UNIX ASSIGNMENT – 7

NAME:C.PAVITHRA

ROLL NO:422127

SECTION: A

Shell Scripts :

1)b

```
#!/bin/bash

# Define list of machines machines=("172.50.11.106" "172.50.9.213" "127.0.0.1")  # Add your
machine names or IPs here

# SSH username username="your_username"

# Function to get memory usage get_memory_usage() {
   ssh "$username@$1" "free -m | awk 'NR==2{printf \"Memory Usage: %.2f%%\\n\", \$3*100/\$2}'"
}

# Function to get CPU usage get_cpu_usage() {
   ssh "$username@$1" "top -bn1 | grep 'Cpu(s)' | sed 's/.*, *\([0-9.]*\)%* id.*/\1/' | awk '{print \"CPU
Usage: \"100 - \$1\"%\"}'"
}

# Iterate over machines for machine in
"${machines[@]}"; do     echo
"Machine: $machine"
```

```
get_memory_usage "$machine"
get_cpu_usage "$machine"
   echo "----------------------------------------"
done
```

2)more

```
#!/bin/bash

# Check if a filename is provided as argument if
[ $# -eq 0 ]; then     echo "Usage: $0 states.txt"
   exit 1
fi

# Check if the file exists if [ ! -f "$1"
]; then     echo "File '$1' does not
exist."
   exit 1
fi

# Display the content of the file using the more command
more "$1" 3)nice
#!/bin/bash

# Check if a command is provided as argument
if [ $# -eq 0 ]; then
   echo "Usage: $0 <command>"
   exit 1
fi

# Run the provided command with a nice value of 10
```

```
nice -n 10 "$@"
```

4)nl

```
#!/bin/bash

# Check if a filename is provided as argument
if [ $# -eq 0 ]; then     echo
"Usage: $0 capital.txt"
   exit 1
fi

# Check if the file exists if [ ! -f "$1"
]; then     echo "File '$1' does not
exist."     exit 1
fi

# Display the content of the file with line numbers using the nl command nl
"$1" 5)pr
#!/bin/bash

# Check if a filename is provided as argument if
[ $# -eq 0 ]; then     echo "Usage: $0 states.txt>"
   exit 1 fi

# Check if the file exists if [ ! -f "$1"
]; then     echo "File '$1' does not
exist."
   exit 1
fi
```

```
# Display the content of the file with pagination using the pr command pr
"$1"
```

6)psswd

```
#!/bin/bash

# Prompt user for username read -p
"Enter username: " username

# Check if the username is provided if
[ -z "$username" ]; then     echo
"Username is required."
    exit 1
fi

# Check if the user exists if ! id "$username"
&>/dev/null; then     echo "User '$username'
does not exist."
    exit 1 fi

# Prompt user for a new password read -s -p "Enter new
password for $username: " password echo

# Prompt user to confirm the new password read -s -p
"Confirm new password: " password_confirm echo

# Check if passwords match if [ "$password" !=
"$password_confirm" ]; then     echo "Passwords do
not match. Please try again."
```

```
    exit 1 fi

# Change the user's password using the passwd command
echo "$password" | passwd --stdin "$username"

# Check if the password change was successful
if [ $? -eq 0 ]; then       echo "Password for user '$username' has been successfully updated." else       echo "Failed to update password for user '$username'."
    exit 1
fi
```

7)rcp

```
#!/bin/bash

# Check if source and destination files are provided
if [ $# -lt 2 ]; then     echo "Usage: $0 <source_file> <destination_host>:<destination_file>"
    exit 1 fi

# Extract source and destination information
source_file="$1" destination_host="${2%%:*}"
destination_file="${2##*:}"

# Check if the source file exists if [ ! -f "$source_file" ]; then     echo "Source file '$source_file' does not exist."
    exit 1
fi
```

```
# Check if the destination host is provided if
[ -z "$destination_host" ]; then     echo
"Destination host is required."
   exit 1 fi

# Check if the destination file is provided if
[ -z "$destination_file" ]; then     echo
"Destination file is required."
   exit 1
fi

# Copy the file using rcp rcp "$source_file"
"$destination_host":"$destination_file"

# Check if the copy operation was successful if [ $? -eq 0 ]; then     echo "File '$source_file'
copied successfully to '$destination_host:$destination_file'." else     echo "Failed to copy
file '$source_file' to '$destination_host:$destination_file'."
   exit 1 fi
```

