

## **Practical 2**

### **Distributed Operating System**

**Name - Ritesh Parkhi**

**Roll No. – 46**

**Batch – 3**

#### **AIM:**

Construct a program to demonstrate the concept of logical clock synchronization in a distributed environment using Lamport Logical clock.

#### **THEORY:**

A **Lamport logical clock** is an incrementing counter maintained in each process. Conceptually, this logical clock can be thought of as a clock that only has meaning in relation to messages moving between processes. When a process receives a message, it resynchronizes its logical clock with that sender (causality).

The algorithm of Lamport Timestamps can be captured in a few rules:

- All the process counters start with value 0.
- A process increments its counter for each event (internal event, message sending, message receiving) in that process.
- When a process sends a message, it includes its (incremented) counter value with the message.
- On receiving a message, the counter of the recipient is updated to the greater of its current counter and the timestamp in the received message, and then incremented by one.

Looking at these rules, we can see the algorithm will create a minimum overhead, since the counter consists of just one integer value and the messaging piggybacks on inter-process messages.

One of the shortcomings of Lamport Timestamps is rooted in the fact that they only partially order events (as opposed to total order). Partial order indicates that not every pair of events need be comparable. If two events can't be compared, we call these events concurrent. The problem with Lamport Timestamps is that they can't tell if events are concurrent or not. This problem is solved by Vector Clocks.

## PROGRAM:

```
// C program to illustrate the Lamport's
#include <stdio.h>

// Function to find the maximum timestamp
// between 2 events
int max1(int a, int b)
{
    // Return the greatest of th two
    if (a > b)
        return a;
    else
        return b;
}

// Function to display the Logical timestamp
void display(int e1, int e2,
            int p1[5], int p2[3])
{
    int i;

    printf("\nThe time stamps of "
           "events in P1:\n");

    for (i = 0; i < e1; i++) {
        printf("%d ", p1[i]);
    }

    printf("\nThe time stamps of "
           "events in P2:\n");

    // Print the array p2[]
    for (i = 0; i < e2; i++)
        printf("%d ", p2[i]);
}

// Function to find the timestamp of events
void lamportLogicalClock(int e1, int e2,
```

```

        int m[5][3])
{
    int i, j, k, p1[e1], p2[e2];

    // Initialize p1[] and p2[]
    for (i = 0; i < e1; i++)
        p1[i] = i + 1;

    for (i = 0; i < e2; i++)
        p2[i] = i + 1;

    for (i = 0; i < e2; i++)
        printf("\te2%d", i + 1);

    for (i = 0; i < e1; i++) {

        printf("\n e1%d \t", i + 1);

        for (j = 0; j < e2; j++)
            printf("%d\t", m[i][j]);
    }

    for (i = 0; i < e1; i++) {
        for (j = 0; j < e2; j++) {

            // Change the timestamp if the
            // message is sent
            if (m[i][j] == 1) {
                p2[j] = max1(p2[j], p1[i] + 1);
                for (k = j + 1; k < e2; k++)
                    p2[k] = p2[k - 1] + 1;
            }

            // Change the timestamp if the
            // message is received
            if (m[i][j] == -1) {
                p1[i] = max1(p1[i], p2[j] + 1);
                for (k = i + 1; k < e1; k++)
                    p1[k] = p1[k - 1] + 1;
            }
        }
    }

    // Function Call
    display(e1, e2, p1, p2);
}

// Driver Code

