Linux Practical Assignment Solution

Dr. J. V. Smart

Table of Contents

Linux Practical Problems Solutions

1. Display the current directory

   Solution

   ```
   pwd
   ```

2. Go to your home directory

   Solution

   ```
   cd
   ```

3. Create a directory called `dir1`

   Solution

   ```
   mkdir dir1
   ```

4. Go to the directory `dir1`

   Solution

```
cd dir1
```

5. Create a file `a1.txt` and type some content in it

   Solution

```
nano a1.txt
```

6. Display the contents of `a1.txt` on screen

   Solution

```
cat a1.txt
```

7. Create a directory called `dir2`

   Solution

```
mkdir dir2
```

8. Go to the directory `dir2`

   Solution

```
cd dir2
```

9. create a file `a2.txt` and type some content in it

   Solution

```
nano a2.txt
```

10. Go to your home directory

    Solution

```
cd
```

11. Display the contents of the file `a2.txt` using relative path

Solution

```
cat dir1/dir2/a2.txt
```

12. Display the contents of the file `a2.txt` using absolute path

Solution

```
cat /home/jignesh/dir1/dir2/a2.txt
```

13. Go to the directory `dir2`

Solution

```
cd dir1/dir2
```

14. Delete the file `a2.txt`

Solution

```
rm a2.txt
```

15. Go to the parent directory

Solution

```
cd ..
```

16. Copy the file `a1.txt` to `a3.txt`

Solution

```
cp a1.txt a3.txt
```

17. Rename the directory `dir2` to `dir3`

Solution

```
mv dir2 dir3
```

18. Rename the file `a3.txt` to `a4.txt`

    Solution

    ```
    mv a3.txt a4.txt
    ```

19. Move the file `a4.txt` to `dir3`

    Solution

    ```
    mv a4.txt dir3
    ```

20. Copy the file `a1.txt` to `dir3`

    Solution

    ```
    cp a1.txt dir3
    ```

21. Go to the directory `dir3`

    Solution

    ```
    cd dir3
    ```

22. Display the contents of the current directory

    Solution

    ```
    ls
    ```

23. Display the detailed list of the contents of the current directory

    Solution

    ```
    ls -l
    ```

24. Go to your home directory

    Solution

```
cd
```

25. Display the detailed list of the contents of the directory `dir3` using relative path

    Solution

    ```
    ls -l dir1/dir3
    ```

26. Display the detailed list of the contents of the directory `dir3` using absolute path

    Solution

    ```
    ls -l /home/jignesh/dir1/dir3
    ```

27. Go to the directory `dir3`

    Solution

    ```
    cd dir1/dir3
    ```

28. Edit the file `a1.txt` and change the contents

    Solution

    ```
    nano a1.txt
    ```

29. Delete the file `a1.txt`

    Solution

    ```
    rm a1.txt
    ```

30. Go to the parent directory of the parent directory

    Solution

    ```
    cd ../..
    ```

31. Go to the directory `dir1`

    Solution

    ```
    cd dir1
    ```

32. Delete the directory `dir3`

    Solution

    ```
    rm -r dir3
    ```

33. Go to the parent directory

    Solution

    ```
    cd ..
    ```

34. Create a directory `dir4`

    Solution

    ```
    mkdir dir4
    ```

35. Go into `dir4`

    Solution

    ```
    cd dir4
    ```

36. create a file `a5.txt` with some content

    Solution

    ```
    nano a5.txt
    ```

37. Display the current directory

    Solution

```
ls
```

38. Go to your home directory

    Solution

```
cd
```

39. Move the directory `dir4` in to `dir1`

    Solution

```
mv dir4 dir1
```

40. Delete the directory `dir1` with all the contents

    Solution

```
rm -r dir1
```

41. Create a variable `x`

    Solution

```
x=abc
```

42. Display the value of the variable `x`

    Solution

```
echo $x
```

43. Change the value of the variable `x` to `30`

    Solution

```
x=30
```

44. Assign the value `7` to variable `y`

Solution

```
y=7
```

45. Display the values of both `x` and `y`

Solution

```
echo "x=$x, y=$y"
```

