Lab-6

|  |
| --- |
| #include <stdio.h>  #define n 4 // Number of philosophers and forks  int completedPhilo = 0, i;  struct fork {  int taken; // 0 if free, 1 if taken  } ForkAvail[n];  struct philos {  int left; // 1 if left fork taken  int right; // 1 if right fork taken  } PhiloStatus[n];  void goForDinner(int philID) {  // If already completed dinner  if (PhiloStatus[philID].left == 10 && PhiloStatus[philID].right == 10) {  printf("Philosopher %d completed his dinner\n", philID + 1);  }  // If both forks taken just now  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; // Mark completed  int otherFork = philID - 1;  if (otherFork == -1) otherFork = n - 1;  ForkAvail[philID].taken = 0;  ForkAvail[otherFork].taken = 0;  printf("Philosopher %d released fork %d and fork %d\n", philID + 1, philID + 1, otherFork + 1);  completedPhilo++;  }  // If left fork taken, trying for right  else if (PhiloStatus[philID].left == 1 && PhiloStatus[philID].right == 0) {  if (philID == (n - 1)) { // Last philosopher, try reverse  if (ForkAvail[philID].taken == 0) {  ForkAvail[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;  int rightFork = philID - 1;  if (rightFork == -1) rightFork = n - 1;  if (ForkAvail[rightFork].taken == 0) {  ForkAvail[rightFork].taken = PhiloStatus[dupPhilID].right = 1;  printf("Fork %d taken by Philosopher %d\n", rightFork + 1, dupPhilID + 1);  } else {  printf("Philosopher %d is waiting for Fork %d\n", dupPhilID + 1, rightFork + 1);  }  }  }  // No fork taken yet  else if (PhiloStatus[philID].left == 0) {  if (philID == (n - 1)) {  if (ForkAvail[philID - 1].taken == 0) {  ForkAvail[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 (ForkAvail[philID].taken == 0) {  ForkAvail[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++) {  ForkAvail[i].taken = PhiloStatus[i].left = PhiloStatus[i].right = 0;  }  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;  } |



