**Experiment No: 04**

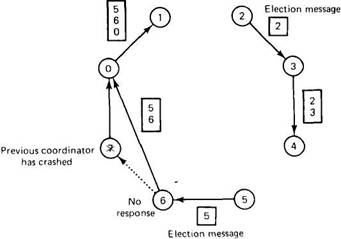
**Aim: To implement Election algorithm using Ring algorithm.**

**Theory:**

**Ring Algorithm:**

Another election algorithm is based on the use of a ring, but without a token. We assume that the processes are physically or logically ordered, so that each process knows who its successor is. When any process notices that the coordinator is not functioning, it builds an ELECTION message containing its own process number and sends the message to its successor. If the successor is down, the sender skips over the successor and goes to the next member along the ring, or the one after that, until a running process is located. At each step, the sender adds its own process number to the list in the message.

Eventually, the message gets back to the process that started it all. That process recognizes this event when it receives an incoming message containing its own process number. At that point, the message type is changed to COORDINATOR and circulated once again, this time to inform everyone else who the coordinator is (the list member with the highest number) and who the members of the new ring are. When this message has circulated once, it is removed and everyone goes back to work.



**Fig. Election algorithm using a ring.**

In above figure, we see what happens if two processes, 2 and 5, discover simultaneously that the previous coordinator, process 7,has crashed. Each of these builds an ELECTION message and starts circulating it. Eventually, both messages will go all the way around, and both 2 and 5 will convert them into COORDINATOR messages, with exactly the same members and in the same order. When both have gone around again, both will be removed. It does no harm to have extra messages circulating; at most it wastes a little bandwidth.

**Program:**

#include<string.h>

#include<iostream.h>

#include<stdio.h>

#include<stdlib.h>

struct rr

{

int index;

int id;

int f;

char state[10];

}proc[10];

int i,j,k,m,n;

void main()

{

int temp;

char str[10];

cout<<"\n enter the number of process\t";

cin>>n;

for(i=0;i<n;i++)

{

proc[i].index;

cout<<"\n enter id of process\t";

cin>>proc[i].id;

strcpy(proc[i].state,"active");

proc[i].f=0;

}

// sorting

for(i=0;i<n-1;i++)

{

for(j=0;j<n-1;j++)

{

if(proc[j].id>proc[j+1].id)

{

temp=proc[j].id;

proc[j].id=proc[j+1].id;

proc[j+1].id=temp;

}

}

}

for(i=0;i<n;i++)

printf("[%d] %d\t",i,proc[i].id);

int init;

int ch;

int temp1;

int temp2;

int arr[10];

strcpy(proc[n-1].state,"inactive");

cout<<"\nprocess "<<proc[n-1].id<<" select as coordinator";

while(1)

{

cout<<"\n1)election 2)quit\n";

scanf("%d",&ch);

for(i=0;i<n;i++)

{

proc[i].f=0;

}

switch(ch)

{

case 1:

cout<<"\nenter the process Number who intialised election";

scanf("%d",&init);

temp2=init;

temp1=init+1;

i=0;

while(temp2!=temp1)

{

if(strcmp(proc[temp1].state,"active")==0 && proc[temp1].f==0 )

{

cout<<"process "<<proc[init].id<<"send message to "<<proc[temp1].id<<"\n";

proc[temp1].f=1;

init=temp1;

arr[i]=proc[temp1].id;

i++;

}

if(temp1==n)

temp1=0;

else

temp1++;

}

cout<<"process "<<proc[init].id<<"send message to "<<proc[temp1].id<<"\n";

arr[i]=proc[temp1].id;

i++;

int max=-1;

for(j=0;j<i;j++)

{

if(max<arr[j])

max=arr[j];

}

cout<<"\nprocess "<<max<<" select as coordinator";

for(i=0;i<n;i++)

{

if(proc[i].id==max)

{

strcpy(proc[i].state,"inactive");

// cout<<"\n"<<i<<" "<<proc[i].id<<"deactivate\n";

} }

break;

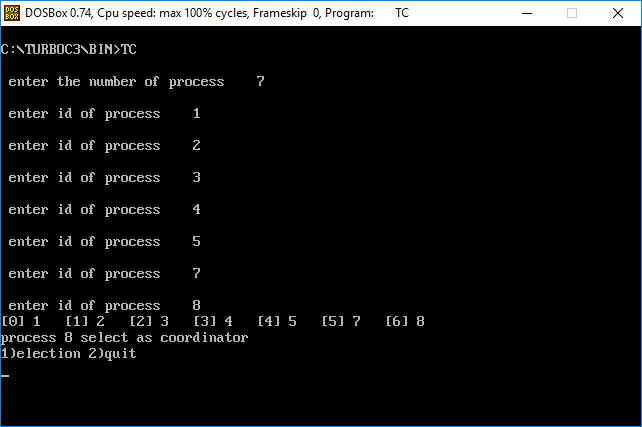
break;

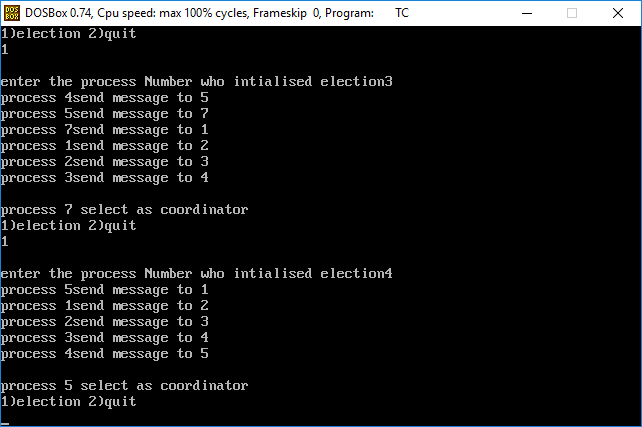
}

}

}

**Output:**





**Conclusion:**

Hence we have studied Election algorithm using Ring algorithm.