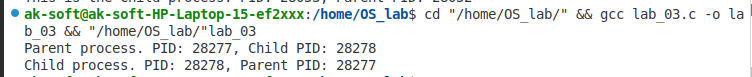
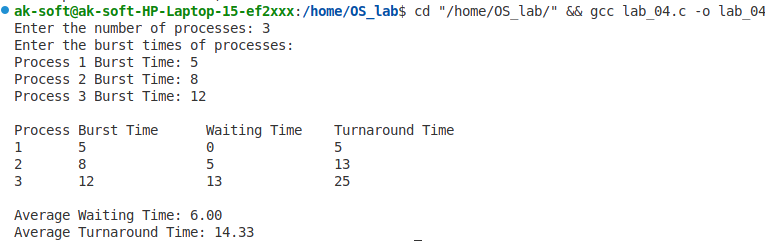
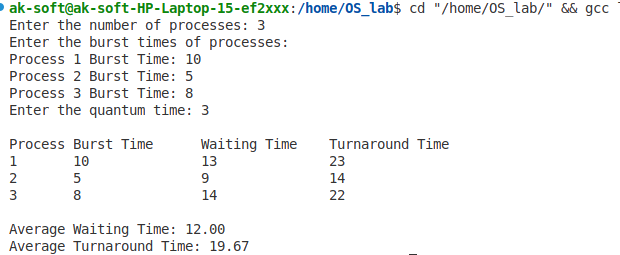
lab3



