1. Display top 10 processes in descending order

Answer: ps aux --sort=-%mem | head -n 11

2. Display processes with highest memory usage

Answer: ps aux --sort=-%mem | head -n 1

3. Display current logged in user and logname

Answer: echo "Logged in user: $(whoami), Logname: $(logname)"

4. Display current shell, home directory, operating system type, current path setting, current working directory

Answer:

```
echo "Shell: $SHELL"
echo "Home directory: $HOME"
echo "OS type: $(uname -o)"
echo "Path setting: $PATH"
echo "Current working directory: $(pwd)"
```

5. Display OS version, release number, kernel version

Answer: lsb_release -a or uname -a

6. Write a command to display the first 15 columns from each line in the file

Answer: cut -c1-15 <filename>

7. Cut specified columns from a file and display them

Answer: cut -d' ' -f2,4 <filename>

8. Sort given file ignoring upper and lower case

Answer: sort -f <filename>

9. Displays only directories in current working directory

Answer: ls -l | grep '^d'

10.Copying files from one place to another

Answer: cp /path/to/source /path/to/destination

11.Moving files from one place to another

Answer: mv /path/to/source /path/to/destination

12.Removing specific directory with various options

Answer: rm -r /path/to/directory

13.List the numbers of users currently logged in the system and then sort it

Answer: who | cut -d' ' -f1 | sort | uniq -c | sort -nr

14.Merge two files into one file

Answer: cat file1 file2 > merged_file

15.Change the access mode of one file

Answer: chmod <permissions> <filename>

16. Display the last ten lines of the file

Answer: tail -n 10 <filename>

17. To locate files in a directory and in a subdirectory

Answer: find /path/to/directory -name 'filename'

18. Display the contents of all files having a name starting with 'ap' followed by any number of characters

Answer: cat ap*.*

19. Rename any file aaa to aaa.aa1, where aa1 is the user login name

Answer: mv aaa aaa.$(whoami)

20. Write a command to search the word ‘picture’ in the file and if found, display the lines containing it on the screen.

Command: grep 'picture' <filename>

21. Write a command to search for all occurrences of ‘Rebecca’ as well as ‘rebecca’ in a file and display the lines that contain either of these words.

Command: grep -i 'Rebecca\|rebecca' <filename>

22. Write a command to search all four-letter words whose first letter is a ‘b’ and last letter is a ‘k’.
    Command: grep '\<b..k\>' <filename>
    Explanation: \< and \> denote the beginning and end of a word respectively, and b..k represents any four-letter word that starts with 'b' and ends with 'k', with any two characters in between.

23. Write a command to display only those lines which do not contain the search patterns.

Command: grep -v 'pattern1\|pattern2' <filename>

Replace 'pattern1\|pattern2' with the actual patterns to exclude from the output. This command displays lines that don't contain any of the specified patterns.

48. Program where parent process sorts array elements in descending order and child process sorts array elements in ascending order.

Ans.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

void bubbleSort(int arr[], int n, int order) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (order == 0) { // Ascending order
                if (arr[j] > arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            } else { // Descending order
                if (arr[j] < arr[j + 1]) {
```

```
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    int status;

    pid_t pid = fork();

    if (pid < 0) {
        fprintf(stderr, "Fork failed\n");
        return 1;
    } else if (pid == 0) { // Child process
        printf("Child process sorting array in ascending order:\n");
        bubbleSort(arr, n, 0); // Sorting in ascending order
        for (int i = 0; i < n; i++) {
            printf("%d ", arr[i]);
        }
```

```
        printf("\n");
    } else { // Parent process
        waitpid(pid, &status, 0);
        printf("Parent process sorting array in descending order:\n");
        bubbleSort(arr, n, 1); // Sorting in descending order
        for (int i = 0; i < n; i++) {
            printf("%d ", arr[i]);
        }
        printf("\n");
    }