8)rlogin

```
#!/bin/bash

# Check if a hostname is provided as argument
if [ $# -eq 0 ]; then     echo
"Usage: $0 student"
   exit 1
fi
# Check if the hostname is provided
if [ -z "$1" ]; then     echo
"Hostname is required."
```

```
    exit 1 fi

# Prompt user for username read -p "Enter
username for $1: " username

# Check if the username is provided if
[ -z "$username" ]; then     echo
"Username is required."
    exit 1
fi

# Attempt to login using rlogin rlogin
"$1" -l "$username"
9)rsh
#!/bin/bash

# Check if a hostname and command are provided
if [ $# -lt 2 ]; then
    echo "Usage: $0 <hostname> <command>"
    exit 1
fi

# Extract hostname and command hostname="$1"
shift
command="$@"
# Check if the hostname is provided if
[ -z "$hostname" ]; then     echo
"Hostname is required."
    exit 1 fi
```

```
# Check if the command is provided if
[ -z "$command" ]; then     echo
"Command is required."
   exit 1
fi

# Check if the 'rsh' command is available if !
command -v rsh &>/dev/null; then     echo "The
'rsh' command is not available."
   exit 1 fi

# Execute the command on the remote system using rsh rsh
"$hostname" "$command"
10)talk
#!/bin/bash

# Check if a username is provided
if [ $# -eq 0 ]; then     echo
"Usage: $0 student"
   exit 1
fi

# Check if the username is provided if [ -
z "$1" ]; then
   echo "Username is required."
   exit 1 fi
```

```
# Check if the 'talk' command is available if !
command -v talk &>/dev/null; then     echo "The
'talk' command is not available."
   exit 1
fi

# Start the talk session with the specified user talk
"$1"
```

11)telnet

```
#!/bin/bash

# Check if host and port are provided
if [ $# -lt 2 ]; then
   echo "Usage: $0 <host> <port>"
   exit 1
fi

# Extract host and port
host="$1" port="$2"

# Check if host and port are provided
if [ -z "$host" ] || [ -z "$port" ]; then     echo
"Both host and port are required."
   exit 1
fi
# Check if the file exists if [ ! -f "$1" ]; then echo "File '$1' does not exist." exit 1 f
# Run telnet command
telnet "$host" "$port"
```

12)tput

```
#!/bin/bash

# Clear the screen tput
clear

# Get the number of columns and rows of the terminal
cols=$(tput cols) rows=$(tput lines) echo "Terminal size:
$cols columns x $rows rows"

# Set terminal text color to red
tput setaf 1 echo "This is red
text"

# Set terminal text color to green
tput setaf 2 echo "This is green
text"

# Set terminal text color to default tput
sgr0

# Set terminal background color to yellow
tput setab 3 echo "This has a yellow
background"

# Reset terminal background color tput
sgr0
```

Move cursor to specific position tput cup 10

20 echo "Cursor moved to row 10, column

20"

13)tty

#!/bin/bash

Get the terminal filename terminal=$(tty)

Print the terminal filename echo

"Terminal filename: $terminal"

Get the terminal type terminal_type=$(tty -s && echo

"$TERM" || echo "not a tty") echo "Terminal type:

$terminal_type"

14)uname

#!/bin/bash

Get system name

system_name=$(uname -s) echo

"System name: $system_name"

Get node (machine) name

node_name=$(uname -n) echo

"Node name: $node_name" # Get

kernel release

kernel_release=$(uname -r) echo

"Kernel release: $kernel_release"

```
# Get kernel version
kernel_version=$(uname -v) echo "Kernel
version: $kernel_version"

# Get machine hardware name
machine=$(uname -m) echo "Machine
hardware: $machine"

# Get processor type
processor=$(uname -p) echo
"Processor type: $processor"

# Get operating system name
operating_system=$(uname -o) echo "Operating
system: $operating_system"
15)wc
#!/bin/bash

# Check if a filename is provided if
[ $# -eq 0 ]; then     echo "Usage:
$0 states.txt"
   exit 1
fi

# Check if the file exists if [ !
-f "$1" ]; then

   echo "File '$1' does not exist."
   exit 1 fi
```

```
# Get line count lines=$(wc -l <
"$1") echo "Number of lines:
$lines"

# Get word count words=$(wc -w <
"$1") echo "Number of words:
$words"

# Get byte count bytes=$(wc -c <
"$1") echo "Number of bytes:
$bytes"
```