

Project-2

1. Berkley

In The First Assignment, I implemented the Berkley Algorithm to synchronize the clocks of the processes.

Here I took the Server as Daemon process and client processes where they would be synchronized with the server.

Sockets have been used to communicate between processes. For each process, a random clock value is generated.

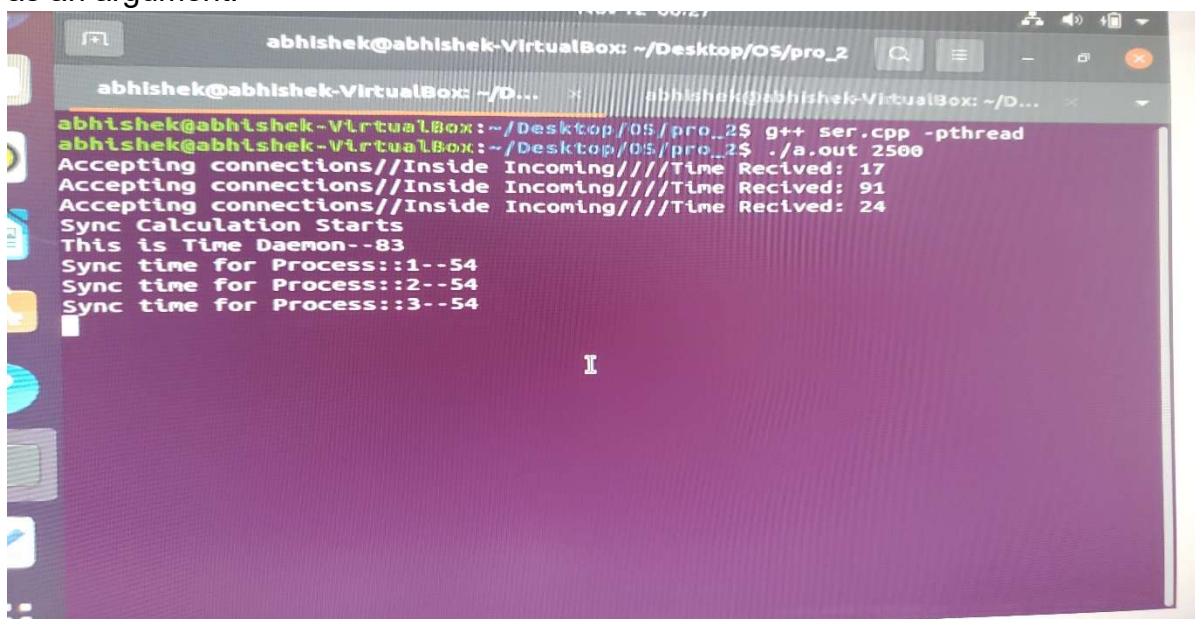
The Daemon server broadcasts its clock value to all clients, and each client responds with an offset value. After that, the Daemon server calculates the average, updates its own clock, and broadcasts the revised number to all processes once more.

As a result, clock synchronization between all processes is established. Here I wrote a python script with the number of processes to be spawn as an argument. This would generate multiple client processes which would be communicated to the server.

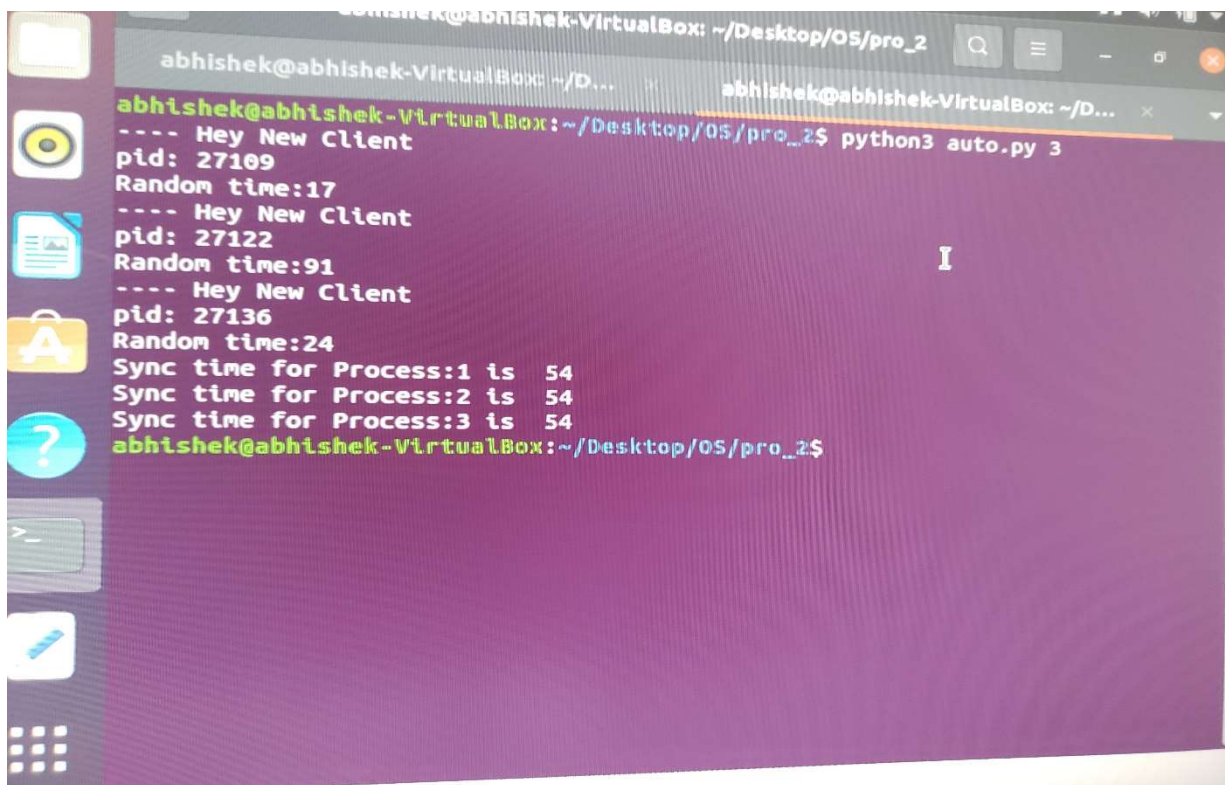
I am also updating the number of processes in a text file and a counter for each process being spawn.

