Requirement

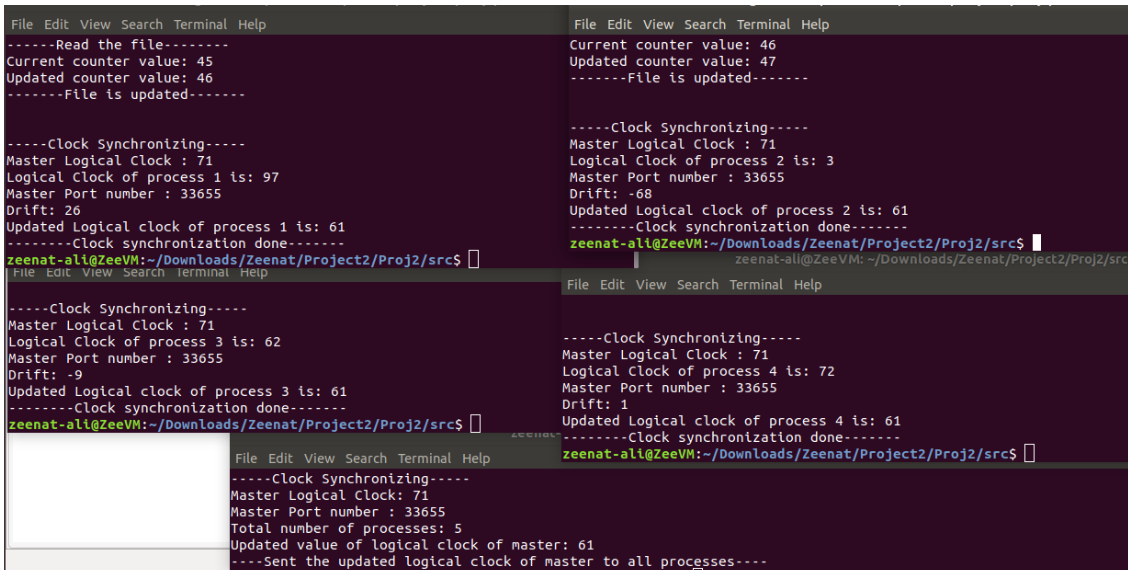
1. Implement Berkeley's algorithm to synchronize clocks.
2. Implement causal ordered multicasting for the distributed system and compare the result with non-causal multicast.
3. Implement distributed locking system to protect a shared file.

Program Design

The program has been developed in C++ on Ubuntu Linux.

1. <u>Clock Synchronization using Berkeley algorithm</u>:

The communication between server with multiple clients has been implemented using datagram sockets. The clocks of daemon(master) and slaves(clients) have been randomly initialized. The master clock, i.e. the server multicasts its time to all the slave processes, each process then calculates the difference between master's time and their own time and send the difference to the server. The master takes the difference and calculates the average of all slave clocks and itself and send the final updated values to all the slave processes and updates the same time for itself.



```
------Read the file--------
Current counter value: 45
Updated counter value: 46
-------File is updated-------


-----Clock Synchronizing-----
Master Logical Clock : 71
Logical Clock of process 1 is: 97
Master Port number : 33655
Drift: 26
Updated Logical clock of process 1 is: 61
--------Clock synchronization done-------
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$
```

```
Current counter value: 46
Updated counter value: 47
-------File is updated-------


-----Clock Synchronizing-----
Master Logical Clock : 71
Logical Clock of process 2 is: 3
Master Port number : 33655
Drift: -68
Updated Logical clock of process 2 is: 61
--------Clock synchronization done-------
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$
```

```
-----Clock Synchronizing-----
Master Logical Clock : 71
Logical Clock of process 3 is: 62
Master Port number : 33655
Drift: -9
Updated Logical clock of process 3 is: 61
--------Clock synchronization done-------
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$
```

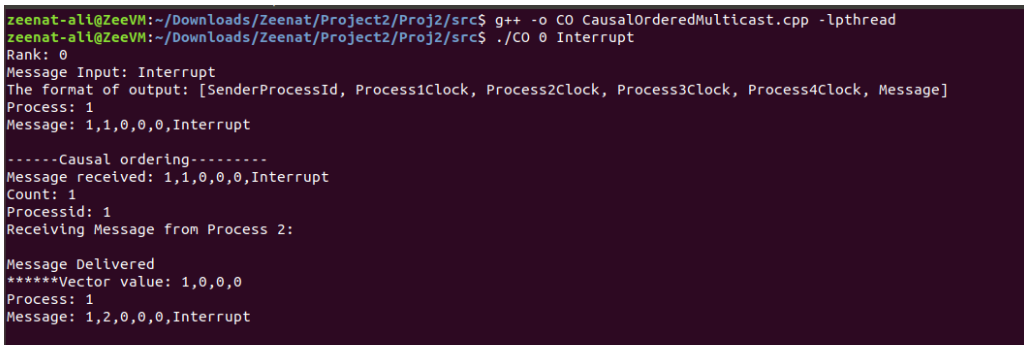
```
-----Clock Synchronizing-----
Master Logical Clock : 71
Logical Clock of process 4 is: 72
Master Port number : 33655
Drift: 1
Updated Logical clock of process 4 is: 61
--------Clock synchronization done-------
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$
```

```
-----Clock Synchronizing-----
Master Logical Clock: 71
Master Port number : 33655
Total number of processes: 5
Updated value of logical clock of master: 61
----Sent the updated logical clock of master to all processes----
```