46. Store the sum of `x` and `y` in to the variable `z`

Solution

```
let z=x+y
```

47. Store the subtraction of `x` and `y` in to the variable `z`

Solution

```
let z=x-y
```

48. Store the multiplication of `x` and `y` in to the variable `z`

Solution

```
let z=x*y
```

49. Store the division of `x` and `y` in to the variable `z`

Solution

```
let z=x/y
```

50. Store the remainder of the division of `x` and `y` in to the variable `z`

Solution

```
let z=x%y
```

51. Store the value of `z*(x+y)-5` in to the variable `a`

    Solution

    ```
    let a=z*(x+y)-5
    ```

52. Display the message `a={value of a}` (display the actual value of a)

    Solution

    ```
    echo "a=$a"
    ```

53. Display all files whose names begin with `b`

    Solution

    ```
    ls b*
    ```

54. Display all files whose names begin with `b` and end in `n`

    Solution

    ```
    ls b*n
    ```

55. Display all files whose names begin with `b` , end in `n` and have exactly 3 characters in-between

    Solution

    ```
    ls b???n
    ```

56. Display all files whose names begin with `asg`

    Solution

    ```
    ls asg*
    ```

57. Display all files whose names begin with `asg` and end with `.txt`

    Solution

```
ls asg*.txt
```

58. Display all files whose names begin with `asg` and have exactly two characters after that

    Solution

```
ls asg??
```

59. Display all files whose names have exactly two characters followed by `.txt`

    Solution

```
ls ??.txt
```

60. Assign the value `10` to `x` and `20` to `y`

    Solution

```
x=10
y=20
```

61. If `x` is greater than `y`, then display `Greater`

    Solution

```
if [ "$x" -gt "$y" ]
then
    echo Greater
fi
```

62. If `x` is less than `y`, then display `Less`

    Solution

```
if [ "$x" -lt "$y" ]
then
    echo Less
fi
```

63. If `x` is greater than or equal to `y`, then display `Greater than or equal`

Solution

```
if [ "$x" -ge "$y" ]
then
    echo Greater than or equal
fi
```

64. If `x` is less than or equal to `y`, then display `Less than or equal`

Solution

```
if [ "$x" -le "$y" ]
then
    echo Less than or equal
fi
```

65. If `x` is equal to `y`, then display `Equal`

Solution

```
if [ "$x" -eq "$y" ]
then
    echo Equal
fi
```

66. If `x` is not equal to `y`, then display `Not equal`

Solution

```
if [ "$x" -ne "$y" ]
then
    echo Not equal
fi
```

67. Assign `abc` to `x`

Solution

```
x=abc
```

68. Display `Zero length` if length of `x` is zero

Solution

```
if [ -z "$x" ]
then
    echo "Zero length"
fi
```

69. Display `Non-zero length` if length of `x` is non-zero

Solution

```
if [ -n "$x" ]
then
    echo "Non-zero length"
fi
```

70. Assign `abc` to `y`

Solution

```
y=abc
```

71. Display `Equal` if `x` is equal to `y`

Solution

```
if [ "$x" = "$y" ]
then
    echo Equal
fi
```

72. Display `Not equal` if `x` is not equal to `y`

Solution

```
if [ "$x" != "$y" ]
then
    echo Not equal
fi
```

73. Assign `querty` to `y`

Solution

```
y=querty
```

74. Display `Equal` if `x` is equal to `y`

Solution

```
if [ "$x" = "$y" ]
then
    echo Equal
fi
```

75. Display `Not equal` if `x` is not equal to `y`

Solution

```
if [ "$x" != "$y" ]
then
    echo Not equal
fi
```

76. Write a shell script to input the values of two numbers `n1` and `n2`. Display `Equal` if the numbers are equal, `Greater` if `n1>n2` and `Less` if `n1<n2`

Solution

```
read -p "Enter  the first number: " no1
read -p "Enter the second number: " no2
if [ "$no1" -gt "$no2" ]
then
    echo "Greater"
elif [ "$no1" -eq "$no2" ]
then
    echo "Equal"
else
    echo "Less"
fi
```

77. Write a shell script to input the values of two numbers `n1` and `n2`. Display all the integers between `n1` and `n2`, including `n1` and `n2`

