation console tools into system path.

Configure Shortcuts:

Decide if you want to create shortcuts for VMware Workstation on your desktop or in the Start menu. Click Next.

Start Installation:

Click Install to begin the installation process. This may take a few minutes.

Finish Installation:

Once the installation is complete, you may be prompted to restart your computer.

Click Finish to exit the installer.

To activate the license key press License key Button and enter the key and press enter

Click **Finish** to **exit** the installer.

Troubleshooting

Ensure Virtualization is Enabled: Check that virtualization technology (VT-x/AMD-V) is enabled in your system's BIOS/UEFI settings.

Check for Updates: Regularly check for updates to VMware Workstation to ensure you have the latest features and security patches.

How to Install Kali Linux on VMware

To download the Kali VMware image, follow these steps:

Go to the official Kali Linux website (Download Kali).

https://www.kali.org/get-kali/#kali-virtual-machines

Scroll down to the Virtual Machines section.

Choose the VMware platform and click on the download link.

How to Install Kali Linux on VMware

Once the download is complete, you will notice it saved as a .zip file. Double click on the downloaded file to open it with your default zip program. If you need a zip program, we recommend 7Zip.

Set where you would like the file to be extracted and click OK.

Locate the extracted Kali VMware virtual configuration file and click Open

Click on “Power on this virtual machine” and Kali will l

EXPERIMENT -2

AIM : Working on basic Linux Shell Commands: ls, mkdir, rmdir, cd, cat, banner, touch, file, wc,sort, cut, grep, dd, dfspace, du, ulimit.

ls: Lists files and directories in the current directory.

>> ls

```

**Output:**

file1.txt file2.txt directory1

**mkdir**: Creates a new directory.

&gt;&gt; mkdir new_directory

**Output:** (No output if successful)

**rmdir**: Removes an empty directory.

&gt;&gt; rmdir old_directory

**Output:** (No output if successful)

**cd**: Changes the current directory.
```

>>cd new_directory

Output: *(No output if successful)*

`cat`: Concatenates and displays file contents.

>> cat file1.txt

Output:

This is the content of file1.txt.

banner: Creates a large ASCII banner of the text (Note: not available on all distributions; on some, `figlet` might be used instead).

>> banner Hello

Output:

```
_ _ _ _
| | | | | |
| |_| | ___| | | ____
| __| |/ _ \ | |/ _ \
| |_| | __/ | | (_) |
\__._|\___|_|_|\___/
```

touch: Creates a new empty file or updates the timestamp of an existing file.

>> touch newfile.txt

```

**Output:** *(No output if successful)*

**`file`**: Determines the type of a file.
```

>> file file1.txt

```

**Output:**

file1.txt: ASCII text

**`wc`**: Counts lines, words, and characters in a file.

&amp;gt;&amp;gt; wc file1.txt

**Output:**

10 50 200 file1.txt

*(Where 10 is the number of lines, 50 is the number of words, and 200 is the number of characters)*

**`sort`**: Sorts the contents of a file.

&amp;gt;&amp;gt;sort file1.txt **Output:** apple banana cherry

```

`cut`: Removes sections from each line of files.

>> cut -d',' -f1 file.csv

Output:

Name1

Name2

Name3

```

**`grep`**: Searches for patterns within files.

&gt;&gt; grep &amp;#39;pattern&amp;#39; file1.txt

**Output:**
```

```

This line contains the pattern.

```

`dd`: Converts and copies files.

>> dd if=inputfile of=outputfile bs=1M

```

**Output:**

1024+0 records in

1024+0 records out

```

`df`: Reports file system disk space usage.

>> df -h

```

**Output:**

Filesystem Size Used Avail Use% Mounted on

/dev/sda1 50G 20G 28G 43% /

```

`du`: Estimates file space usage.

>> du -h file1.txt

Output:

4.0K file1.txt

```

**`ulimit`**: Displays or sets user process resource limits.

&gt;&gt; ulimit -a

**Output:**
```

core file size (blocks, -c) 0 data seg size (kbytes, -d) 8192 max nice (-e) 40
open files (-n) 1024 stack size (kbytes, -s) 8192

EXPERIMENT- 3 (A)

AIM : To check whether number is prime or not.

```
number=43
i=2 f=0
while test $i -le `expr $number / 2`
do
if test `expr $number % $i` -eq 0
then f=1
fi
i=`expr $i + 1`
done
if test $f -eq 1
then
echo Not Prime
else
echo Prime
fi
```

Output :

```
Input: N = 43
Output: Prime

Input: N = 35
Output: Not Prime
```

EXPERIMENT – 3 (B)

AIM: To implement fibonacci series.

```
echo Enter number: read N a=0 b=1

echo -n The Fibonacci series is :

for (( i=0; i&lt;N; i++ ))

do

echo -n $a fn=$((a + b)) a=$b b=$fn done

echo “”;
```

