

## **OS LAB 06**

```
#include <stdio.h>

#define N 4

int completedPhilo = 0;
int i;

struct Fork {
    int taken;
} ForkAvailable[N];

struct Philosopher {
    int left;
    int right;
} PhilosopherStatus[N];

void goForDinner(int philID) {
    // Case 1: Philosopher has already completed dinner
    if (PhilosopherStatus[philID].left == 10 && PhilosopherStatus[philID].right == 10) {
        printf("Philosopher %d already completed his dinner\n", philID + 1); }
    // Case 2: Philosopher has both forks and completes dinner
    else if (PhilosopherStatus[philID].left == 1 && PhilosopherStatus[philID].right == 1) {
        printf("Philosopher %d completed his dinner\n", philID + 1);
        PhilosopherStatus[philID].left = PhilosopherStatus[philID].right = 10;

        int otherFork = philID - 1;
        if (otherFork == -1)
            otherFork = (N - 1);

        ForkAvailable[philID].taken = ForkAvailable[otherFork].taken = 0;
        printf("Philosopher %d released fork %d and fork %d\n", philID + 1, philID + 1,
otherFork + 1);

        completedPhilo++;
    }
    // Case 3: Has left fork, trying for right fork
    else if (PhilosopherStatus[philID].left == 1 && PhilosopherStatus[philID].right == 0) { if
        (philID == (N - 1)) {
            if (ForkAvailable[philID].taken == 0) {
```

```
ForkAvailable[phillID].taken = PhilosopherStatus[phillID].right = 1;
printf("Fork %d taken by philosopher %d\n", phillID + 1, phillID + 1); }
else {
    printf("Philosopher %d is waiting for fork %d\n", phillID + 1, phillID + 1); }
} else {
    int dupPhillID = phillID;
    phillID -= 1;
    if (phillID == -1)
        phillID = (N - 1);

    if (ForkAvailable[phillID].taken == 0) {
        ForkAvailable[phillID].taken = PhilosopherStatus[dupPhillID].right = 1;
        printf("Fork %d taken by philosopher %d\n", phillID + 1, dupPhillID + 1); } else {
        printf("Philosopher %d is waiting for fork %d\n", dupPhillID + 1, phillID + 1); }
    }
}
// Case 4: Has not taken any fork yet
else if (PhilosopherStatus[phillID].left == 0) {
    if (phillID == (N - 1)) {
        if (ForkAvailable[phillID - 1].taken == 0) {
            ForkAvailable[phillID - 1].taken = PhilosopherStatus[phillID].left = 1;
            printf("Fork %d taken by philosopher %d\n", phillID, phillID + 1);
        } else {
            printf("Philosopher %d is waiting for fork %d\n", phillID + 1, phillID); }
        } else {
            if (ForkAvailable[phillID].taken == 0) {
                ForkAvailable[phillID].taken = PhilosopherStatus[phillID].left = 1;
                printf("Fork %d taken by philosopher %d\n", phillID + 1, phillID + 1); }
            else {
                printf("Philosopher %d is waiting for fork %d\n", phillID + 1, phillID + 1); }
            }
        }
    }
}

int main() {
    // Initialization
    for (i = 0; i < N; i++) {
        ForkAvailable[i].taken = 0;
        PhilosopherStatus[i].left = 0;
        PhilosopherStatus[i].right = 0;
    }
```

```
// Simulation loop
while (completedPhilo < N) {
    for (i = 0; i < N; i++) {
        goForDinner(i);
    }
    printf("\nTill now, number of philosophers completed dinner: %d\n\n", completedPhilo); }

return 0;
```

```
Till now, number of philosophers completed dinner: 2
Philosopher 1 already completed his dinner
Philosopher 2 already completed his dinner
Philosopher 3 completed his dinner
Philosopher 3 released fork 3 and fork 2
Fork 3 taken by philosopher 4

Till now, number of philosophers completed dinner: 3
Philosopher 1 already completed his dinner
Philosopher 2 already completed his dinner
Philosopher 3 already completed his dinner
Fork 4 taken by philosopher 4

Till now, number of philosophers completed dinner: 3
Philosopher 1 already completed his dinner
Philosopher 2 already completed his dinner
Philosopher 3 already completed his dinner
Philosopher 4 completed his dinner
Philosopher 4 released fork 4 and fork 3

Till now, number of philosophers completed dinner: 4

-----
Process exited after 0.5355 seconds with return value 0
Press any key to continue . . .
```

```
Philosopher 4 already completed his dinner
Philosopher 5 completed his dinner
Philosopher 5 released fork 5 and fork 4
Fork 5 taken by philosopher 6

Till now, number of philosophers completed dinner: 5

Philosopher 1 already completed his dinner
Philosopher 2 already completed his dinner
Philosopher 3 already completed his dinner
Philosopher 4 already completed his dinner
Philosopher 5 already completed his dinner
Fork 6 taken by philosopher 6

Till now, number of philosophers completed dinner: 5

Philosopher 1 already completed his dinner
Philosopher 2 already completed his dinner
Philosopher 3 already completed his dinner
Philosopher 4 already completed his dinner
Philosopher 5 already completed his dinner
Philosopher 6 completed his dinner
Philosopher 6 released fork 6 and fork 5

Till now, number of philosophers completed dinner: 6

-----
Process exited after 0.6122 seconds with return value 0
Press any key to continue . . .
```

```
Philosopher 1 already completed his dinner
Philosopher 2 already completed his dinner
Philosopher 3 already completed his dinner
Philosopher 4 completed his dinner
Philosopher 4 released fork 4 and fork 3
Fork 4 taken by philosopher 5

Till now, number of philosophers completed dinner: 4

Philosopher 1 already completed his dinner
Philosopher 2 already completed his dinner
Philosopher 3 already completed his dinner
Philosopher 4 already completed his dinner
Fork 5 taken by philosopher 5

Till now, number of philosophers completed dinner: 4

Philosopher 1 already completed his dinner
Philosopher 2 already completed his dinner
Philosopher 3 already completed his dinner
Philosopher 4 already completed his dinner
Philosopher 5 completed his dinner
Philosopher 5 released fork 5 and fork 4

Till now, number of philosophers completed dinner: 5

-----
Process exited after 0.6883 seconds with return value 0
Press any key to continue . . .
```