Solution

```
read -p "Enter the value of the first number: " n1
read -p "Enter the value of the second number: " n2
```

```
no="$n1"
while [ "$no" -le "$n2" ]
do
    echo $no
    let no++
done
```

-- or --

```
read -p "Enter the value of the first number: " n1
read -p "Enter the value of the second number: " n2
for ((no=n1; no<=n2; no++))
do
    echo $no
done
```

78. Write a shell script to process the natural numbers between `1` and `15`. If the number is divisible by 3, display the number followed by `divisible by 3`. If the number is not divisible by 3, display the number followed by `not divisible by 3`.

    Solution

```
n=1
while [ "$n" -le 15 ]
do
    let rem=n%3
    if [ "$rem" -eq 0 ]
    then
        echo $n divisible by 3
    else
        echo $n not divisible by 3
    fi
    let n++
done
```

79. Write a shell script to accept a number `n` and display `n` lines of the following pattern:

```
*
**
***
****
*****
******
```

Solution

```
read -p "Enter a number: " n
i=1
while [ "$i" -le "$n" ]
do
    j=1
    while [ "$j" -le "$i" ]
    do
        echo -n '*'
        let j++
    done
    echo
    let i++
done
```

-- or --

```
read -p "Enter a number: " n
for ((i=1; i<=n; i++))
do
    for ((j=1; j<=i; j++))
    do
        echo -n '*'
    done
    echo
done
```

80. Write a shell script to accept a number `n` and display `n` lines of the following pattern:

```
1
12
123
1234
12345
123456
```

Solution

```
read -p "Enter a number: " n
i=1
while [ "$i" -le "$n" ]
do
    j=1
    while [ "$j" -le "$i" ]
```

```
    do
        echo -n "$j"
        let j++
    done
    echo
    let i++
done
```

-- or --

```
read -p "Enter a number: " n
for ((i=1; i<=n; i++))
do
    for ((j=1; j<=i; j++))
    do
        echo -n "$j"
    done
    echo
done
```

81. Write a shell script to accept a number `n` and display `n` lines of the following pattern:

```
******
*****
****
***
**
*
```

Solution

```
read -p "Enter a number: " n
i=1
while [ "$i" -le "$n" ]
do
    j="$n"
    while [ "$j" -ge "$i" ]
    do
        echo -n "*"
        let j--
    done
    echo
    let i++
done
```

-- or --

```
read -p "Enter a number: " n
for ((i=1; i<=n; i++))
do
    for ((j=n; j>=i; j--))
    do
        echo -n "*"
    done
    echo
done
```

82. Write a shell script to accept a number `n` and display `n` lines of the following pattern:

```
     *
    *
   *
  *
 *
*
```

Solution

```
read -p "Enter a number: " n
i=1
while [ "$i" -le "$n" ]
do
    j="$n"
    while [ "$j" -gt "$i" ]
    do
        echo -n " "
        let j--
    done
    echo "*"
    let i++
done
```

-- or --

```
read -p "Enter a number: " n
for ((i=1; i<=n; i++))
do
    for ((j=n; j>i; j--))
    do
        echo -n " "
```

```
    done
    echo "*"
done
```

83. Write a shell script to accept a number `n` and display `n` lines of the following pattern:

```
     *
    **
   ***
  ****
 *****
******
```

Solution

```
read -p "Enter a number: " n
i=1
while [ "$i" -le "$n" ]
do
    j="$n"
    while [ "$j" -gt "$i" ]
    do
        echo -n " "
        let j--
    done
    while [ "$j" -gt 0 ]
    do
        echo -n "*"
        let j--
    done
    echo
    let i++
done
```

-- or --

```
read -p "Enter a number: " n
for ((i=1; i<=n; i++))
do
    for ((j=n; j>i; j--))
    do
        echo -n " "
    done
    for (( ; j>0; j--))
    do
        echo -n "*"
```

```
    done
    echo
done
```

84. Write a shell script to accept a number `n` and display `n` lines of the following pattern:

```
6
65
654
6543
65432
654321
```

