Madan Bhandari Memorial College

Lab No. 10 **Date: 2025/09/08**

Title: Write a program to calculate the seek time for user input pending request, total no of cylinders and current position of I/O read/write head using FCFS disk scheduling algorithm.

Introduction:
FCFS is the simplest disk scheduling algorithm where requests are serviced in the order they arrive. It is non-preemptive, easy to implement and ensures that all requests are handled fairly without starvation. However, the performance may vary depending on the order of requests, and the average seek time can be high.

Algorithm Steps:

1. Maintain a queue of disk requests.
2. Serve each request in the order it arrives.
3. Move the disk head to the requested track and continue to the next request in the queue.

Advantages:

- Simple and fair.

Disadvantages:

- High average seek time in some cases.

Programming Language: C++

IDE: Dev C++

Source Code:

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
   int n, head, cylinders, seek = 0;
   cout << "==================================" << endl;
   cout << "||      FCFS Disk Scheduling   ||" << endl;
   cout << "|| Compiled by :- Jonash Chataut  ||" << endl;
   cout << "==================================" << endl << endl;
   cout << "Enter total number of cylinders: ";
   cin >> cylinders;
   cout << "Enter number of requests: ";
   cin >> n;
   int arr[n];
   cout << "Enter the request sequence: ";
   for (int i = 0; i < n; i++)
      cin >> arr[i];
   cout << "Enter initial head position: ";
   cin >> head;
   int current = head;
   cout << "\nSeek Sequence with Movements:" << endl;
   for (int i = 0; i < n; i++)
   {
      int move = abs(current - arr[i]);
      cout << "Move from " << current << " -> " << arr[i] << " (Seek = " << move << ")" << endl;
      seek += move;
      current = arr[i];
   }
   // Print head movements
   current = head;
   cout << "\nHead movements: " << current;
   for (int i = 0; i < n; i++)
      cout << " -> " << arr[i];
   cout << "\n\nTotal seek time = " << seek << endl;
   cout << "Average seek time = " << fixed << setprecision(2) << (float)seek / n << endl;
   return 0;
}
```

Output:

```
C:\csit\fourth_sem_jonash\OS
==================================
||        FCFS Disk Scheduling    ||
|| Compiled by :- Jonash Chataut  ||
==================================

Enter total number of cylinders: 200
Enter number of requests: 10
Enter the request sequence: 12 45 78 89 120 56 45 12 30 77
Enter initial head position: 19

Seek Sequence with Movements:
Move from 19 -> 12 (Seek = 7)
Move from 12 -> 45 (Seek = 33)
Move from 45 -> 78 (Seek = 33)
Move from 78 -> 89 (Seek = 11)
Move from 89 -> 120 (Seek = 31)
Move from 120 -> 56 (Seek = 64)
Move from 56 -> 45 (Seek = 11)
Move from 45 -> 12 (Seek = 33)
Move from 12 -> 30 (Seek = 18)
Move from 30 -> 77 (Seek = 47)