Once all the processes are ready clock synchronization process starts. The updated value is sent to all the processes to synchronize with the Daemon clock.

Below is the snap of the resulted implementations. Here 3 client processes are spawn, and a server is present as Daemon. Here 3 is given as an argument to the python script to spawn the processes. In the server script we have given port number as an argument.



```
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2$ g++ ser.cpp -pthread
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2$ ./a.out 2500
Accepting connections//Inside Incoming///Time Recived: 17
Accepting connections//Inside Incoming///Time Recived: 91
Accepting connections//Inside Incoming///Time Recived: 24
Sync Calculation Starts
This is Time Daemon--83
Sync time for Process::1--54
Sync time for Process::2--54
Sync time for Process::3--54
```

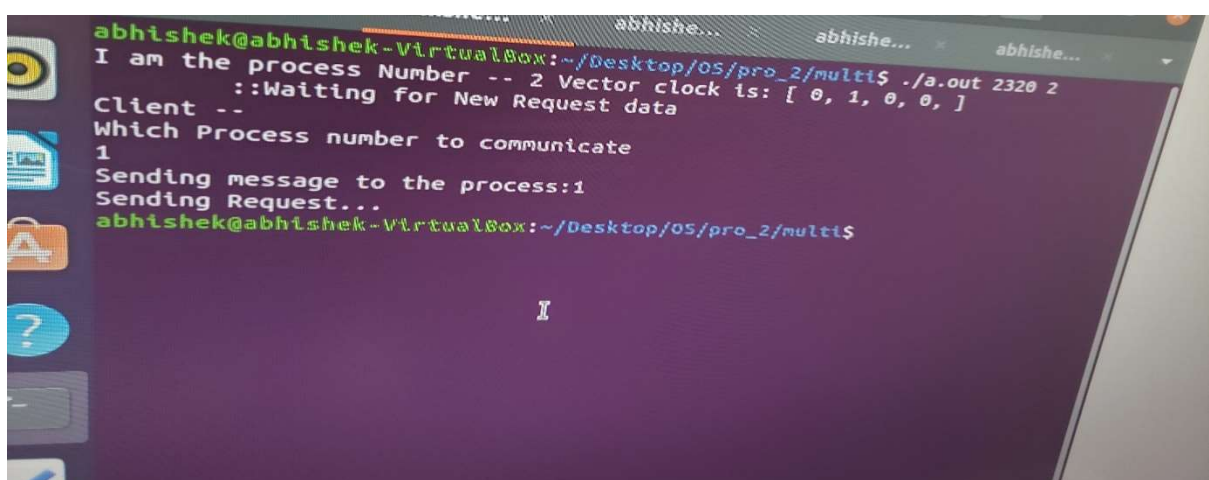


```
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2
abhishek@abhishek-VirtualBox: ~/D...
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2$ python3 auto.py 3
---- Hey New Client
pid: 27109
Random time:17
---- Hey New Client
pid: 27122
Random time:91
---- Hey New Client
pid: 27136
Random time:24
Sync time for Process:1 is 54
Sync time for Process:2 is 54
Sync time for Process:3 is 54
abhishek@abhishek-VirtualBox:~/Desktop/OS/pro_2$
```