Output:

```
umang@Chell:~$ ./exp11Q2b.sh
Enter the limit of fibonaci series
12
0
1
1
2
3
5
8
umang@Chell:~$
```

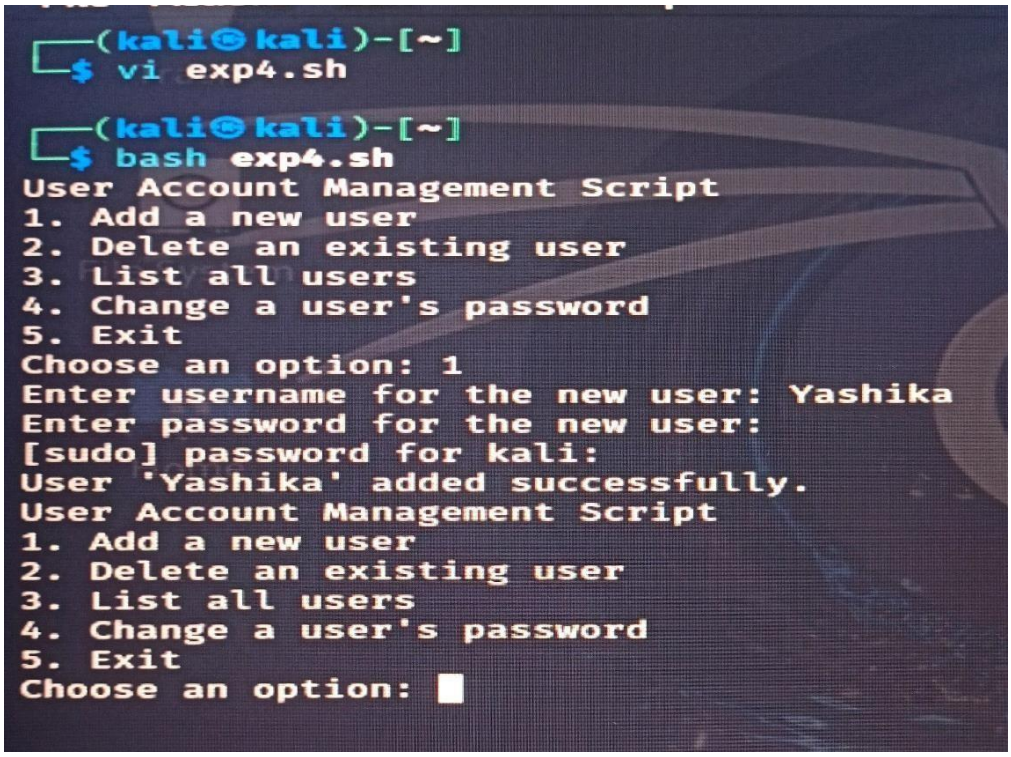
EXPERIMENT – 4

AIM: Write a shell script to manage the User accounts with its credentials.

```
#!/bin/bash

# Function to display menu options

show_menu() {

echo "User Account Management Script"

echo "1. Add a new user" echo "2. Delete an existing user"

echo "3. List all users"

echo "4. Change a user's password"

echo "5. Exit"

}

# Function to add a new user

add_user() {

read -p "Enter username for the new user: " username read -s -p "Enter
password for the new user: " password

echo if id "$username" &>/dev/null; then echo "Error: User '$username'
already exists." else sudo useradd "$username"

echo "$username:$password" | sudo chpasswd

echo "User '$username' added successfully." fi

}

# Function to delete a user

delete_user() {

read -p "Enter the username to delete: " username if id "$username"
&>/dev/null; then sudo userdel "$username"
```

```
echo "User '$username' deleted successfully." else

echo "Error: User '$username' does not exist." fi

}

# Function to list all users list_users() { echo "Listing all users:" cut -d: -f1
/etc/passwd

}

# Function to change a user's password

change_password() {

read -p "Enter the username to change password: " username

if id "$username" &>/dev/null;

then

read -s -p "Enter the new password: " password

echo "$username:$password" | sudo chpasswd

echo "Password for user '$username' changed successfully."

else

echo "Error: User '$username' does not exist."

fi

}

# Main program loop while true; do show_menu

read -p "Choose an option: " option case $option in 1) add_user ;;

delete_user ;;

list_users ;;

change_password ;;

echo "Exiting script."; exit 0 ;; *) echo "Invalid option. Please try again." ;;
esac

done
```

Output:



