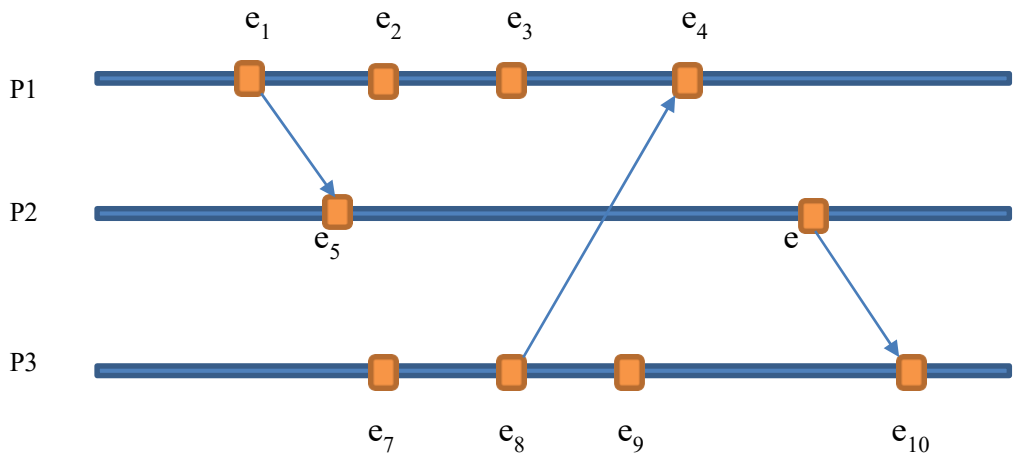
BITS, Pilani Hyderabad Campus
Advanced Operating Systems (CS G623)
1st Sem 2025-26
Lab 6: Vector Clock Simulation (Fidge-Mattern's)

In this lab, you will be simulating Vector clocks (Fidge Mattern's) using multiple threads, similar to the Lamport's clock simulation by passing messages (with vector timestamps) from one thread to another, using FIFO.



(Fig.1: The Time-Space Diagram for the 3 processes you will be simulating)

P1, P2 and P3 are represented using 3 threads, each having their individual vector timestamps (TS). Messages are passed between Pi and Pj, and they carry vector timestamp of Pi to Pj, e.g. the message passed from e1 to e5 will contain the timestamp of P1 at e1.

Our objective is to view the value of each timestamp for P1, P2 and P3 at occurrence of every event.
Time stamps are represented as array clock_val[][], initially all its values are 0.

clock_val[0][] represents the vector for Process P1.
clock_val[1][] represents the vector for Process P2.
clock_val[2][] represents the vector for Process P3.

P1, P2 and P3 and their respective events are simulated in individual functions where events are ***printf*** statements.