Solution

```
read -p "Enter a number: " n
i="$n"
while [ "$i" -ge 1 ]
do
    j="$n"
    while [ "$j" -ge "$i" ]
    do
        echo -n "$j"
        let j--
    done
    echo
    let i--
done
```

-- or --

```
read -p "Enter a number: " n
for ((i=n; i>=1; i--))
do
    for ((j=n; j>=i; j--))
    do
        echo -n "$j"
    done
    echo
done
```

85. Write a shell script to accept a number `n` and display `n` lines of the following pattern:

```
*    *
**   *
* *  *
*  * *
*   **
*    *
```

Solution

```
read -p "Enter a number: " n
i=1
while [ "$i" -le "$n" ]
do
    echo -n "*"
    j=2
    while [ "$j" -lt "$i" ]
    do
        echo -n " "
        let j++
    done
    if [ "$i" -gt 1 ]
    then
        echo -n "*"
    fi
    j=1
    let count=n-i
    while [ "$j" -lt "$count" ]
    do
        echo -n " "
        let j++
    done
    if [ "$i" -lt "$n" ]
    then
        echo -n "*"
    fi
    echo
    let i++
done
```

-- or --

```
read -p "Enter a number: " n
for ((i=1; i<=n; i++))
do
    echo -n "*"
    for ((j=2; j<i; j++))
    do
        echo -n " "
```

```
    done
    if [ "$i" -gt 1 ]
    then
        echo -n "*"
    fi
    for ((j=1; j<(n-i); j++))
    do
        echo -n " "
    done
    if [ "$i" -lt "$n" ]
    then
        echo -n "*"
    fi
    echo
done
```

86. Write a shell script to accept a number `n` and display `n` lines of the following pattern:

```
*          *
**        **
* *      * *
*  *    *  *
*   * *   *
*    *    *
```

Solution

```
read -p "Enter a number: " n
i=1
while [ "$i" -le "$n" ]
do
    echo -n "*"
    j=2
    while [ "$j" -lt "$i" ]
    do
        echo -n " "
        let j++
    done
    if [ "$i" -gt 1 ]
    then
        echo -n "*"
    fi
    j=1
    let count=n*2-i*2-1
    while [ "$j" -le "$count" ]
    do
        echo -n " "
```

```
        let j++
    done
    if [ "$i" -lt "$n" ]
    then
        echo -n "*"
    fi
    j=2
    while [ "$j" -lt "$i" ]
    do
        echo -n " "
        let j++
    done
    if [ "$i" -gt 1 ]
    then
        echo -n "*"
    fi
    echo
    let i++
done
```

-- or --

```
read -p "Enter a number: " n
for ((i=1; i<=n; i++))
do
    echo -n "*"
    for ((j=2; j<i; j++))
    do
        echo -n " "
    done
    if [ "$i" -gt 1 ]
    then
        echo -n "*"
    fi
    for ((j=1; j<=(n*2-i*2-1); j++))
    do
        echo -n " "
    done
    if [ "$i" -lt "$n" ]
    then
        echo -n "*"
    fi
    for ((j=2; j<i; j++))
    do
        echo -n " "
    done
    if [ "$i" -gt 1 ]
    then
        echo -n "*"
    fi
```

```
    echo
done
```

87. Write a shell script to accept a number `n` and display `(n×2)+1` lines of the following pattern:

```
*     *
 *   *
  * *
   *
  * *
 *   *
*     *
```

Solution

```
read -p "Enter a number: " n
let n2=n*2+1
x=0
let y=n2-1
i=1
while [ "$i" -le "$n2" ]
do
    if [ "$x" -le "$y" ]
    then
        x1="$x"
        let x2=y-x-1
    else
        x1="$y"
        let x2=x-y-1
    fi
    j=1;
    while [ "$j" -le "$x1" ]
    do
        echo -n ' '
        let j++
    done
    echo -n '*'
    if [ $x -ne $n ]
    then
        j=1;
        while [ "$j" -le "$x2" ]
        do
            echo -n ' '
            let j++
        done
        echo -n '*'
    fi
```

```
    echo
    let x++
    let y--
    let i++
done
```