EXPERIMENT – 5

AIM: Write a shell script to display the contents of a file between the given line numbers.

```
#!/bin/bash

# Check if the correct number of arguments is provided

if [ "$#" -ne 3 ]; then

echo "Error: Incorrect number of arguments" echo "Usage: $0 filename start_line end_line" exit 1 fi

filename="$1" start_line="$2" end_line="$3"

# Check if the file exists if [ ! -f "$filename" ]; then

echo "Error: File '$filename' does not exist" exit 1 fi
```

```
# Check if start and end lines are valid numbers if ! [[ "$start_line" =~ ^[0-9]+$ ]] || ! [[ "$end_line" =~ ^[0-9]+$ ]]; then echo "Error: Start and end line numbers must be positive integers" exit 1 fi

# Check if start line is less than or equal to end line if [ "$start_line" -gt "$end_line" ]; then

echo "Error: Start line cannot be greater than end line" exit 1 fi

line_count=$(wc -l < "$1") echo " The total number of lines in $2: $line_count"

# Display the lines between start_line and end_line (inclusive) sed -n "${start_line},${end_line}p" "$filename"
```

Output:

```
exp5.sh abc.txt 12
The total number of lines in 1: 6
hii good morning.
how was your day.
```

(kali@kali)-[~]

```
bash exp5.sh abc.txt 3 1
Error: Start line cannot be greater than end line
```

(kalikali)-[~]

```
bash exp5.sh abc.txt 15
$ The total number of lines in 1:6
hii good morning. how was your day
```

EXPERIMENT – 6

AIM: Write a shell script the deletes all lines containing a specified word among the files.

```
#!/bin/bash

# Check if at least two arguments are provided

if [ $# -lt 2 ]; then

echo "Usage: $0 <word> <file1> [file2 ...]" exit 1 fi

# The first argument is the word to search for word="$1"

shift

# Loop through all the files provided as arguments for file in "$@" do

# Check if the file exists if [ ! -f "$file" ]; then echo "File not found: $file"
continue

fi

# Use sed to delete lines containing the word and save to a temporary file sed
"/$word/d" "$file" > "$file.tmp"

# Replace the original file with the modified one mv "$file.tmp" "$file"

echo "Processed $file: Removed lines containing '$word'" done

echo “All files processed”
```

Output:

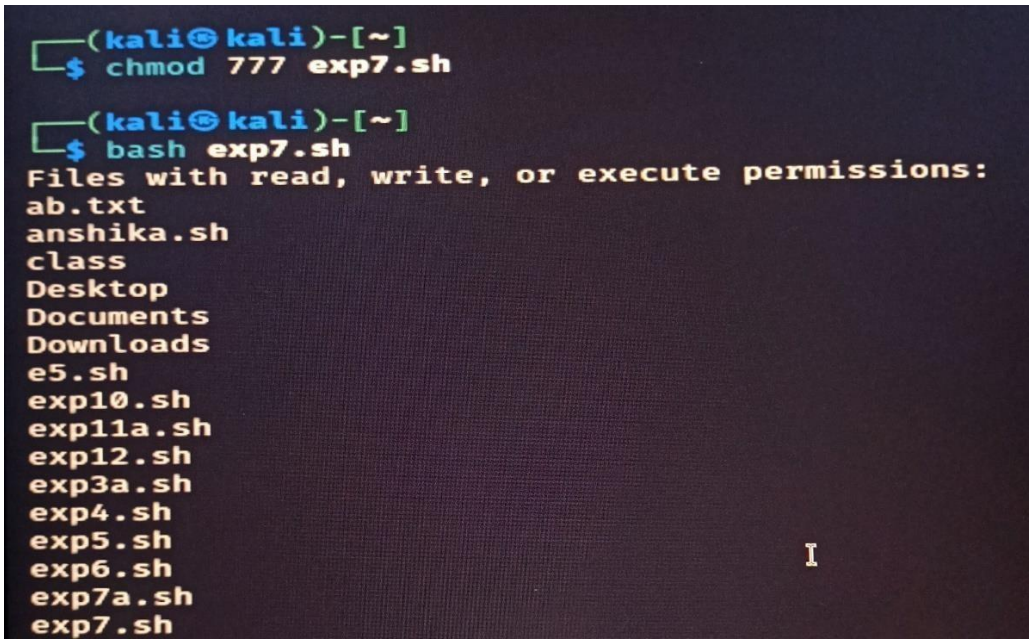
```
./exp6 are abc.txt

processed abc.txt: Removed lines containing ‘are’

All files processed
```

EXPERIMENT – 7

AIM: Shell script to print all files in the directory that have read , write and execute permissions.

```
#!/bin/bash
echo "Files with read, write, and execute permissions:"
# Loop through all files in the current directory for file in
do
# Check if it is a file and has read, write, and execute permissions
if [ -f "$file" ] && [ -r "$file" ] && [ -w "$file" ] && [ -x "$file" ];
then
echo "$file"
fi
```

Output :



EXPERIMENT – 8A

AIM: Bash script to check if given argument are files or directories and display the count if it’s a file.

```
#!/bin/bash
# Check if any arguments were provided
if [ $# -eq 0 ];
then
echo "Usage: $0 file1 [file2 ...]"
echo "Please provide at least one file or directory name as argument."
exit 1
fi
# Loop through all arguments for item in "$@"; do
# Check if the item exists
if [ -e "$item" ];
then
# Check if it's a directory
if [ -d "$item" ];
then
echo "'$item' is a directory"
# Check if it's a regular file
elif [ -f "$item" ];
then
 # Count the number of lines in the file
line_count=$(wc -l < "$item")
echo "'$item' is a file with $line_count lines"
else
echo "'$item' exists but is neither a regular file nor a directory"
fi
else
echo "'$item' does not exist"
fi
done
```