Head movements: 19 -> 12 -> 45 -> 78 -> 89 -> 120 -> 56 -> 45 -> 12 -> 30 -> 77

Total seek time = 288
Average seek time = 28.80
```

Madan Bhandari Memorial College

Lab No. 11 **Date: 2025/09/08**

Title: Write a program to calculate the seek time for user input pending request, total no of cylinders and current position of I/O read/write head using SCAN disk scheduling algorithm.

Introduction:
SCAN also called the elevator algorithm, moves the disk head in one direction servicing all requests until it reaches the end of the disk, then reverses direction. This approach reduces large variations in response time and ensures that all requests are eventually serviced.

Algorithm Steps:

1. Decide the initial direction of head movement (toward higher or lower track numbers).
2. Move the head in that direction, servicing all pending requests.
3. At the end, reverse the direction and continue servicing requests in the opposite direction.

Advantages:

- More efficient than FCFS.
- Reduces the chance of starvation.

Disadvantages:

- Requests at the far end of the disk may have to wait longer.

Programming Language: C++

IDE: Dev C++

Source Code:

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
   int n, head, cylinders, seek = 0;
   char direction;
   cout << "====================================" << endl;
   cout << "||      SCAN Disk Scheduling    ||" << endl;
   cout << "|| Compiled by :- Jonash Chataut  ||" << endl;
   cout << "====================================" << endl << endl;
   cout << "Enter total number of cylinders: ";
   cin >> cylinders;
   cout << "Enter number of requests: ";
   cin >> n;
   int arr[n], left[20], right[20];
   int lcount = 0, rcount = 0;
   cout << "Enter the request sequence: ";
   for (int i = 0; i < n; i++)
      cin >> arr[i];
   cout << "Enter initial head position: ";
   cin >> head;
   cout << "Enter direction (L/R): ";
   cin >> direction;
   // Split requests
   for (int i = 0; i < n; i++)
   {
      if (arr[i] < head)
         left[lcount++] = arr[i];
      else
         right[rcount++] = arr[i];
   }
   if (direction == 'L')
      left[lcount++] = 0;
   else
      right[rcount++] = cylinders - 1;
   // Sort arrays
   for (int i = 0; i < rcount - 1; i++)
      for (int j = i + 1; j < rcount; j++)
         if (right[i] > right[j])
            swap(right[i], right[j]);
   for (int i = 0; i < lcount - 1; i++)
      for (int j = i + 1; j < lcount; j++)
         if (left[i] < left[j])
            swap(left[i], left[j]);
   // Step-by-step movements
   cout << "\nSeek Sequence with Movements:" << endl;
   int current = head;
```

```
  int movement[n + 2], mcount = 0;
  if (direction == 'R')
  {
    for (int i = 0; i < rcount; i++)
    {
      int move = abs(current - right[i]);
      cout << "Move from " << current << " -> " << right[i] << " (Seek = " << move << ")" << endl;
      seek += move;
      current = right[i];
      movement[mcount++] = current;
    }
    for (int i = 0; i < lcount; i++)
    {
      int move = abs(current - left[i]);
      cout << "Move from " << current << " -> " << left[i] << " (Seek = " << move << ")" << endl;
      seek += move;
      current = left[i];
      movement[mcount++] = current;
    }
  }
  else
  {
    for (int i = lcount - 1; i >= 0; i--)
    {
      int move = abs(current - left[i]);
      cout << "Move from " << current << " -> " << left[i] << " (Seek = " << move << ")" << endl;
      seek += move;
      current = left[i];
      movement[mcount++] = current;
    }
    for (int i = 0; i < rcount; i++)
    {
      int move = abs(current - right[i]);
      cout << "Move from " << current << " -> " << right[i] << " (Seek = " << move << ")" << endl;
      seek += move;
      current = right[i];
      movement[mcount++] = current;
    }
  }
  // Head movements line
  cout << "\nHead movements: " << head;
  for (int i = 0; i < mcount; i++)
    cout << " -> " << movement[i];
  cout << "\n\nTotal seek time = " << seek << endl;
  cout << "Average seek time = " << fixed << setprecision(2) << (float)seek / n << endl;
  return 0;
}
```

Output:

```
C:\csit\fourth_sem_jonash\OS
==================================
||        SCAN Disk Scheduling    ||
|| Compiled by :- Jonash Chataut  ||
==================================

Enter total number of cylinders: 150
Enter number of requests: 10
Enter the request sequence: 35 70 45 15 60 20 80 90 130 75
Enter initial head position: 30
Enter direction (L/R): L

Seek Sequence with Movements:
Move from 30 -> 0 (Seek = 30)
Move from 0 -> 15 (Seek = 15)
Move from 15 -> 20 (Seek = 5)
Move from 20 -> 35 (Seek = 15)
Move from 35 -> 45 (Seek = 10)
Move from 45 -> 60 (Seek = 15)
Move from 60 -> 70 (Seek = 10)
Move from 70 -> 75 (Seek = 5)
Move from 75 -> 80 (Seek = 5)
Move from 80 -> 90 (Seek = 10)
Move from 90 -> 130 (Seek = 40)

Head movements: 30 -> 0 -> 15 -> 20 -> 35 -> 45 -> 60 -> 70 -> 75 -> 80 -> 90 -> 130

Total seek time = 160
Average seek time = 16.00
```