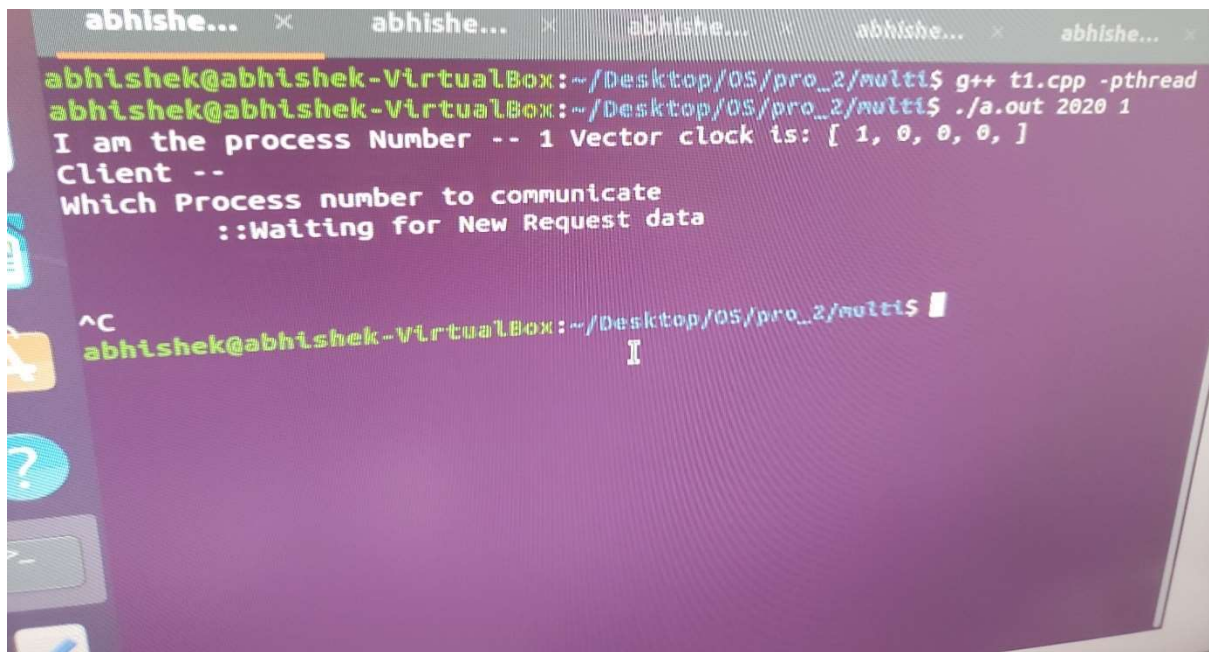
2. Multicasting

For transmitting and receiving messages to and from all processes, two threads have been formed. When a process delivers a message, it also sends a vector value that is associated with the message.

During the receiving process, the values in the vector are compared to the respective values in each process. These Updated values are the sent to the server.



```
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/multi$ ./a.out 2320 2
I am the process Number -- 2 Vector clock is: [ 0, 1, 0, 0, ]
Client --
Which Process number to communicate
1
Sending message to the process:1
Sending Request...
abhishek@abhishek-VirtualBox:~/Desktop/OS/pro_2/multi$
```



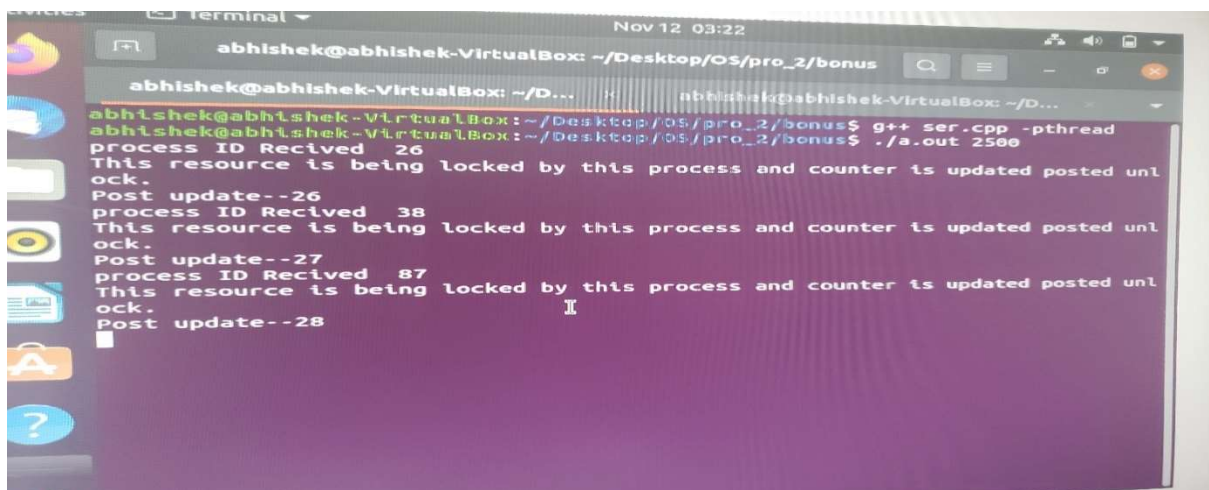
```
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/multi$ g++ t1.cpp -pthread
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/multi$ ./a.out 2020 1
I am the process Number -- 1 Vector clock is: [ 1, 0, 0, 0, ]
Client --
Which Process number to communicate
::Waiting for New Request data

^C
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/multi$
```