Output:

```
┌──(kali㉿kali)-[~]
└─$ bash exp8a.sh
Usage: exp8a.sh file1 [file2 ... ]
Please provide at least one file or directory name as argument.

┌──(kali㉿kali)-[~]
└─$ bash exp8a.sh library.txt
'library.txt' is a file with 7 lines
```

EXPERIMENT – 8B

AIM: To read words from a main file and count their occurrences in other specified files, outputting the total occurrence count of each word across these files

```
#!/bin/bash

# Check if at least two files are provided as arguments

if [ "$#" -lt 2 ]; then

echo "Usage: $0 file1 file2 [file3 ... fileN]" exit 1 fi

main_file="$1"

shift while read -r word; do total_count=0 for file in "$@"; do

if [ -f "$file" ];

then

count=$(grep -wo "$word" "$file" | wc -l) total_count=$((total_count + count))
else

echo "File $file does not exist." fi

done

echo "Word '$word' occurs

$total_count times in the other files."

done < "$main_file"
```

Output :

./bash exp8b.sh tx.txt ac.txt
Word ‘good morning’ occurs 5 times in the other files.
Word ‘abc’ occurs 2 times in the other files.

EXPERIMENT – 8C

AIM: To read content from a main file and count the occurances of lines , words and characters and outputting the total number of occurances of lines , words and characters.

```
#!/bin/bash

lines=0 words=0 chars=0

while IFS= read -r line || [[ -n "$line" ]]; do

 # Count lines

  ((lines++))

 # Count words     word_count=$(echo "$line" | tr -s ' ' '\n' | grep -c .)

  ((words += word_count))

# Count characters     char_count=${#line}

((chars += char_count)) done

echo "Lines: $lines" echo "Words: $words" echo "Characters: $chars"
```

Output :

./bash exp8c.sh < abc.txt
Lines: 6
Words: 22
Characters: 103

EXPERIMENT – 9

AIM: Write shell script for :

Showing the count of users logged in.

Printing Column list of files in your home directory.

Listing your job with normal priority.

Continue running your job after logging out.

```
#!/bin/bash
# Function to show count of logged in users
show_logged_users() {
echo "=== Currently Logged In Users ==="
who | wc -l
echo "Detailed user list:"
who
echo "----------------"
}
 # Function to show column listing of home directory
show_home_files() {
echo "=== Files in Home Directory ==="
ls -1 ~/
echo "----------------"
}
# Function to show current user's jobs with normal priority
show_jobs() {
echo "=== Current Jobs with Normal Priority ==="
```

```
ps -u $USER -o pid,ppid,nice,cmd | grep -v "TIME CMD" | awk '$3 == 0
{print}'
echo "----------------"
}
# Function to demonstrate nohup usage
run_background_job() {
echo "=== Starting a Background Job ==="
echo "Example: Running 'date' command every 5 seconds in background"
# Create a simple loop script
cat << 'EOF' > loop_script.sh
#!/bin/bash while true;
do
date >> output.log
sleep 5
done
EOF
chmod +x loop_script.sh
# Run the script with nohup
nohup ./loop_script.sh &     echo "Job started with PID: $!"
echo "Output will be logged to nohup.out"
echo "----------------" }
# Main execution echo "System Information and Job Management Script"
echo
# Execute all functions
show_logged_users
```