-- or --

```
read -p "Enter a number: " n
let n2=n*2+1
x=0
let y=n2-1
for ((i=1; i<=n2; i++))
do
    if [ "$x" -le "$y" ]
    then
        x1="$x"
        let x2=y-x-1
    else
        x1="$y"
        let x2=x-y-1
    fi
    for ((j=1; j<=x1; j++))
    do
        echo -n ' '
    done
    echo -n '*'
    if [ $x -ne $n ]
    then
        for ((j=1; j<=x2; j++))
        do
            echo -n ' '
        done
        echo -n '*'
    fi
    echo
    let x++
    let y--
done
```

88. Write a shell script to accept a number `n` and display the first `n` terms of the following series:

`1 3 5 7 9 11`

Solution

```
read -p "Enter a number: " n
```

```
i=1
while [ "$i" -le "$n" ]
do
    let x=i*2-1
    echo -n "$x "
    let i++
done
echo
```

-- or --

```
read -p "Enter a number: " n
for ((i=1; i<=n; i++))
do
    let x=i*2-1
    echo -n "$x "
done
echo
```

89. Write a shell script to accept a number `n` and display the first `n` terms of the Fibonacci series:

`1` `1` `2` `3` `5` `8` `13` `21` `34`

Solution

```
read -p "Enter a number: " n
n1=1
n2=1
i=1
while [ "$i" -le "$n" ]
do
    echo -n "$n1 "
    let n3=n1+n2
    n1="$n2"
    n2="$n3"
    let i++
done
echo
```

-- or --

```
read -p "Enter a number: " n
n1=1
n2=1
for ((i=1; i<=n; i++))
```

```
do
    echo -n "$n1 "
    let n3=n1+n2
    n1="$n2"
    n2="$n3"
done
echo
```

90. Write a shell script to accept two numbers `n1` and `n2` and display their total

    Solution

```
read -p "Enter the first number: " n1
read -p "Enter the second number: " n2
let result=n1+n2
echo $result
```

91. Write a shell script to accept two numbers `n1` and `n2` and display their difference

    Solution

```
read -p "Enter the first number: " n1
read -p "Enter the second number: " n2
let result=n1-n2
echo $result
```

92. Write a shell script to accept two numbers `n1` and `n2` and display their multiplication

    Solution

```
read -p "Enter the first number: " n1
read -p "Enter the second number: " n2
let result=n1*n2
echo $result
```

93. Write a shell script to accept two numbers `n1` and `n2` and display the quotient of their division

    Solution

```
read -p "Enter the first number: " n1
```

```
read -p "Enter the second number: " n2
let result=n1/n2
echo $result
```

94. Write a shell script to accept two numbers `n1` and `n2` and display the remainder of their their division

    Solution

```
read -p "Enter the first number: " n1
read -p "Enter the second number: " n2
let result=n1%n2
echo $result
```

95. Write a shell script to accept the principal amount `p`, the rate of interest `r` and the number of terms `n` and display the simple interest using the formula `p×r×n/100`

    Solution

```
read -p "Enter the principal amount: " p
read -p "Enter the rate of interest: " r
read -p "Enter the number of terms: " n
let interest=p*r*n/100
echo $interest
```

96. Write a shell script to accept two numbers `n1` and `n2` and an operator `op` (`op` may be `+`, `-`, `*` or `/`) and display the result of applying the operator to the numbers

    Solution

```
read -p "Enter the first number: " n1
read -p "Enter the second number: " n2
read -p "Enter the operator (+, -, * or /): " op
if [ "$op" = "+" ]
then
    let result=n1+n2
elif [ "$op" = "-" ]
then
    let result=n1-n2
elif [ "$op" = "*" ]
then
    let result=n1*n2
elif [ "$op" = "/" ]
```

```
then
    let result=n1/n2
else
    result="ERROR: Invalid operator"
fi
echo $result
```

97. Write a shell script to accept marks in 7 subjects and display their total

Solution

```
read -p "Enter marks in the subject-1: " m1
read -p "Enter marks in the subject-2: " m2
read -p "Enter marks in the subject-3: " m3
read -p "Enter marks in the subject-4: " m4
read -p "Enter marks in the subject-5: " m5
read -p "Enter marks in the subject-6: " m6
read -p "Enter marks in the subject-7: " m7
let total=m1+m2+m3+m4+m5+m6+m7
echo "Total marks=$total"
```