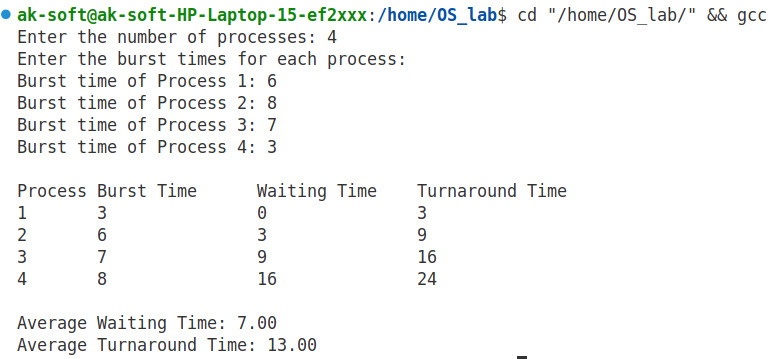
lab 4



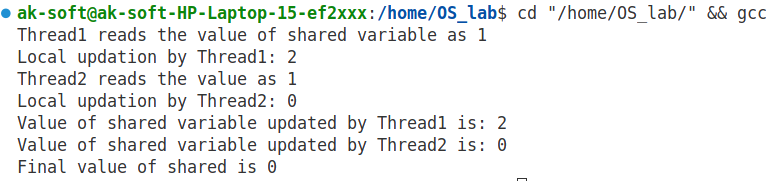
lab 5



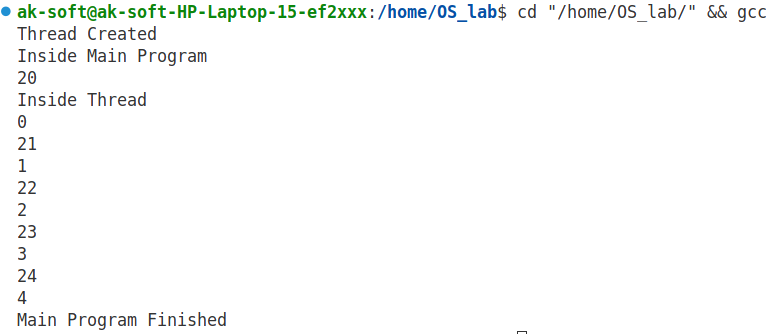
lab 6



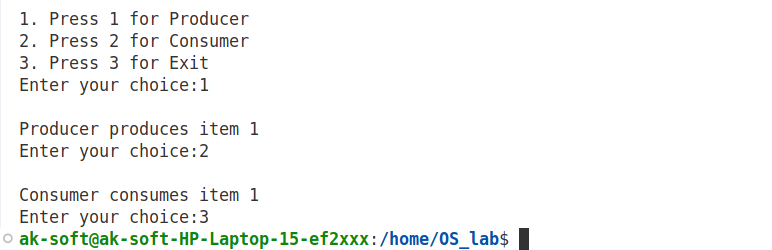
lab 7



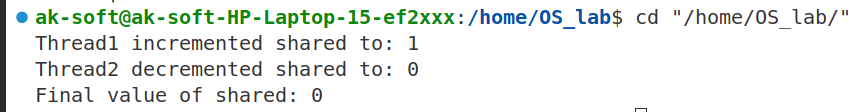
lab 8



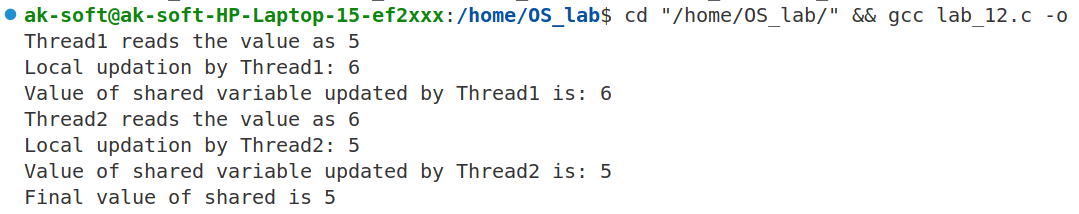
lab 9



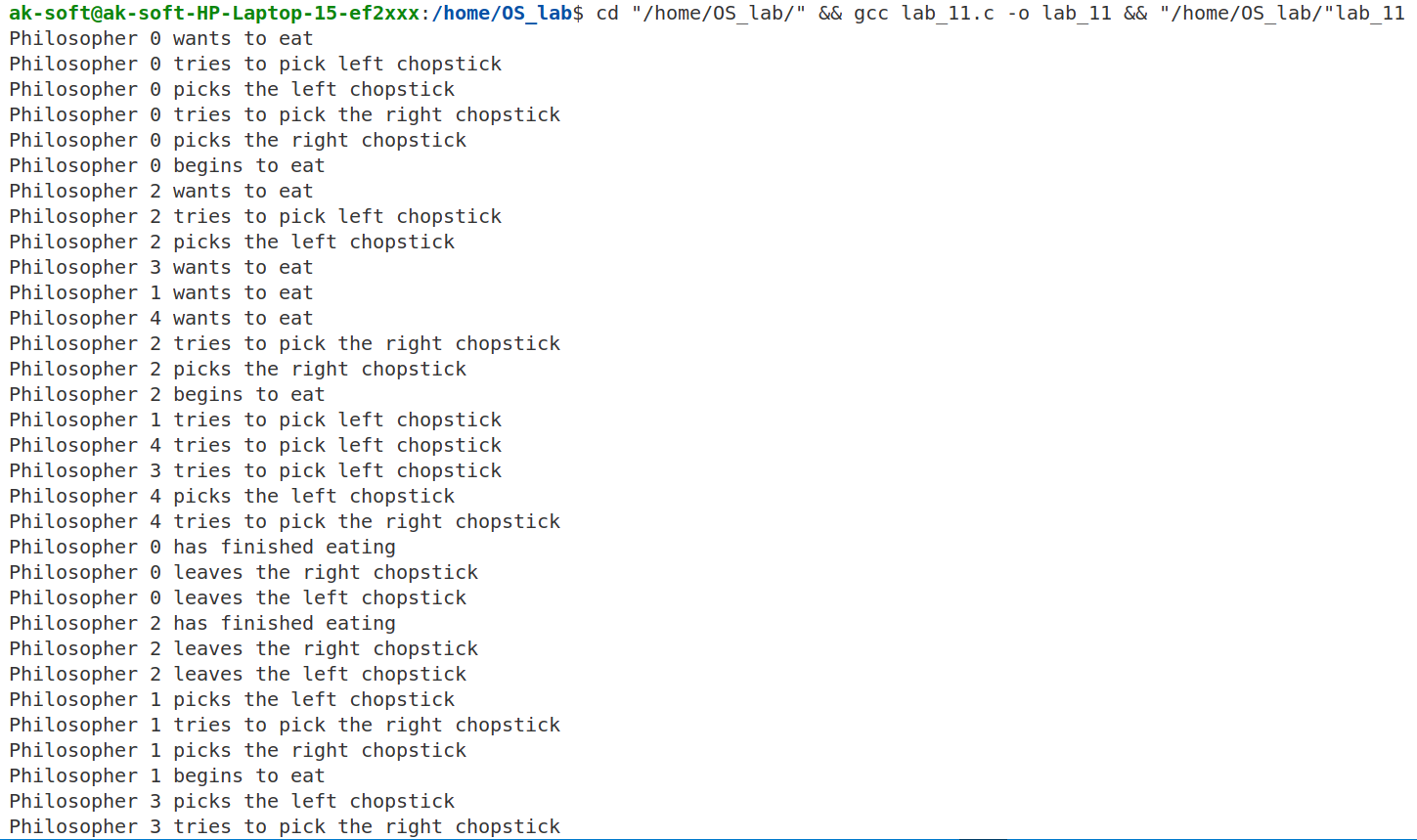
lab 10



lab 12

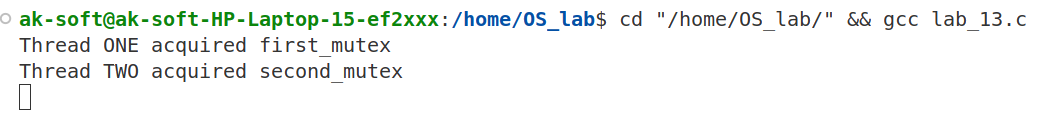


lab 11

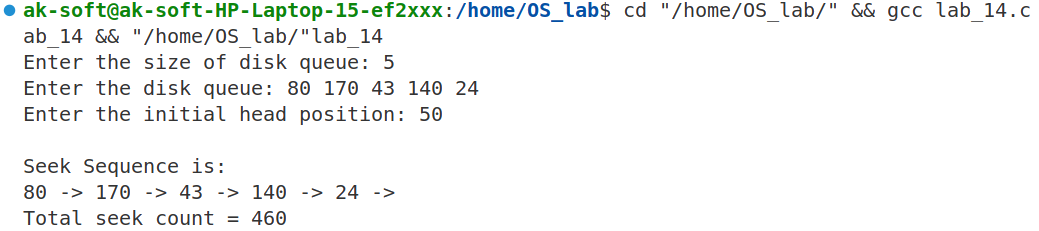




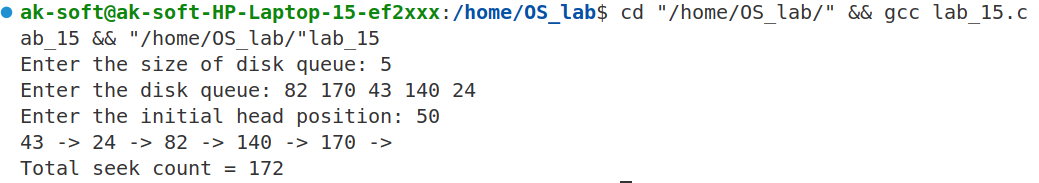
lab 13



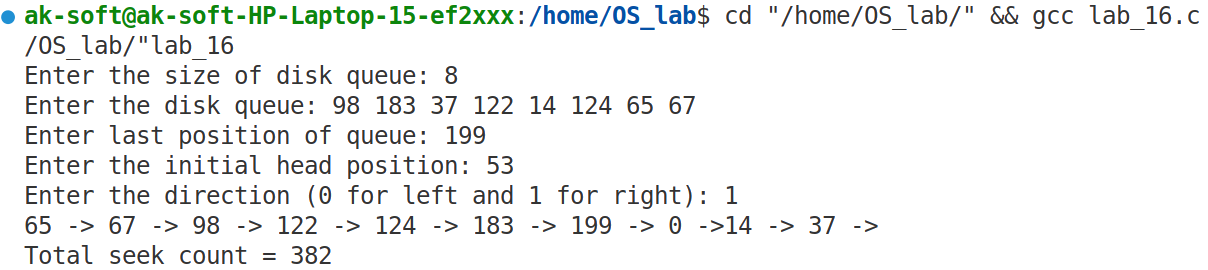
lab 14



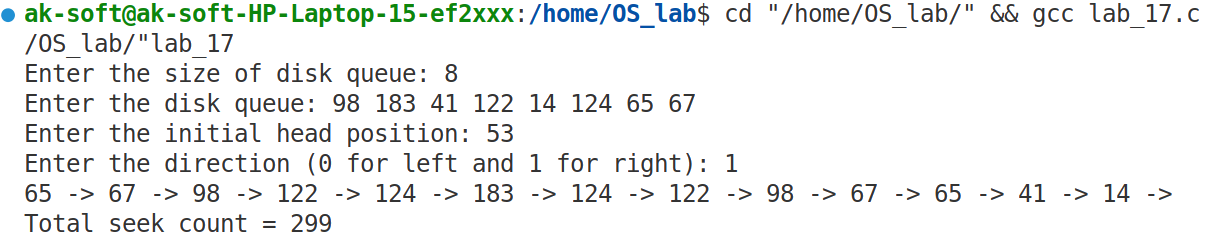
lab 15



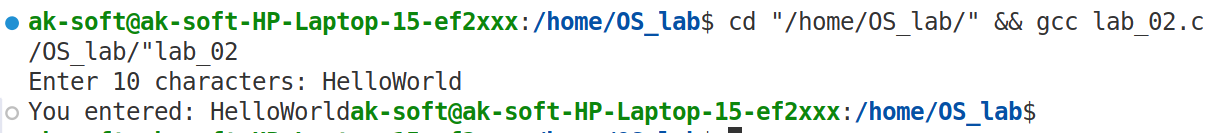
lab 16



lab 17

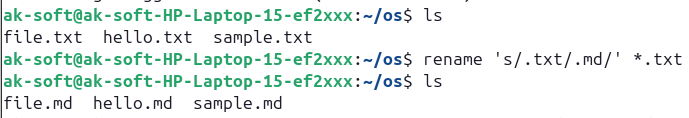


lab 02

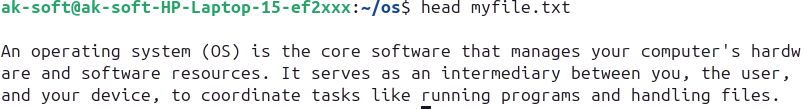


lab 01

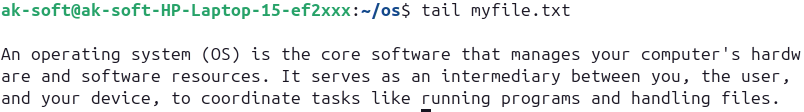
rename



head



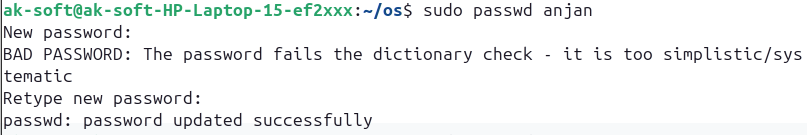
tail



useradd



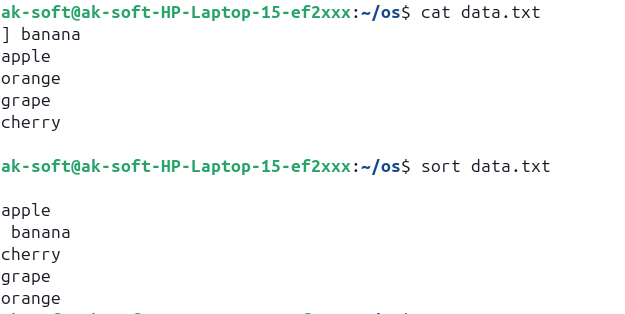
passwd



groupadd



sort



find



locate

date



|  |  |
| --- | --- |
| **Lab 1** | Execute the following commands in Ubuntu and obtain the suitable  output. For each command, describe its purpose and show the  corresponding output with your device name included.  1. rename  2. head  3. tail  4. useradd  5. passwd  6. groupadd  7. sort  8. find  9. locate  10. date |
| **Lab 2** | Write a C program using read() and write() system call that reads  10 bytes of input from the standard input device and stores them in  a buffer. |