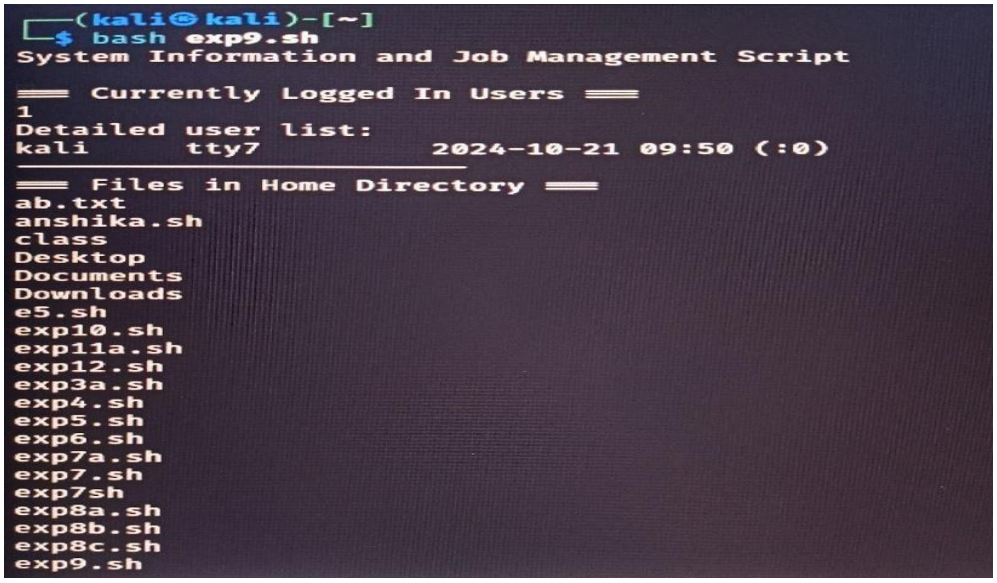
show_home_files

show_jobs

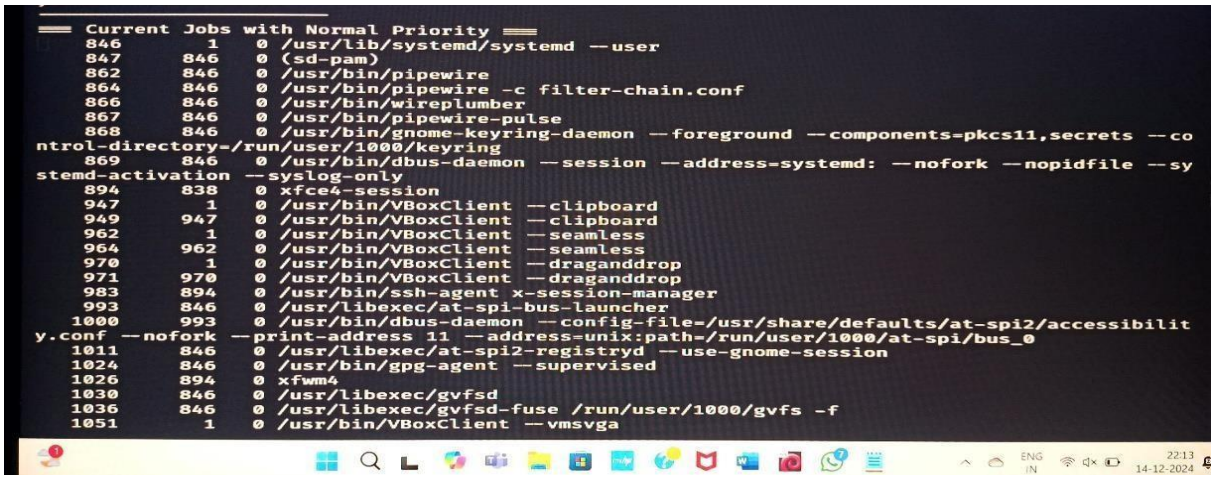
run_background_job

echo "Script completed!"

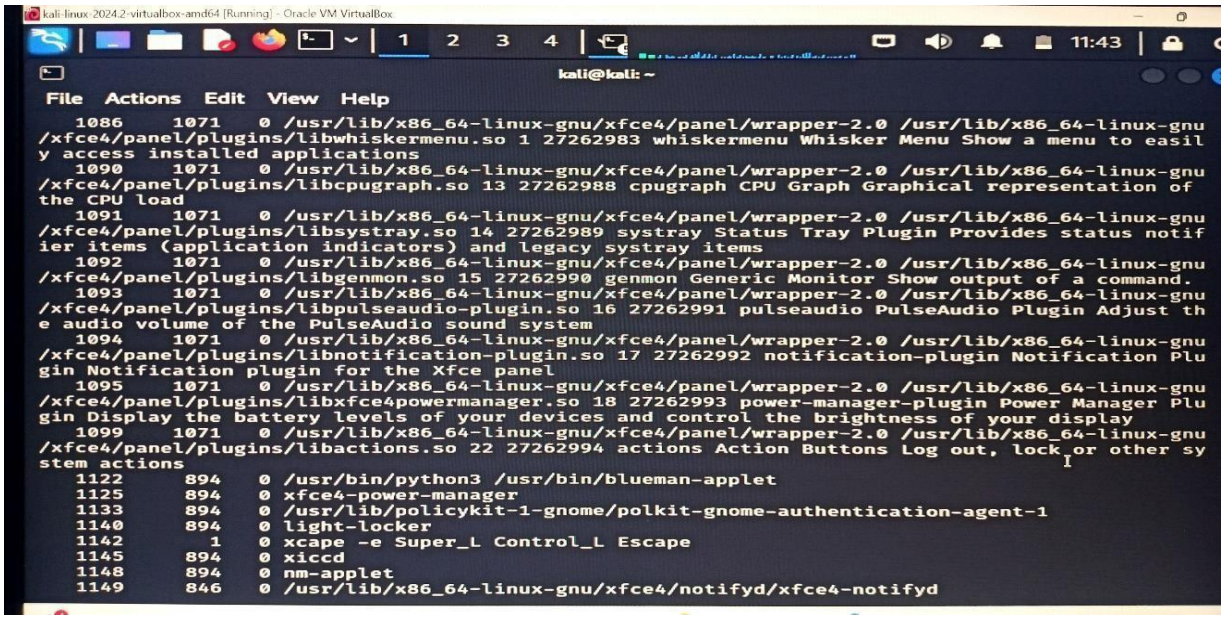
Output :