Madan Bhandari Memorial College

Lab No. 12 **Date: 2025/09/08**

Title: Write a program to calculate the seek time for user input pending request, total no of cylinders and current position of I/O read/write head using LOOK disk scheduling algorithm.

Introduction:
LOOK is a modified version of SCAN. The head only moves as far as the last request in the current direction before reversing, instead of going all the way to the end of the disk. This reduces unnecessary head movement and improves efficiency.

Algorithm Steps:

1. Move the head in the initial direction, servicing requests until the last request in that direction.
2. Reverse direction and continue servicing.

Advantages:

- Optimized SCAN with less head movement.
- More efficient and faster.

Disadvantages:

- Slightly more complex than SCAN.

Programming Language: C++

IDE: Dev C++

Source Code:

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main()
{
   int n, head, cylinders, seek = 0;
   char direction;
   cout << "====================================" << endl;
   cout << "||      LOOK Disk Scheduling    ||" << endl;
   cout << "|| Compiled by :- Jonash Chataut  ||" << endl;
   cout << "====================================" << endl << endl;
   cout << "Enter total number of cylinders: ";
   cin >> cylinders;
   cout << "Enter number of requests: ";
   cin >> n;
   int arr[n], left[20], right[20];
   int lcount = 0, rcount = 0;
   cout << "Enter the request sequence: ";
   for (int i = 0; i < n; i++)
      cin >> arr[i];
   cout << "Enter initial head position: ";
   cin >> head;
   cout << "Enter direction (L/R): ";
   cin >> direction;
   for (int i = 0; i < n; i++)
   {
      if (arr[i] < head)
         left[lcount++] = arr[i];
      else
         right[rcount++] = arr[i];
   }
   // Sort arrays
   for (int i = 0; i < rcount - 1; i++)
      for (int j = i + 1; j < rcount; j++)
         if (right[i] > right[j])
            swap(right[i], right[j]);
   for (int i = 0; i < lcount - 1; i++)
      for (int j = i + 1; j < lcount; j++)
         if (left[i] < left[j])
            swap(left[i], left[j]);
   cout << "\nSeek Sequence with Movements:" << endl;
   int current = head;
   int movement[n], mcount = 0;
   if (direction == 'R')
   {
      for (int i = 0; i < rcount; i++)
      {
         int move = abs(current - right[i]);
```

```
            cout << "Move from " << current << " -> " << right[i] << " (Seek = " << move << ")" << endl;
            seek += move;
            current = right[i];
            movement[mcount++] = current;
        }
        for (int i = lcount - 1; i >= 0; i--)
        {
            int move = abs(current - left[i]);
            cout << "Move from " << current << " -> " << left[i] << " (Seek = " << move << ")" << endl;
            seek += move;
            current = left[i];
            movement[mcount++] = current;
        }
    }
    else
    {
        for (int i = lcount - 1; i >= 0; i--)
        {
            int move = abs(current - left[i]);
            cout << "Move from " << current << " -> " << left[i] << " (Seek = " << move << ")" << endl;
            seek += move;
            current = left[i];
            movement[mcount++] = current;
        }
        for (int i = 0; i < rcount; i++)
        {
            int move = abs(current - right[i]);
            cout << "Move from " << current << " -> " << right[i] << " (Seek = " << move << ")" << endl;
            seek += move;
            current = right[i];
            movement[mcount++] = current;
        }
    }
    // Head movements line
    cout << "\nHead movements: " << head;
    for (int i = 0; i < mcount; i++)
        cout << " -> " << movement[i];
    cout << "\n\nTotal seek time = " << seek << endl;
    cout << "Average seek time = " << fixed << setprecision(2) << (float)seek / n << endl;

    return 0;
}
```

Output:

C:\csit\fourth_sem_jonash\OS

```
==================================
||        LOOK Disk Scheduling    ||
|| Compiled by :- Jonash Chataut  ||
==================================