2. <u>Causally vs non-causally ordered multicasting</u>:

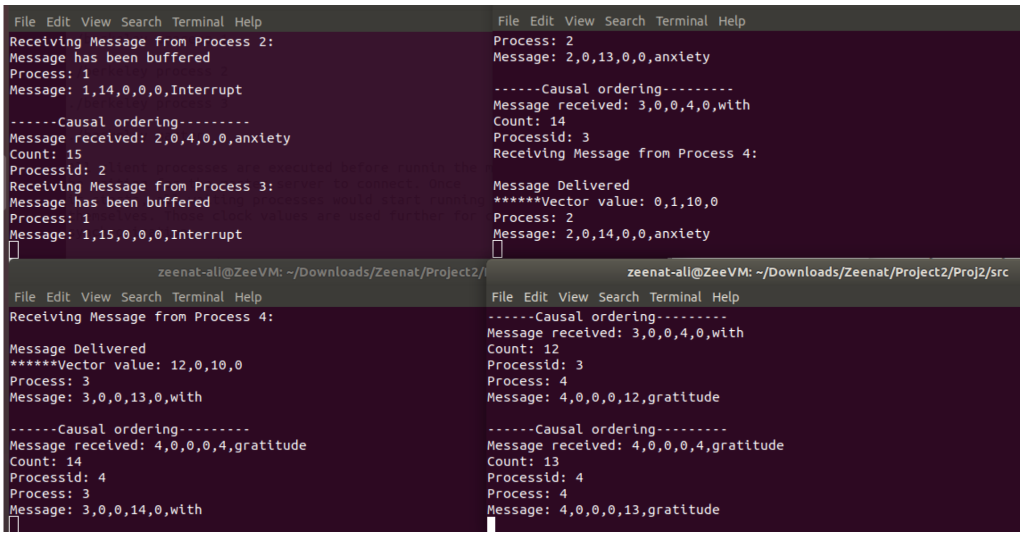
The communication between server with multiple clients has been implemented using datagram sockets. The program takes in 2 parameters, viz. the process id and the message to multicast and is implemented for up to 4 processes. The multicasting has been done in two ways – with causal order and without causal order. In causal order multicast, two threads, one each for receive and one for send operation has been implemented. When a process sends a message to other processes, it attaches its local clock in the form of no. of events happened on that process as a vector value. When this message is received at another process, the vector value is being compared with its existing vector value and if the received vector is less than or equal to current value, only then the message is delivered, else it is buffered. On the contrary, for multicast without causal order, the last condition of comparison is removed, and the messages are delivered as received without performing the causality check. This leads to unordered delivery of messages violating causality.

Causal order multicast – message format:



```
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$ g++ -o CO CausalOrderedMulticast.cpp -lpthread
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$ ./CO 0 Interrupt
Rank: 0
Message Input: Interrupt
The format of output: [SenderProcessId, Process1Clock, Process2Clock, Process3Clock, Process4Clock, Message]
Process: 1
Message: 1,1,0,0,0,Interrupt

------Causal ordering---------
Message received: 1,1,0,0,0,Interrupt
Count: 1
Processid: 1
Receiving Message from Process 2:

Message Delivered
******Vector value: 1,0,0,0
Process: 1
Message: 1,2,0,0,0,Interrupt
```

Causal order multicast – messages buffered to maintain causality:



```
Receiving Message from Process 2:
Message has been buffered
Process: 1
Message: 1,14,0,0,0,Interrupt

------Causal ordering---------
Message received: 2,0,4,0,0,anxiety
Count: 15
Processid: 2
Receiving Message from Process 3:
Message has been buffered
Process: 1
Message: 1,15,0,0,0,Interrupt
```