The time stamp messages are passed using FIFOs. Each Pi has its individual FIFOi

```
/* vectors.c*/ #in-
clude<stdio.h> #in-
clude<stdlib.h> #in-
clude<pthread.h>
#define NO_OF_THREADS 3

int clock_val[NO_OF_THREADS][NO_OF_THREADS]={{0,0,0},{0,0,0},{0,0,0}};
// the vector clocks for the 3 threads (simulating 3 Processes)
char fifo1[]="FIFO1"; // FIFO used by thread 1 char
fifo2[]="FIFO2"; // FIFO used by thread 2 char
fifo3[]="FIFO3"; // FIFO used by thread 3
```

```
void *func1(void* aa) // This function simulates all events of Process P1
 {
        int a,b,c;
        int i;
 // e1 ========================== Event e1 in process P1
        clock_val[0][0]++;
        FILE*f=fopen(fifo2,"w");// sending the vector clock to P2
        fprintf(f,"%d %d %d",clock_val[0][0],clock_val[0][1],clock_val[0][2]);
        fclose(f);
        printf("Val of clock e1 :-\n");
        for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[0][i]);
        printf("\n");
 // e2 ========================== Event e2 in process P1
        clock_val[0][0]++;
        printf("Val of clock e2 :-\n");
        for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[0][i]);
        for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[0][i]);
        printf("\n");
 // e3 ========================== Event e3 in process P1
        clock_val[0][0]++;
        printf("Val of clock e3 :-\n");
        for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[0][i]);
        printf("\n");
 // e4 ========================== Event e4 in process P1
        clock_val[0][0]++;
        FILE* f1=fopen(fifo1,"r");
        fscanf(f1,"%d %d %d",&a,&b,&c); // reading the message sent by P3
        fclose(f1);
        if(clock_val[0][0]<a)
                clock_val[0][0]=a; // checking the maximum
        if(clock_val[0][1]<b)
                clock_val[0][1]=b; // checking the maximum
        if(clock_val[0][2]<c)
                clock_val[0][2]=c; // checking the maximum
        printf("Val of clock e4 :-\n");
        for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[0][i]);
        printf("\n"); //return NULL;
 }
 void *func2(void* aa) // This function simulates all events of Process P2
 {
        int a,b,c,i;
        // e5 ========================== Event e5 in process P2
        clock_val[1][1]++;
        FILE*f=fopen(fifo2,"r");
        fscanf(f,"%d %d %d",&a,&b,&c); // reading the message sent by P1
        fclose(f);
        if(clock_val[1][0]<a)
                clock_val[1][0]=a; // checking the maximum
        if(clock_val[1][1]<b)
                clock_val[1][1]=b; // checking the maximum
        if(clock_val[1][2]<c)
                clock_val[1][2]=c; // checking the maximum
        printf("Val of clock e5 :-\n");
```

```
                for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[1][i]);
        printf("\n");
        // e6 ============================ Event e6 in process P2
        clock_val[1][1]++;
        FILE*f1=fopen(fifo3,"w");// sending the vector clock to P3 fprintf(f1,"%d %d
        %d",clock_val[1][0],clock_val[1][1],clock_val[1][2]); fclose(f1);
        printf("Val of clock e6 :-\n");
        for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[1][i]);
        printf("\n");
        return NULL;
}
void *func3(void* aa) // This function simulates all events of Process P3s
{
        int a,b,c,i;
        // e7 ============================ Event e7 in process P3
        clock_val[2][2]++;
        printf("Val of clock e7 :-\n");
        for(i=0;i<NO_OF_THREADS;i++)
        printf("%d\t",clock_val[2][i]);
        printf("\n");
        // e8 ============================ Event e8 in process P3
        clock_val[2][2]++;
        FILE*f=fopen(fifo1,"w");// sending the vector clock to P1
        fprintf(f,"%d %d %d",clock_val[2][0],clock_val[2][1],clock_val[2][2]);
        fclose(f);
        printf("Val of clock e8 :-\n");
        for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[2][i]);
        printf("\n");
        // e9 ============================ Event e9 in process P3
        clock_val[2][2]++;
        printf("Val of clock e9 :-\n");

        for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[2][i]);
        printf("\n");

        clock_val[2][2]++;
// e10 ============================ Event e10 in process P3
        clock_val[2][2]++;
        FILE* f1=fopen(fifo3,"r");
        fscanf(f1,"%d %d %d",&a,&b,&c); // reading the message sent by P2
        fclose(f1);
        if(clock_val[2][0]<a)
                clock_val[2][0]=a; // checking the maximum
        if(clock_val[2][1]<b)
                clock_val[2][1]=b; // checking the maximum
        if(clock_val[2][2]<c)
                clock_val[2][2]=c; // checking the maximum
        printf("Val of clock e10 :-\n");
        for(i=0;i<NO_OF_THREADS;i++)
                printf("%d\t",clock_val[2][i]);
        printf("\n");
        return NULL;
}
```

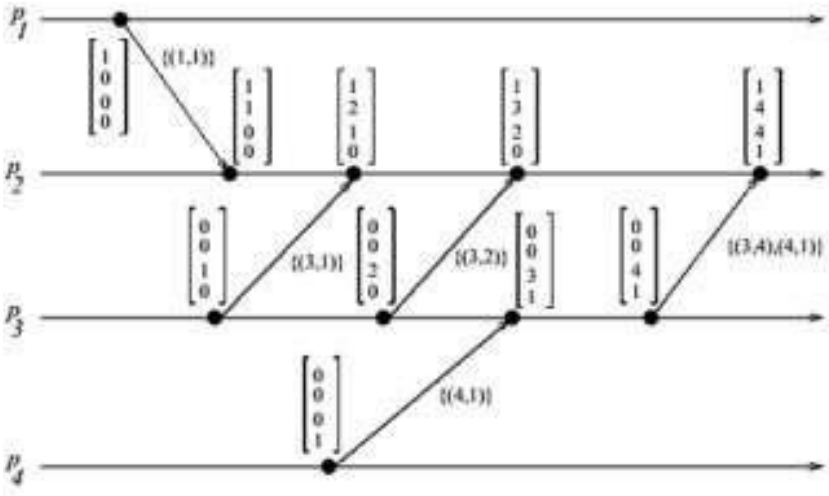
```
int main()
{
        pthread_t thr[NO_OF_THREADS]; // the threads representing P1, P2 and P3 int i;

        if(pthread_create(&thr[0],NULL,func1,NULL))
        {
                printf("Thread Error\n");
                exit(1);
        }
        if(pthread_create(&thr[1],NULL,func2,NULL))
        {
                printf("Thread Error\n");
                exit(1);
        }
        if(pthread_create(&thr[2],NULL,func3,NULL))
        {
                printf("Thread Error\n");
                exit(1);
        }
        for(i=0;i<NO_OF_THREADS;i++)
        {

        }
        exit(0); pthread_join(thr[i],NULL)
}
```

Task: Implement an optimized version of vector clocks proposed by Singhal-Kshemkalyani as shown in the figure below. (Fig.2)

Output:



(Fig.2)