Enter total number of cylinders: 200
Enter number of requests: 10
Enter the request sequence: 45 78 96 102 23 45 86 78 42 100
Enter initial head position: 49
Enter direction (L/R): R

Seek Sequence with Movements:
Move from 49 -> 78 (Seek = 29)
Move from 78 -> 78 (Seek = 0)
Move from 78 -> 86 (Seek = 8)
Move from 86 -> 96 (Seek = 10)
Move from 96 -> 100 (Seek = 4)
Move from 100 -> 102 (Seek = 2)
Move from 102 -> 23 (Seek = 79)
Move from 23 -> 42 (Seek = 19)
Move from 42 -> 45 (Seek = 3)
Move from 45 -> 45 (Seek = 0)

Head movements: 49 -> 78 -> 78 -> 86 -> 96 -> 100 -> 102 -> 23 -> 42 -> 45 -> 45

Total seek time = 154
Average seek time = 15.40
```

Lab No. 13 **Date: 2025/09/08**

Title: Write a program to test if the system is free from dead lock or not for the user input allocation, max and available matrix.

Introduction:
In multiprogramming systems, processes compete for limited resources like CPU, memory, and I/O devices. If processes hold some resources and wait indefinitely for others, the system may enter a deadlock. Deadlock prevents processes from making progress and must be detected or prevented.

The Banker's Algorithm is used to avoid deadlock by ensuring the system is always in a safe state. A state is considered safe if the system can allocate resources to each process in some order and still avoid deadlock.

- **Important Terms**
  - Allocation Matrix → Resources currently allocated to each process.
  - Max Matrix → Maximum resources each process may need.
  - Available Vector → Number of resources currently available in the system.
  - Need Matrix → Additional resources a process may still request, calculated as:

$$Need[i][j] = Max[i][j] - Allocation[i][j]$$

- **Algorithm:**

1. Initialization:
   - Work = Available
   - Finish[i] = false for all processes
2. Find a process Pi such that:
   - Finish[i] = false
   - Need[i] ≤ Work
3. If such a process is found:
   - Allocate its resources back: Work = Work + Allocation[i]
   - Mark Finish[i] = true
   - Add process to the safe sequence
4. Repeat until:
   - All processes finish → System is Safe (Deadlock-free)
   - No such process exists → System is Unsafe (Deadlock present)

Programming Language: C++

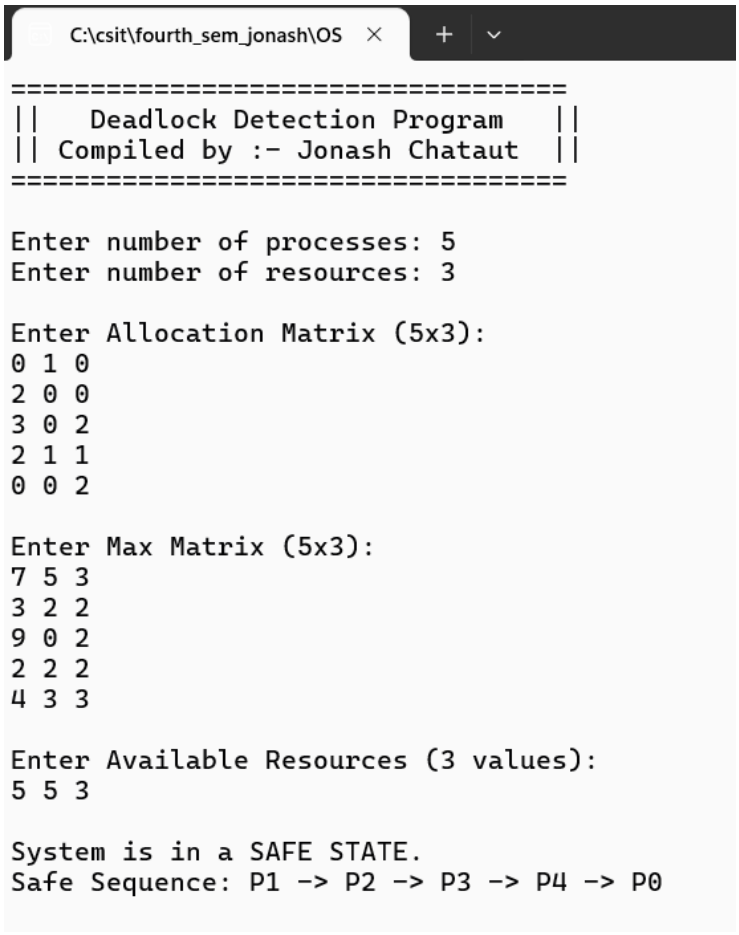
IDE: Dev C++

Source Code:

```
#include <iostream>
using namespace std;
int main()
{
   int n, m; // n = number of processes, m = number of resources
   cout << "====================================" << endl;
   cout << "||   Deadlock Detection Program   ||" << endl;
   cout << "|| Compiled by :- Jonash Chataut  ||" << endl;
   cout << "====================================" << endl << endl;
   cout << "Enter number of processes: ";
   cin >> n;
   cout << "Enter number of resources: ";
   cin >> m;
   int allocation[n][m], maxNeed[n][m], available[m];
   int need[n][m];
   cout << "\nEnter Allocation Matrix (" << n << "x" << m << "):\n";
   for (int i = 0; i < n; i++)
      for (int j = 0; j < m; j++)
         cin >> allocation[i][j];
   cout << "\nEnter Max Matrix (" << n << "x" << m << "):\n";
   for (int i = 0; i < n; i++)
      for (int j = 0; j < m; j++)
         cin >> maxNeed[i][j];
   cout << "\nEnter Available Resources (" << m << " values):\n";
   for (int j = 0; j < m; j++)
      cin >> available[j];
   // Calculate Need matrix
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < m; j++)
      {
         need[i][j] = maxNeed[i][j] - allocation[i][j];
      }
   }
   // Work array = available
   int work[m];
   for (int j = 0; j < m; j++)
      work[j] = available[j];
   bool finish[n];
   for (int i = 0; i < n; i++)
      finish[i] = false;
   int safeSeq[n], count = 0;
   // Safety Algorithm
   bool found;
   while (count < n)
   {
      found = false;
      for (int i = 0; i < n; i++)
      {
```

```
            if (!finish[i])
            {
                int j;
                for (j = 0; j < m; j++)
                {
                    if (need[i][j] > work[j])
                        break;
                }
                if (j == m)
                {
                    // process i can be satisfied
                    for (int k = 0; k < m; k++)
                        work[k] += allocation[i][k];
                    safeSeq[count++] = i;
                    finish[i] = true;
                    found = true;
                }
            }
        }
        if (!found)
            break; // no process could be satisfied
    }
    // Check if system is safe
    if (count == n)
    {
        cout << "\nSystem is in a SAFE STATE." << endl;
        cout << "Safe Sequence: ";
        for (int i = 0; i < n; i++)
        {
            cout << "P" << safeSeq[i];
            if (i != n - 1)
                cout << " -> ";
        }
        cout << endl;
    }
    else
    {
        cout << "\nSystem is in DEADLOCK (Not Safe)." << endl;
    }
    return 0;
}
```

Output:


C:\csit\fourth_sem_jonash\OS
==================================
|| Deadlock Detection Program ||
|| Compiled by :- Jonash Chataut ||
==================================
Enter number of processes: 5
Enter number of resources: 3
Enter Allocation Matrix (5x3):
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter Max Matrix (5x3):
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter Available Resources (3 values):
5 5 3
System is in a SAFE STATE.
Safe Sequence: P1 -> P2 -> P3 -> P4 -> P0

Lab No. 14 **Date: 2025/09/09**

Title: Prepare a lab report for basic linux commands.

Introduction:

Linux is a widely used open-source operating system that provides a powerful command-line interface (CLI) to interact with the system. Unlike GUI-based systems, the Linux shell allows users to execute commands directly, which provides better control and flexibility. Learning basic Linux commands is essential for system navigation, file management, process control, and overall system administration.

Basic linux commands:

1) pwd → Show current working directory

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ pwd
/home/jonash_chataut
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

2) ls → List files and directories

ls -a → Show hidden files

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ ls
jonash_chataut@LAPTOP-LA4DFAI2:~$ ls -a
.  ..  .bash_logout  .bashrc  .cache  .motd_shown  .profile
```