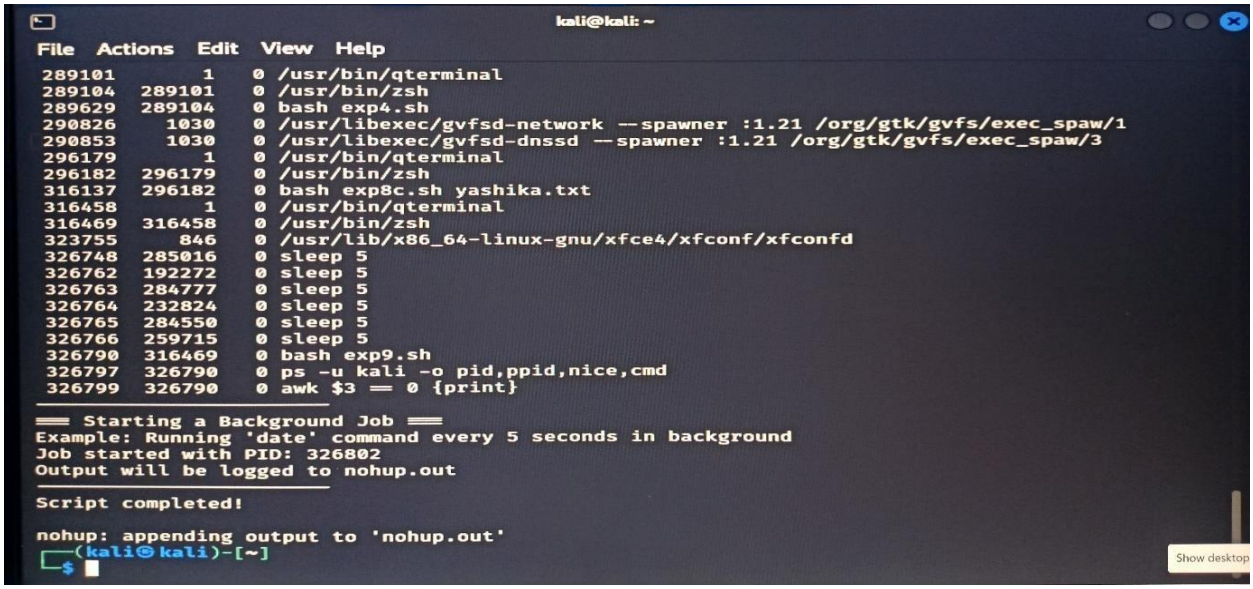


```
┌──(kali㉿kali)-[~]
└─$ bash exp9.sh
System Information and Job Management Script

=== Currently Logged In Users ===
1
Detailed user list:
kali     tty7         2024-10-21 09:50 (:0)

=== Files in Home Directory ===
ab.txt
anshika.sh
class
Desktop
Documents
Downloads
e5.sh
exp10.sh
exp11a.sh
exp12.sh
exp3a.sh
exp4.sh
exp5.sh
exp6.sh
exp7a.sh
exp7.sh
exp7sh
exp8a.sh
exp8b.sh
exp8c.sh
exp9.sh
```



```
=== Current Jobs with Normal Priority ===
    846       1    0 /usr/lib/systemd/systemd --user
    847     846    0 (sd-pam)
    862     846    0 /usr/bin/pipewire
    864     846    0 /usr/bin/pipewire -c filter-chain.conf
    866     846    0 /usr/bin/wireplumber
    867     846    0 /usr/bin/pipewire-pulse
    868     846    0 /usr/bin/gnome-keyring-daemon --foreground --components=pkcs11,secrets --co
ntrol-directory=/run/user/1000/keyring
    869     846    0 /usr/bin/dbus-daemon --session --address=systemd: --nofork --nopidfile --sy
stemd-activation --syslog-only
    894     838    0 xfce4-session
    947       1    0 /usr/bin/VBoxClient --clipboard
    949     947    0 /usr/bin/VBoxClient --clipboard
    962       1    0 /usr/bin/VBoxClient --seamless
    964     962    0 /usr/bin/VBoxClient --seamless
    970       1    0 /usr/bin/VBoxClient --draganddrop
    971     970    0 /usr/bin/VBoxClient --draganddrop
    983     894    0 /usr/bin/ssh-agent x-session-manager
    993     846    0 /usr/libexec/at-spi-bus-launcher
   1000     993    0 /usr/bin/dbus-daemon --config-file=/usr/share/defaults/at-spi2/accessibilit
y.conf --nofork --print-address 11 --address=unix:path=/run/user/1000/at-spi/bus_0
   1011     846    0 /usr/libexec/at-spi2-registryd --use-gnome-session
   1024     846    0 /usr/bin/gpg-agent --supervised
   1026     894    0 xfwm4
   1030     846    0 /usr/libexec/gvfsd
   1036     846    0 /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f
   1051       1    0 /usr/bin/VBoxClient --vmsvga
```