    return 0;
}
```

49. Program where parent process Counts number of vowels in the given sentence and child process will count number of words in the same sentence. The above programs should use UNIX calls like fork, exec and wait. And also show the orphan and zombie states

Ans.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int isVowel(char c) {
    c = tolower(c);
```

```
    return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');
}

int countVowels(const char *sentence) {
    int count = 0;
    for (int i = 0; sentence[i] != '\0'; i++) {
        if (isVowel(sentence[i])) {
            count++;
        }
    }
    return count;
}

int countWords(const char *sentence) {
    int count = 0;
    int inWord = 0;
    for (int i = 0; sentence[i] != '\0'; i++) {
        if (sentence[i] == ' ' || sentence[i] == '\t' || sentence[i] == '\n') {
            inWord = 0;
        } else if (inWord == 0) {
            inWord = 1;
            count++;
        }
    }
    return count;
```

```
}

int main() {
    char sentence[100];
    printf("Enter a sentence: ");
    fgets(sentence, sizeof(sentence), stdin);

    pid_t pid = fork();

    if (pid < 0) {
        fprintf(stderr, "Fork failed\n");
        return 1;
    } else if (pid == 0) { // Child process
        int words = countWords(sentence);
        printf("Child Process - Number of words in the sentence: %d\n", words);
    } else { // Parent process
        int status;
        waitpid(pid, &status, 0);

        if (WIFEXITED(status)) {
            int vowels = countVowels(sentence);
            printf("Parent Process - Number of vowels in the sentence: %d\n", vowels);
        }
    }
```

```
    // To demonstrate orphan state
    sleep(5); // Parent process sleeps for 5 seconds before terminating

    return 0;
}
```

50. Write Shell script to copy files from one folder to another

Ans.

```
cp /path/to/source/* /path/to/destination/
```

51. Write Shell script Count number of words, characters and lines.

Ans.

```
#!/bin/bash
echo "Number of words: $(wc -w < $1)"
echo "Number of characters: $(wc -m < $1)"
echo "Number of lines: $(wc -l < $1)"
```

52. Write Shell script To describe files in different format.

Ans.

```
#!/bin/bash
file "$1"
stat "$1"
ls -l "$1"
```

Explanation: Execute this script by passing the filename as an argument (bash script.sh filename).

53. Write Shell script to find factorial of given number using bash script

Ans.

```
#!/bin/bash
echo -n "Enter a number: "
read num
fact=1
for ((i = 1; i <= num; i++)); do
   fact=$((fact * i))
done
echo "Factorial of $num is $fact"
```

54. Display first 10 natural numbers using bash script.

Ans.

```
#!/bin/bash
for ((i = 1; i <= 10; i++)); do
   echo "$i"
done
```

55. Display Fibonacci series using bash script

Ans.

```
#!/bin/bash
echo -n "Enter the number of terms: "
read n
a=0
b=1
echo -n "Fibonacci Series: "
for ((i = 0; i < n; i++)); do
    echo -n "$a "
    fn=$((a + b))
    a=$b
    b=$fn
done
echo
```

56. Find given number is prime or nor using bash script.

Ans.

```
#!/bin/bash
echo -n "Enter a number: "
read num

if [ "$num" -eq 1 ]; then
    echo "$num is not a prime number"
    exit
```

```
fi

for ((i = 2; i <= num / 2; i++)); do
    if [ $((num % i)) -eq 0 ]; then
        echo "$num is not a prime number"
        exit
    fi
done

echo "$num is a prime number"
```

57. Write shell script to finding biggest of three numbers

Ans.

```
#!/bin/bash
echo -n "Enter three numbers: "
read num1 num2 num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]; then
    echo "$num1 is the biggest number"
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]; then
    echo "$num2 is the biggest number"
else
    echo "$num3 is the biggest number"
fi
```

58. Write shell script to reversing a number

Ans.

```
#!/bin/bash
echo -n "Enter a number: "
read num

rev=0
while [ $num -gt 0 ]; do
   rem=$((num % 10))
   rev=$((rev * 10 + rem))
   num=$((num / 10))
done

echo "Reversed number: $rev"
```

59. Write shell script find Sum of individual digits (1234 -> 1+2+3+4=10)

Ans.

```
#!/bin/bash
echo -n "Enter a number: "
read num

sum=0
while [ $num -gt 0 ]; do
   rem=$((num % 10))
   sum=$((sum + rem))
```

```
    num=$((num / 10))
done

echo "Sum of digits: $sum"
```