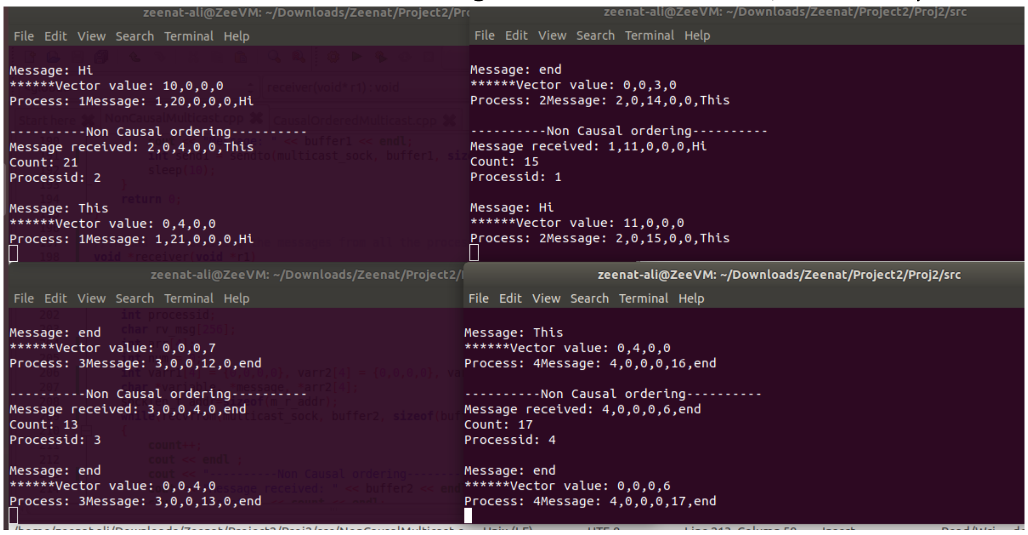
```
Process: 2
Message: 2,0,13,0,0,anxiety

------Causal ordering---------
Message received: 3,0,0,4,0,with
Count: 14
Processid: 3
Receiving Message from Process 4:

Message Delivered
******Vector value: 0,1,10,0
Process: 2
Message: 2,0,14,0,0,anxiety
```

```
Receiving Message from Process 4:

Message Delivered
******Vector value: 12,0,10,0
Process: 3
Message: 3,0,0,13,0,with

------Causal ordering---------
Message received: 4,0,0,0,4,gratitude
Count: 14
Processid: 4
Process: 3
Message: 3,0,0,14,0,with
```

```
------Causal ordering---------
Message received: 3,0,0,4,0,with
Count: 12
Processid: 3
Process: 4
Message: 4,0,0,0,12,gratitude

------Causal ordering---------
Message received: 4,0,0,0,4,gratitude
Count: 13
Processid: 4
Process: 4
Message: 4,0,0,0,13,gratitude
```

zeenat-ali@ZeeVM: ~/Downloads/Zeenat/Project2/Proj2/src
File Edit View Search Terminal Help

Multicast without causal order – messages delivered as received, no causality maintained:



```
Message: Hi
******Vector value: 10,0,0,0
Process: 1Message: 1,20,0,0,0,Hi

---------Non Causal ordering---------
Message received: 2,0,4,0,0,This
Count: 21
Processid: 2

Message: This
******Vector value: 0,4,0,0
Process: 1Message: 1,21,0,0,0,Hi
```

```
Message: end
******Vector value: 0,0,3,0
Process: 2Message: 2,0,14,0,0,This

---------Non Causal ordering---------
Message received: 1,11,0,0,0,Hi
Count: 15
Processid: 1

Message: Hi
******Vector value: 11,0,0,0
Process: 2Message: 2,0,15,0,0,This
```