```

```

int main()
{
    int e1 = 5, e2 = 3, m[5][3];

    // message is sent and received
    // between two process

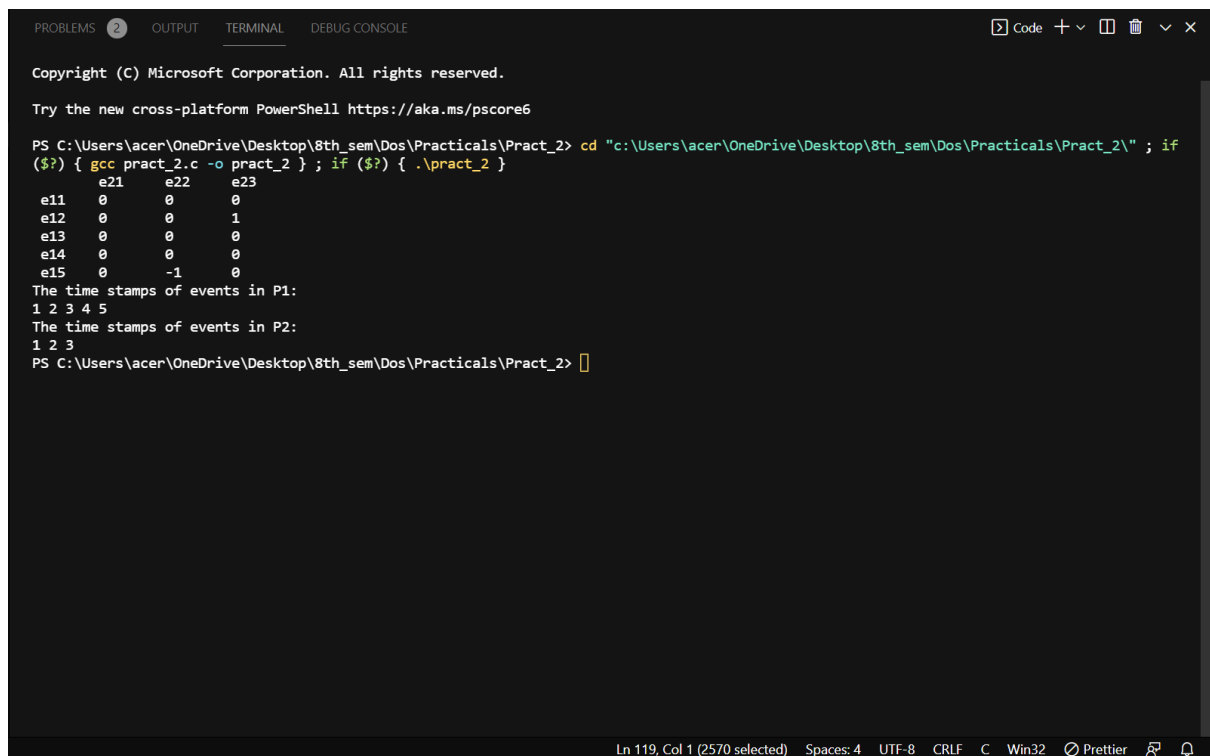
    /*dep[i][j] = 1, if message is sent
                from ei to ej
    dep[i][j] = -1, if message is received
                by ei from ej
    dep[i][j] = 0, otherwise*/
    m[0][0] = 0;
    m[0][1] = 0;
    m[0][2] = 0;
    m[1][0] = 0;
    m[1][1] = 0;
    m[1][2] = 1;
    m[2][0] = 0;
    m[2][1] = 0;
    m[2][2] = 0;
    m[3][0] = 0;
    m[3][1] = 0;
    m[3][2] = 0;
    m[4][0] = 0;
    m[4][1] = -1;
    m[4][2] = 0;

    // Function Call
    lamportLogicalClock(e1, e2, m);

    return 0;
}

```

## OUTPUT:



```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\acer\OneDrive\Desktop\8th_sem\Dos\Practicals\Pract_2> cd "c:\Users\acer\OneDrive\Desktop\8th_sem\Dos\Practicals\Pract_2\" ; if ($?) { gcc pract_2.c -o pract_2 } ; if ($?) { .\pract_2 }
    e21      e22      e23
e11      0      0      0
e12      0      0      1
e13      0      0      0
e14      0      0      0
e15      0     -1      0
The time stamps of events in P1:
1 2 3 4 5
The time stamps of events in P2:
1 2 3
PS C:\Users\acer\OneDrive\Desktop\8th_sem\Dos\Practicals\Pract_2> 
```

## CONCLUSION:

Hence we successfully studied about Lamport Logical Clock commands.