3) mkdir <dir> → Create new directory

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ mkdir jonashchataut
jonash_chataut@LAPTOP-LA4DFAI2:~$ ls
jonashchataut
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

4) touch <file> → Create empty file / update timestamp

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ touch jonash.txt
jonash_chataut@LAPTOP-LA4DFAI2:~$ ls
jonash.txt  jonashchataut
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

5) cat <file> → Display file contents

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ cat jonash.txt
Hi my name is Jonash Chataut
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

6) head <file> → Show first 10 lines

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ head -n 2 jonash.txt
Hi my name is Jonash Chataut
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

7) whoami → Show current user

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ whoami
jonash_chataut
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

8) who → Show logged-in users

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ who
jonash_chataut pts/1        2025-09-08 10:13
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

9) uname -a → Show system info (kernel, OS)

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ uname -a
Linux LAPTOP-LA4DFAI2 6.6.87.2-microsoft-standard-WSL2 #1 SMP PREEMPT_DYNAMI
C Thu Jun  5 18:30:46 UTC 2025 x86_64 x86_64 x86_64 GNU/Linux
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

10) date → Show current date and time

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ date
Mon Sep  8 10:42:18 UTC 2025
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

11) top → Live view of processes

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ top
top - 10:42:54 up 30 min,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  23 total,   1 running,  22 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.
MiB Mem :   7794.2 total,   7263.7 free,    505.5 used,    176.2 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7288.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
      1 root      20   0   21524  11984   9296 S   0.0   0.2   0:00.76
      2 root      20   0    3060   1664   1664 S   0.0   0.0   0:00.07
      6 root      20   0    3076   1792   1792 S   0.0   0.0   0:00.01
     55 root      19  -1   50356  15488  14592 S   0.0   0.2   0:00.19
    112 root      20   0   25112   6272   4864 S   0.0   0.1   0:00.18
    122 systemd+  20   0   21456  12160  10112 S   0.0   0.2   0:00.15
    123 systemd+  20   0   91024   7680   6912 S   0.0   0.1   0:00.09
    198 root      20   0    4236   2432   2304 S   0.0   0.0   0:00.01
    199 message+  20   0    9532   4992   4480 S   0.0   0.1   0:00.05
    207 root      20   0   17964   8576   7552 S   0.0   0.1   0:00.08
    237 root      20   0 1756096  12032  10112 S   0.0   0.2   0:00.15
    241 syslog    20   0  222508   5120   4224 S   0.0   0.1   0:00.08
    250 root      20   0    3160   1920   1792 S   0.0   0.0   0:00.01
    260 root      20   0    3116   1792   1664 S   0.0   0.0   0:00.00
    282 root      20   0  107012  22656  13312 S   0.0   0.3   0:00.09
    452 root      20   0    6692   4352   3712 S   0.0   0.1   0:00.00
    496 jonash_+  20   0   20296  11136   9088 S   0.0   0.1   0:00.06
    497 jonash_+  20   0   21152   3516   1792 S   0.0   0.0   0:00.00
    510 jonash_+  20   0    6056   5120   3584 S   0.0   0.1   0:00.02
    593 root      20   0    3076    896    896 S   0.0   0.0   0:00.00
    594 root      20   0    3076   1152   1024 S   0.0   0.0   0:00.08
    595 jonash_+  20   0    6068   5248   3584 S   0.0   0.1   0:00.14
    680 jonash_+  20   0    9292   5504   3328 R   0.0   0.1   0:00.01
```

12) grep <pattern> <file> → Search text in file.

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ grep "Hi" jonash.txt
Hi my name is Jonash Chataut
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

13) wc <file> → Count words, lines, characters