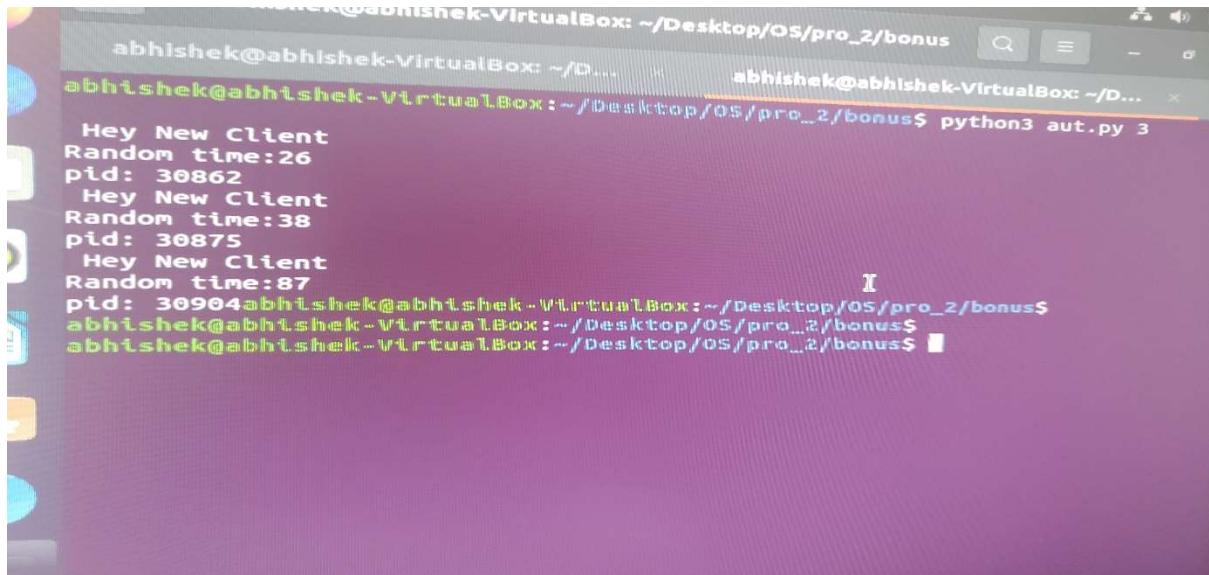
From the above snap we can see that first vector, where its counter is incremented so it comes as [1,0,0,0] and for the second process the vector clock becomes [0,1,0,0] where its local counter is updated. When process 2 sends to process 1 then the received vector is [1,1,0,0]

3. Distributed Locking

By using Multiple Clients and a server, where clients request for a resource concurrently, so this is protected by mutex. I used mutex on a shared file to implement distributed locking. All processes attempt to connect, resulting in a mutex lock on the file. Each process must read and update a value that is stored in the file. The mutex is unlocked only after they have read and updated the value in the file. Here I increment the value in the file after the process has performed its operations that is usage of the resource.



```
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/bonus$ g++ ser.cpp -pthread
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/bonus$ ./a.out 2500
process ID Recived 26
This resource is being locked by this process and counter is updated posted unl
ock.
Post update--26
process ID Recived 38
This resource is being locked by this process and counter is updated posted unl
ock.
Post update--27
process ID Recived 87
This resource is being locked by this process and counter is updated posted unl
ock.
Post update--28
```



```
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/bonus
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/bonus$ python3 aut.py 3
Hey New Client
Random time:26
pid: 30862
Hey New Client
Random time:38
pid: 30875
Hey New Client
Random time:87
pid: 30904
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/bonus$
abhishek@abhishek-VirtualBox: ~/Desktop/OS/pro_2/bonus$
```

From the above Snap we can observe the implementation results as server would be started as shown and in the python script processes are spawn and the argument is given as number of processes. Here we have taken the number of processes as 3, so 3 clients are spawn with their processes ID as shown. In the Server side we can see that the processes ID are received, and the resource is being used and post that it is updating in the file. The updated value is read from the file.