Experiment 5 **Surya Narayanan**

Dining Philosophers Problem

s5 csb 48

---

1. Algorithm

Step 1: START
Step 2: Include necessary header files
Step 3: Define fixed values such as, N, LEFT, RIGHT, EATING, HUNGRY, THINKING.
Step 4: Initialize semaphore required (s, mutex)
Step 5: Do the following for all 5 philosophers.
    Step 5.1 Initialize semaphore s with attributes.
Step 6: Do the following for all 5 philosophers
    Step 6.1 - Create philosopher process by creating condition variable
    Step 6.2 - Initialize it to default attributes
    Step 6.3: Display the philosopher is thinking.
Step 7: for each philosopher check for the completion of condition variable each time.
Step 8: Define functions philosopher(), take fork(), put fork(), and test();

Take fork()

Step 1: START
Step 2: Acquire the semaphore for that particular (mutex) philosopher.
Step 3: set the philosopher state as hungry. and display the same

Step 4. Call the function test to check if neighbouring philosphers are eating.

Step 5. Release up the semaphore mutex and wait for the semaphore S.

Step 6. Wait for the signal

Step 7. STOP.

Put form()

Step 1: START

Step 2: Acquire the semaphore (mutex) for the philosopher. Set the state of philosopher as thinking.

Step 3: Display the message with the forks that are left over the table (LEFT, RIGHT)

Step 4: Call the function test to check for the available forks.

Step 5: Release the semaphore.

Step 6: STOP

Philosopher()

Step 1: START

Step 2: Repeat the following for each philosopher

Step 2.1: call the function take fork() and wait for its signal after its state is eating.

Step 2.2: Call the function put fork() after the philosopher has eaten or if the forks are not available.

TEST()

Step 1: START

Step 2: If the current philosopher is hungry and the left and right forks are available;

Step 2.1: & Set the state of philosopher as eating

Step 2.2 Display the message and declare the semaphore S

Step 3: STOP.