98. Write a shell script to accept marks in 7 subjects and display the percentage

Solution

```
read -p "Enter marks in the subject-1: " m1
read -p "Enter marks in the subject-2: " m2
read -p "Enter marks in the subject-3: " m3
read -p "Enter marks in the subject-4: " m4
read -p "Enter marks in the subject-5: " m5
read -p "Enter marks in the subject-6: " m6
read -p "Enter marks in the subject-7: " m7
let total=m1+m2+m3+m4+m5+m6+m7
let percentage=total/7*100
echo "Percentage=$percentage"
```

99. Write a shell script to accept marks in 7 subjects and display `PASS` if the percentage is greater than or equal to 40 and `DETAINED` otherwise

Solution

```
read -p "Enter marks in the subject-1: " m1
read -p "Enter marks in the subject-2: " m2
read -p "Enter marks in the subject-3: " m3
read -p "Enter marks in the subject-4: " m4
```

```
read -p "Enter marks in the subject-5: " m5
read -p "Enter marks in the subject-6: " m6
read -p "Enter marks in the subject-7: " m7
let total=m1+m2+m3+m4+m5+m6+m7
let percentage=total/7*100
if [ "$percentage" -ge 40 ]
then
    echo PASS
else
    echo DETAINED
fi
```

100. Write a shell script to accept 7 numbers and display their total

Solution

```
sum=0
i=1
while [ "$i" -le 7 ]
do
    read -p "Enter a number: " no
    let sum+=no
    let i++
done
echo "The total is $sum"
```

101. Write a shell script to store the output of the `pwd` command into the file `list`

Solution

```
pwd >list
```

102. Write a shell script to append the output of the `ls -l` command to the file `list`

Solution

```
ls -l >>list
```

103. Write a shell script to run the command `ls /non-existent` and store its standard error into the file `err`

Solution

```
ls /non-existent 2> err
```

104. Write a shell script to append the standard error of the command `ls / | grep non-existent` to the file `err`

Solution

```
ls / | grep non-existent 2>> err
```

105. Write a shell script `sum.sh` to accept two numbers from the standard input and display their sum on the standard output. Create a file `input.txt` with the numbers 10 and 20 on two lines. Run the script so that it takes input from `input.txt` and produces output on standard output

Solution

```
# input.txt
10
20

# sum.sh
read -p "Enter the first number: " no1
read -p "Enter the second number: " no2
let sum=no1+no2
echo "The sum is $sum"

# Command line
bash sum.sh < input.txt
```

106. Write a shell script `sum.sh` to accept two numbers from the standard input and display their sum on the standard output. Create a file `input.txt` with the numbers 10 and 20 on two lines. Run the script so that it takes input from `input.txt` and stores the output in `output.txt`

Solution

```
# input.txt
10
20

# sum.sh
read -p "Enter the first number: " no1
read -p "Enter the second number: " no2
let sum=no1+no2
```

```
echo "The sum is $sum"

# Command line
bash sum.sh < input.txt >output.txt
```

107. Write a shell script to take the contents of all files whose name begins with `sedf` and search for `pass` in it in a case-insensitive way

Solution

```
cat sedf* | grep -i pass
```

108. Create a CSV file `employees.csv` containing the following employee records with fields employee number,name,department number,designation,address,area,state code and zip code,masked phone number,CTC (salary)

```
101,Norma Whitaker,0,CEO,757 Freeton Blvd,Clearwater,FL
33575,(813) xxx-xxxx,3500000
102,Ismael Gillespie,4,Manager,512 Anton Dr,Arlington,TX
76010,(817) xxx-xxxx,2000000
103,Michele Garrett,2,Manager,512 Tulip St,Austin,TX
78710,(512) xxx-xxxx,2200000
104,Daryl Whitley,3,Manager,131 Sharon Rd,Rome,GA
30161,(404) xxx-xxxx,2000000
105,Elma Brooks,5,Manager,480 Stonehedge Blvd,Garland,TX
75040,(903) xxx-xxxx,1800000
106,Jeremy Hurley,2,Executive,864 Fairfield Rd,Irving,TX
75061,(903) xxx-xxxx,800000
107,Leslie Stanley,3,Executive,903 Erming Ln,Roanoke,VA
24022,(703) xxx-xxxx,600000
108,Madelyn Raymond,2,Executive,788 Cedarwood Ln,Berkeley,CA
94704,(512) xxx-xxxx,500000
109,Audra Tyler,4,Executive,603 Freeton Blvd,Knoxville,TN
37901,(615) xxx-xxxx,600000
110,Maynard Good,1,Executive,682 Erming Ln,Albany,NY
12212,(518) xxx-xxxx,400000
```