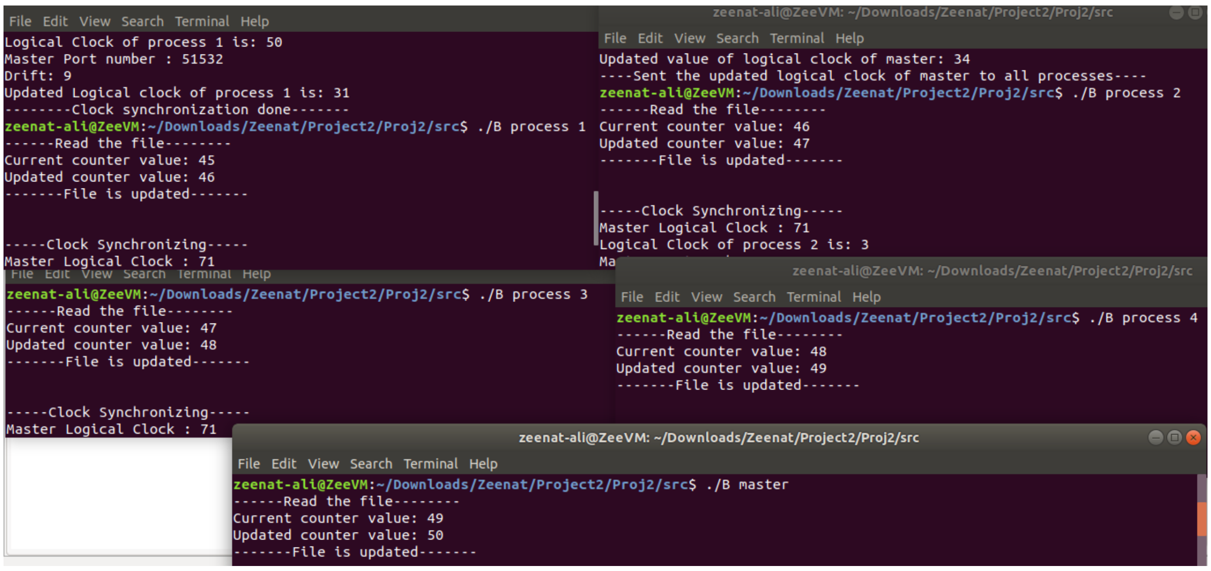
```
Message: end
******Vector value: 0,0,0,7
Process: 3Message: 3,0,0,12,0,end

---------Non Causal ordering---------
Message received: 3,0,0,4,0,end
Count: 13
Processid: 3

Message: end
******Vector value: 0,0,4,0
Process: 3Message: 3,0,0,13,0,end
```

```
Message: This
******Vector value: 0,4,0,0
Process: 4Message: 4,0,0,0,16,end

---------Non Causal ordering---------
Message received: 4,0,0,0,6,end
Count: 17
Processid: 4

Message: end
******Vector value: 0,0,0,6
Process: 4Message: 4,0,0,0,17,end
```

zeenat-ali@ZeeVM: ~/Downloads/Zeenat/Project2/Proj2/src
File Edit View Search Terminal Help

3. Distributed Locking system:

A feature of distributed locking is implemented along with Clock Synchronisation Program using distributed algorithm to protect a shared file that contains a counter value. Each process acquires a mutex lock, accesses the file, updates the counter and releases the mutex lock. Then the next process reads the counter value updated by previous process and update it further and so on. This maintains the mutual exclusion and the value of the counter is always correctly updated.


File Edit View Search Terminal Help
zeenat-ali@ZeeVM: ~/Downloads/Zeenat/Project2/Proj2/src

```
Logical Clock of process 1 is: 50
Master Port number : 51532
Drift: 9
Updated Logical clock of process 1 is: 31
--------Clock synchronization done-------
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$ ./B process 1
------Read the file--------
Current counter value: 45
Updated counter value: 46
-------File is updated------


-----Clock Synchronizing-----
Master Logical Clock : 71
```

```
Updated value of logical clock of master: 34
----Sent the updated logical clock of master to all processes----
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$ ./B process 2
------Read the file--------
Current counter value: 46
Updated counter value: 47
-------File is updated------


-----Clock Synchronizing-----
Master Logical Clock : 71
Logical Clock of process 2 is: 3
Ma
```

```
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$ ./B process 3
------Read the file--------
Current counter value: 47
Updated counter value: 48
-------File is updated------


-----Clock Synchronizing-----
Master Logical Clock : 71
```

```
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$ ./B process 4
------Read the file--------
Current counter value: 48
Updated counter value: 49
-------File is updated------
```

```
zeenat-ali@ZeeVM:~/Downloads/Zeenat/Project2/Proj2/src$ ./B master
------Read the file--------
Current counter value: 49
Updated counter value: 50
-------File is updated------
```

Learnings

1. The clock synchronization program helped me learn implement the sockets using UDP.
2. I also learned how various processes can be synchronized using their local clocks.
3. The multicast program helped me get a clearer understanding of difference between causal and on-causal ordered multicast and what problems can be raised due to unordered message transfer. The words of my sentence were not in sequence during multicast without causal ordering and so the sentence made no sense. This can be applied on larger scale where messages are not sent in sequence and cause violation of causality.
4. I also learned how using distributed algorithm, we can implement mutual exclusion among processes who are sharing the same file.

Issues/ Challenges

1. The major issue I faced during clock synchronization implementation was to keep the processes wait until the daemon process calculates the average time and send to every process. I achieved this by connecting the clients first, so that when daemon clock sends out its time, it already has clients waiting for it.
2. The other challenge was to maintain the vector values of each process and comparing them with previous vector elements. I solved this issue using vector arrays.