EXPERIMENT – 10

AIM: Write a shell script to create a file in the $USER /class/batch directory. Follow the Instructions

Input a page profile to yourself, copy it into other existing file

Start printing file at certain line

Print all the difference between two file

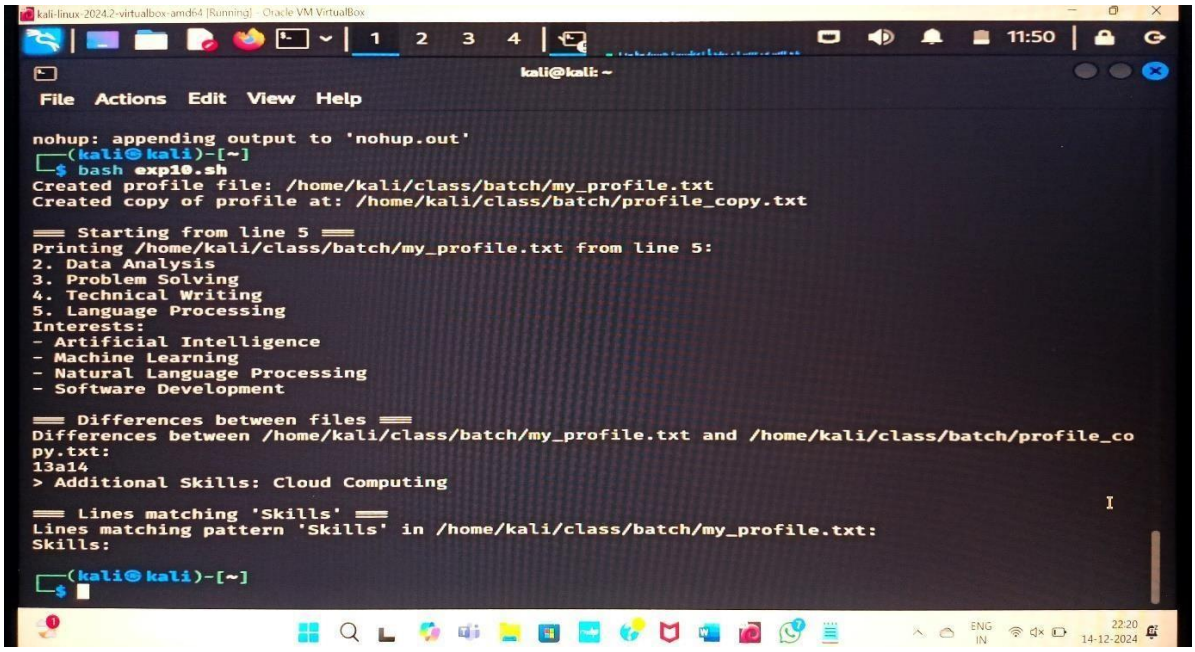
Print lines matching certain word pattern

```
#!/bin/bash
# Check if required directory exists, if not create it
USER_DIR="/home/$USER/class/batch"
if [ ! -d "$USER_DIR" ];
then
mkdir -p "$USER_DIR"
echo "Created directory: $USER_DIR" fi
# Create a profile file with some sample content
PROFILE_FILE="$USER_DIR/my_profile.txt"
cat > "$PROFILE_FILE" << EOF
Name: Claude Role: AI Assistant Skills:
Programming
Data Analysis
Problem Solving
Technical Writing 5. Language Processing Interests:
Artificial Intelligence
Machine Learning
Natural Language Processing
Software Development
EOF
echo "Created profile file: $PROFILE_FILE"
# Copy profile to another file
COPY_FILE="$USER_DIR/profile_copy.txt"
cp "$PROFILE_FILE" "$COPY_FILE"
```

```
echo "Created copy of profile at: $COPY_FILE"

# Modify the copy file slightly to demonstrate diff echo "Additional Skills:
Cloud Computing" >> "$COPY_FILE"

# Function to print file starting from specific line

print_from_line() {

local file=$1

local start_line=$2

echo "Printing $file from line $start_line:"

tail -n "+$start_line" "$file"

}

# Function to find differences between files show_differences() {

local file1=$1

local file2=$2

echo "Differences between $file1 and $file2:"

diff "$file1" "$file2"

}

# Function to search for pattern

search_pattern() {

local file=$1

local pattern=$2

echo "Lines matching pattern '$pattern' in $file:"

grep "$pattern" "$file"

}

 # Demonstrate the functions

echo -e "\n=== Starting from ==="
```

```
print_from_line "$PROFILE_FILE" 5

echo -e "\n=== Differences between files ==="
show_differences "$PROFILE_FILE" "$COPY_FILE"

echo -e "\n=== Lines matching 'Skills' ==="
search_pattern "$PROFILE_FILE" "Skills"
```