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ wc jonash.txt
 1  6 29 jonash.txt
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

14) uptime → Show system uptime

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ uptime
 10:46:53 up 34 min,  1 user,  load average: 0.00, 0.01, 0.00
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

15) df → Checks your Linux system's disk usage

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ df
Filesystem      1K-blocks      Used  Available Use% Mounted on
none              3990624         0    3990624   0% /usr/lib/modules/6.6.87.
2-microsoft-standard-WSL2
none              3990624         4    3990620   1% /mnt/wsl
drivers         498775036 430470748   68304288  87% /usr/lib/wsl/drivers
/dev/sdd       1055762868   1273660 1000785736   1% /
none              3990624        80    3990544   1% /mnt/wslg
none              3990624         0    3990624   0% /usr/lib/wsl/lib
rootfs            3985612      2664    3982948   1% /init
none              3990624       548    3990076   1% /run
none              3990624         0    3990624   0% /run/lock
none              3990624         0    3990624   0% /run/shm
none              3990624        76    3990548   1% /mnt/wslg/versions.txt
none              3990624        76    3990548   1% /mnt/wslg/doc
C:\             498775036 430470748   68304288  87% /mnt/c
tmpfs             3990624        16    3990608   1% /run/user/1000
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

16) ps → Summarizes the status of all running processes in your Linux system at a specific time

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ ps
    PID TTY          TIME CMD
    686 pts/0    00:00:00 bash
    712 pts/0    00:00:00 ps
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

17) tail → It view the last line.

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ tail jonash.txt
Hi my name is Jonash Chataut
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

18) history → Check previously run utilities.

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ history
    1  pwd
    2  ls
    3  ls a
    4  la -a
    5  cd home
    6  cd desktop
    7  ls
    8  ls -a
    9  mkdir jonashchataut
   10  ls
```

19) hostname → Show hostname

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ hostname
LAPTOP-LA4DFAI2
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

20) w → Detailed user sessions

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ w
 10:52:00 up 39 min,  1 user,  load average: 0.00, 0.01, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
jonash_c pts/1    -                10:13   38:10   0.02s  0.02s -bash
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

21) ping → Check connectivity

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ ping youtube.com
PING youtube.com (103.74.15.79) 56(84) bytes of data.
```

22) ip addr → Show IP details

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet 10.255.255.254/32 brd 10.255.255.254 scope global lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1492 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:fc:60:91 brd ff:ff:ff:ff:ff:ff
    inet 172.17.82.247/20 brd 172.17.95.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fefc:6091/64 scope link
       valid_lft forever preferred_lft forever
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

23) lsblk → List block devices

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ lsblk
NAME
    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda   8:0    0 388.4M  1 disk
sdb   8:16   0   186M  1 disk
sdc   8:32   0     2G  0 disk [SWAP]
sdd   8:48   0     1T  0 disk /mnt/wslg/distro
                               /
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

24) echo → Print text

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ echo "Linux Commands"
Linux Commands
jonash_chataut@LAPTOP-LA4DFAI2:~$
```

25) clear → Clear terminal

```
jonash_chataut@LAPTOP-LA4DFAI2:~$ echo "Linux Commands"
Linux Commands
jonash_chataut@LAPTOP-LA4DFAI2:~$ clear
```