| **Lab 3** | Write a C program that demonstrates process creation using the  fork() system call. The program should create a child process using  fork() and print the process IDs for both the parent and child  processes. |
| **Lab 4** | Write a C program to calculate the Average waiting time and turnaround time for a set of processes using the First-Come-First-Served (FCFS) scheduling algorithm. Your program should prompt the user to enter the number of processes and their respective burst times. |
| **Lab 5** | Write a C program to simulate the Round Robin scheduling algorithm for a set of processes. The program should prompt the user to enter the number of processes, burst times, and quantum times. calculate and print the average waiting time and average turnaround time for all processes. |
| **Lab 6** | Write a C program to calculate the Average waiting time and turnaround time for a set of processes using the Shortest Job First (SJF) scheduling algorithm. The program should prompt the user to enter the number of processes and their respective burst times. |
| **Lab 7** | Write a program using pthreads to illustrate the occurrence of a race  condition. |
| **Lab 8** | Write a program using pthreads to create a thread that runs concurrently with the main program. |
| **Lab 9** | Write a program that demonstrates the Producer-Consumer problem using semaphores for synchronization. |
| **Lab 10** | Write a program using pthreads to demonstrate the usage of mutex  locks for synchronization in a multi-threaded environment. |
| **Lab 11** | Write a C program to simulate the Dining Philosophers problem using semaphores for synchronization. Implement five philosophers represented by threads and five chopsticks represented by semaphores. |
| **Lab 12** | Write a C program using pthreads to demonstrate the usage of Binary semaphores for synchronization in a multi-threaded environment. |
| **Lab 13** | Write a C program using pthreads to demonstrate the potential of a  deadlock situation due to improper locking of mutexes. |
| **Lab 14** | Write a C program to implement the FCFS Disk Scheduling algorithm to calculate the total seek count. The program should prompt the user to enter the size of the disk queue, the disk queue itself, and the initial head position. |
| **Lab 15** | Write a C program to implement the Shortest Seek Time First (SSTF) Disk Scheduling algorithm to calculate the total seek count. The program should prompt the user to enter the size of the disk queue, the disk queue itself, and the initial head position. |
| **Lab 16** | Write a C program to implement the C-SCAN Disk Scheduling algorithm to calculate the total seek count. The program should prompt the user to enter the size of the disk queue, the disk queue itself, the last position of the queue, and the initial head position. |
| **Lab 17** | Write a C program to implement the LOOK Disk Scheduling algorithm to calculate the total seek count. The program should prompt the user to enter the size of the disk queue, the disk queue itself, and the initial head position. |

**SWASTIK COLLEGE**

Tribhuvan University

Faculty of Humanities and Social Sciences

A red text on a black background

Description automatically generated

Bachelor of Computer Application

(BCA)

Course: Operating System (CACS251)

Semester: 4th

A Lab Report On:

Operating System

**Submitted to:**

Ayush Chaudhary

(Subject teacher)

**Submitted by:**

Name: Anjan khadka

Roll No.: 03

**Acknowledgement**

We would like to extend our heartfelt thanks to our teacher, Mr. Ayush Chaudhary, for his constant guidance and support in our study of Operating Systems. His thorough explanations, enthusiasm, and approachable teaching style have greatly enhanced our understanding of the subject. His dedication to making complex concepts clear and engaging has inspired us to deepen our knowledge and interest in this field. We are truly grateful for his efforts in fostering a positive and encouraging learning environment. We appreciate his patience in answering our questions and providing valuable insights.