Solution

```
nano employees.csv
```

109. Write a shell script to display all managers from employees.csv

Solution

```
cat employees.csv | grep Manager
```

110. Write a shell script to display all employees living in Texas (state code TX)

    Solution

    ```
    cat employees.csv | grep TX
    ```

111. Write a shell script to display all employees living in either Texas (state code TX) or Tennessee (state code TN)

    Solution

    ```
    cat employees.csv | grep 'T[XN]'
    ```

112. Write a shell script to display all employees living in either Texas (state code TX) or New York (state code NY)

    Solution

    ```
    cat employees.csv | grep -e 'TX' -e 'NY'
    ```

    ```

113. Write a shell script to display all employees whose first name negins with `M`

    **Solution**

    ```
    cat employees.csv | grep '^[0-9]*,M'
    ```

114. Write a shell script to display all employees working in department number 2

    **Solution**

    ```
    cat employees.csv | grep '^[0-9]*,[a-zA-Z ]*,2,'
    ```

115. Write a shell script to display only the name, designation, address, area and state+zip code for all employees
    ```

Solution

```
cat employees.csv | cut -d, -f2,4-7
```

116. Write a shell script to display only the name, designation, address, area and state+zip code for all employees working in department number 2

Solution

```
cat employees.csv | grep '^[0-9]*,[a-zA-Z ]*,2,' | cut -d,
        -f2,4-7
```

117. Write a shell script to count the number of employees in the company

Solution

```
cat employees.csv | wc -l
```

118. Write a shell script to count the number of managers in the company

Solution

```
cat employees.csv | grep Manager | wc -l
```

-- OR --

```
cat employees.csv | grep -c Manager
```

119. Write a shell script to display all the employees getting 7-figure salary

Solution

```
cat employees.csv | grep ',[0-9][0-9][0-9][0-9][0-9]
        [0-9][0-9]$'
```

120. Write a shell script to display the number of employees whose area code in telephone number is `512`

Solution

```
cat employees.csv | grep '(512)' | wc -l
```

-- OR --

```
cat employees.csv | grep -c '(512)'
```

121. Write a shell script to display the names of all the employees whose house number in the address is 512

Solution

```
cat employees.csv | grep ',512 ' | cut -d, -f2
```

122. Write a shell script to sort the employees on their name

Solution

```
cat employees.csv | sort -t, -k2,2
```

123. Write a shell script to sort the employees on their name in descending order

Solution

```
cat employees.csv | sort -t, -k2,2r
```

124. Write a shell script to sort the employees on their department number. When the department number is same, the sorting should be on employee number

Solution

```
cat employees.csv | sort -t, -k3,3n -k1,1n
```

125. Write a shell script to sort the employees on their salary

Solution

```
cat employees.csv | sort -t, -k9,9n
```

126. Write a shell script to sort the employees on their salary in reverse order

Solution

```
cat employees.csv | sort -t, -k9,9nr
```

127. Write a shell script to sort the employees on their department number. When the department number is same, the sorting should be on descending order of salary

Solution

```
cat employees.csv | sort -t, -k3,3n -k9,8nr
```

128. Write a shell script to convert employee.csv to all upper case (capital letters)

Solution

```
cat employees.csv | tr 'a-z' 'A-Z'
```

129. Write a shell script to convert employee.csv to all lower case (small letters)

Solution

```
cat employees.csv | tr 'A-Z' 'a-z'
```