Output :

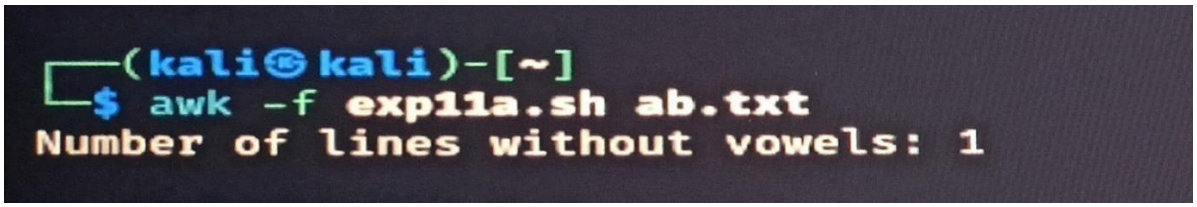


EXPERIMENT – 11A

AIM: AWK script to count the number of lines in a given input file that do not contain any vowels (case insensitive)

```
#!/usr/bin/awk -f

# Count lines that don't contain vowels (case insensitive)

{

# Convert line to lowercase for case-insensitive matching

line = tolower($0)

# If line doesn't match any vowels, increment counter

    if (line !~ /[aeiou]/) {

        count++

    }

}


# Print final count at end of file

END {

    print "Number of lines without vowels:", count

}
```

Output :

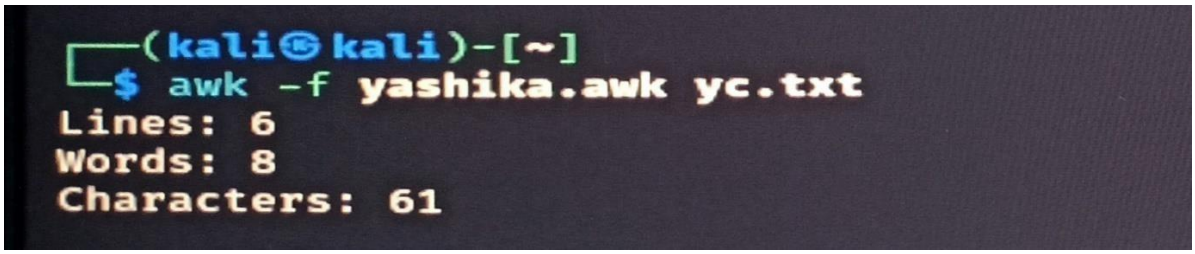


EXPERIMENT – 11B

AIM: AWK script to calculate and display the following statistics for a given text file

```
#!/usr/bin/awk -f

# Initialize counters at the start

BEGIN {

chars = 0     words = 0     lines = 0

}

line {

chars += length($0) + 1  # +1 for the newline character

# Count words by splitting the line into array     words += NF

# Count lines

lines++ }

 # Print results at the end

END {

printf "Lines: %d\n", lines

printf "Words: %d\n", words

printf "Characters: %d\n", chars

}
```

Output :



EXPERIMENT – 12

AIM: Function to validate email address and phone number.

```
#!/bin/bash

# Function to validate email address validate_email() {     local email=$1

   # RFC 5322 compliant email regex     local email_regex="^[a-zA-Z0-
9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"

   if [[ $email =~ $email_regex ]]; then         echo "Valid email address:
$email"

       return 0     else         echo "Invalid email address: $email"

       return 1

   fi

}

# Function to validate Indian phone number validate_phone() {     local
phone=$1

   # Indian phone number regex

   # Supports formats:

   # +91 1234567890

   # +91-1234567890

   # 91-1234567890     # 1234567890     local phone_regex="^(\+91[-
[:space:]]?)?[0-9]{10}$"

   if [[ $phone =~ $phone_regex ]]; then
```

```
# Additional validation for valid mobile prefixes
# Indian mobile numbers start with 6,7,8,9
local first_digit=${phone: -10:1}
if [[ $first_digit =~ [6-9] ]];
then
echo "Valid phone number: $phone"
return 0
fi
    echo "Invalid phone number: $phone"
    return 1
}

# Main script
echo "Email and Phone Number Validation Script"
echo "----------------------------"

# Test email validation while true;
do
read -p "Enter email address (or 'q' to move to phone validation): "
email
if [ "$email" = "q" ]; then
break
fi
validate_email "$email"
done
```

```
# Test phone validation
while true;
do
read -p "Enter phone number (or 'q' to quit): "
phone
if [ "$phone" = "q" ]; then
break
fi
validate_phone "$phone" done
exit 0
```

Output :