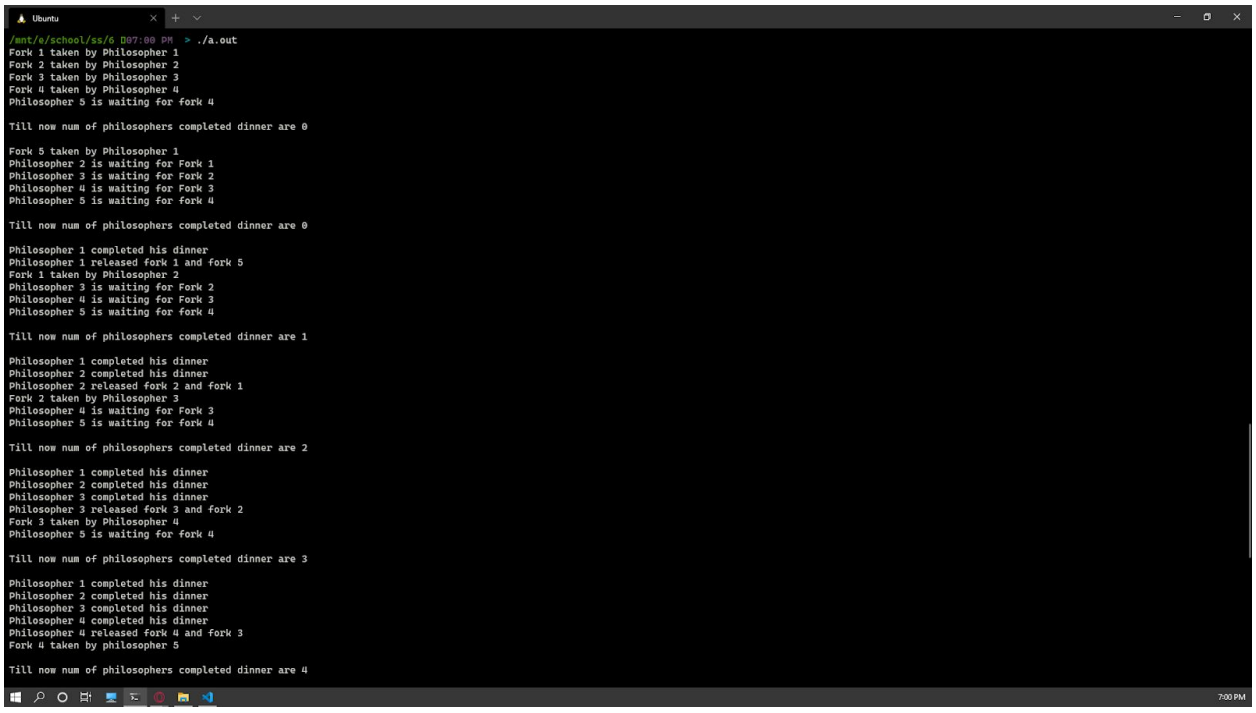
Program

```
#include <stdio.h>

#define n 5
int compltedPhilo = 0, i;
struct fork
{
    int taken;
} ForkAvil[n];
struct philosp
{
    int left;
    int right;
} Philostatus[n];

void goForDinner(int philID)
{
    if (Philostatus[philID].left == 10 && Philostatus[philID].right == 10)
        printf("Philosopher %d completed his dinner\n", philID + 1);
    else if (Philostatus[philID].left == 1 && Philostatus[philID].right ==
1)
    {
        printf("Philosopher %d completed his dinner\n", philID + 1);
        Philostatus[philID].left = Philostatus[philID].right = 10;
        int otherFork = philID - 1;
        if (otherFork == -1)
            otherFork = (n - 1);
        ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;
        printf("Philosopher %d released fork %d and fork %d\n", philID +
1, philID + 1, otherFork + 1);
        compltedPhilo++;
    }
    else if (Philostatus[philID].left == 1 && Philostatus[philID].right ==
0)
    {
        if (philID == (n - 1))
        {
            if (ForkAvil[philID].taken == 0)
```

```
            {
                ForkAvil[philID].taken = Philostatus[philID].right = 1;
                printf("Fork %d taken by philosopher %d\n", philID + 1,
philID + 1);
            }
            else
            {
                printf("Philosopher %d is waiting for fork %d\n", philID +
1, philID + 1);
            }
        }
        else
        {
            int dupphilID = philID;
            philID -= 1;
            if (philID == -1)
                philID = (n - 1);
            if (ForkAvil[philID].taken == 0)
            {
                ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;
                printf("Fork %d taken by Philosopher %d\n", philID + 1,
dupphilID + 1);
            }
            else
            {
                printf("Philosopher %d is waiting for Fork %d\n",
dupphilID + 1, philID + 1);
            }
        }
    }
    else if (Philostatus[philID].left == 0)
    {
        if (philID == (n - 1))
        {
            if (ForkAvil[philID - 1].taken == 0)
            {
                ForkAvil[philID - 1].taken = Philostatus[philID].left = 1;
                printf("Fork %d taken by philosopher %d\n", philID, philID
+ 1);
            }
```

```
            else
                printf("Philosopher %d is waiting for fork %d\n", philID +
1, philID);
        }
        else
        {
            if (ForkAvil[philID].taken == 0)
            {
                ForkAvil[philID].taken = Philostatus[philID].left = 1;
                printf("Fork %d taken by Philosopher %d\n", philID + 1,
philID + 1);
            }
            else
            {
                printf("Philosopher %d is waiting for Fork %d\n", philID +
1, philID + 1);
            }
        }
    }
}
int main()
{
    for (i = 0; i < n; i++)
        ForkAvil[i].taken = Philostatus[i].left = Philostatus[i].right =
0;
    while (compltedPhilo < n)
    {
        for (i = 0; i < n; i++)
            goForDinner(i);
        printf("\nTill now num of philosophers completed dinner are
%d\n\n", compltedPhilo);
    }
    return 0;
}
```

Output


Ubuntu
/mnt/e/school/ss/6 07:00 PM > ./a.out
Fork 1 taken by Philosopher 1
Fork 2 taken by Philosopher 2
Fork 3 taken by Philosopher 3
Fork 4 taken by Philosopher 4
Philosopher 5 is waiting for fork 4
Till now num of philosophers completed dinner are 0
Fork 5 taken by Philosopher 1
Philosopher 2 is waiting for Fork 1
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for Fork 3
Philosopher 5 is waiting for fork 4
Till now num of philosophers completed dinner are 0
Philosopher 1 completed his dinner
Philosopher 1 released fork 1 and fork 5
Fork 1 taken by Philosopher 2
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for Fork 3
Philosopher 5 is waiting for fork 4
Till now num of philosophers completed dinner are 1
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 2 released fork 2 and fork 1
Fork 2 taken by Philosopher 3
Philosopher 4 is waiting for Fork 3
Philosopher 5 is waiting for fork 4
Till now num of philosophers completed dinner are 2
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 3 released fork 3 and fork 2
Fork 3 taken by Philosopher 4
Philosopher 5 is waiting for fork 4
Till now num of philosophers completed dinner are 3
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 4 completed his dinner
Philosopher 4 released fork 4 and fork 3
Fork 4 taken by philosopher 5
Till now num of philosophers completed dinner are 4
7:00 PM