JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

Lecture-21 By Vivek Dubey

COMPUTER SCIENCE AND ENGINEERING

CS403PC

COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

CS403PC

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Scheme of Operating System

S. No.	Course Code	Course Title	L	T	P	Credits
3	CS403PC	Operating Systems	3	0	0	3
6	CS406PC	Operating Systems Lab	0	0	3	1.5

Lecture-21 By Vivek Dubey

Aim:

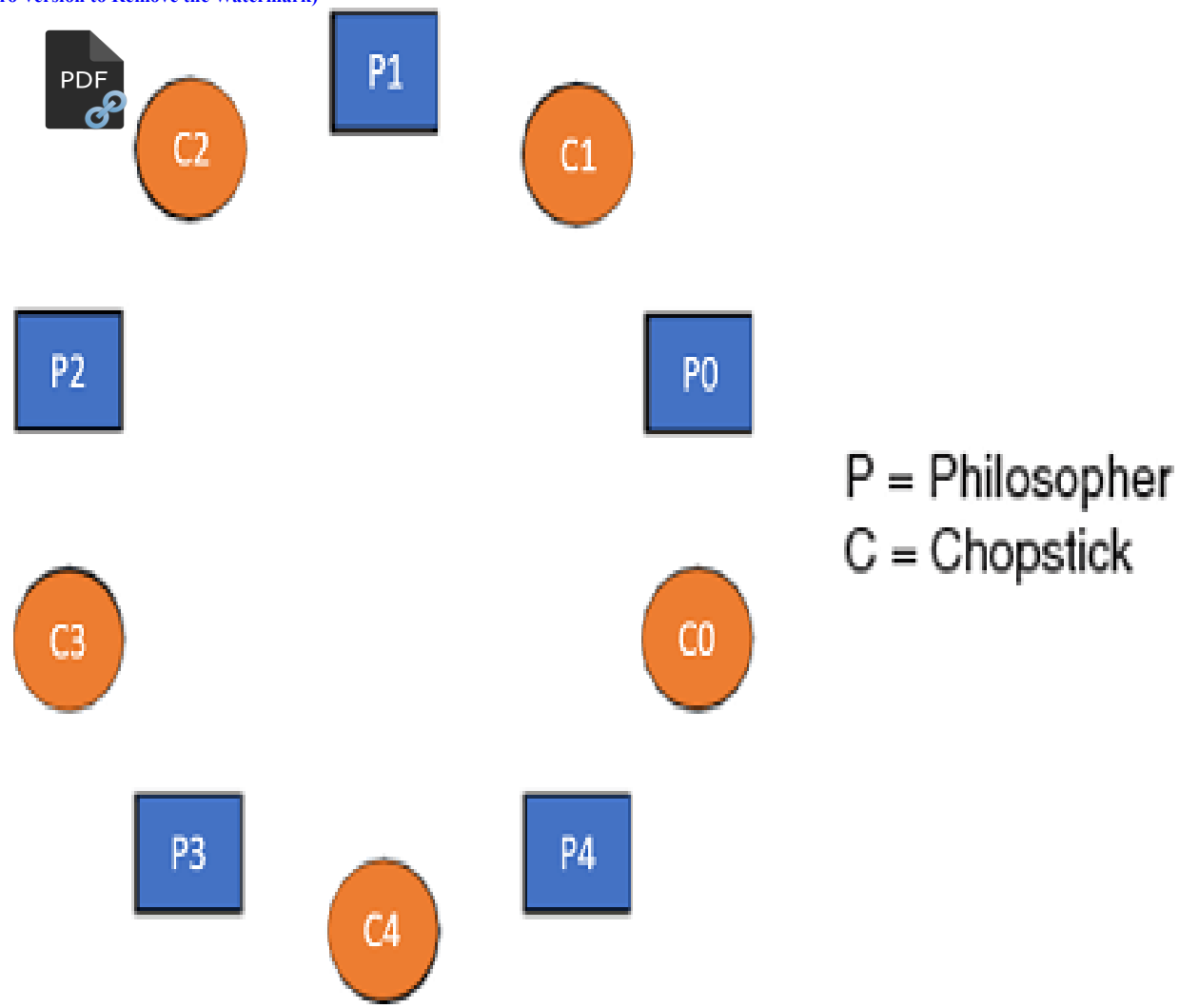
Program to simulate the concept of Dining-Philosophers problem.

THE DINING PHILOSOPHERS PROBLEM

- The dining philosopher's problem is the classical problem of synchronization which says that Five philosophers are sitting around a circular table and their job is to think and eat alternatively.
- A bowl of noodles is placed at the center of the table along with five chopsticks for each of the philosophers.
- To eat a philosopher needs both their right and a left chopstick. A philosopher can only eat if both immediate left and right chopsticks of the philosopher is available.
- In case if both immediate left and right chopsticks of the philosopher are not available then the philosopher puts down their (either left or right) chopstick and starts thinking again.

Dining Philosophers Problem-

- Let's understand the Dining Philosophers Problem with the below code, we have used fig as a reference to make you understand the problem exactly. The five Philosophers are represented as P0, P1, P2, P3, and P4 and five chopsticks by C0, C1, C2, C3, and C4.



What is semaphores?

- Semaphore is a synchronization tool which can be used to deal with critical-section problem.

- A semaphore is a protected variable whose value can be accessed and altered only by the operations P(wait) and V(signal) and an initialization operation we shall call semaphore-initialize.

Algorithm

- Step 1: Start
- Step 2: Define the number of philosophers
- Step 3: Declare one thread per philosopher
- Step 4: Declare one semaphore (represent chopsticks) per philosopher
- Step 5: When a philosopher is hungry
    - See if chopsticks on both sides are free
    - Acquire both chopsticks
    - Eat
    - Restore the chopstick
    - If chopsticks aren’t free
- Step 6: Wait till they are available
- Step 7: Stop

Part-1


```
int tph, philname[20], status[20], howhung, hu[20], cho;
main()
{
int i;
printf("\n\nDINING PHILOSOPHER PROBLEM");
printf("\nEnter the total no. of philosophers: ");
scanf("%d",&tph);
for(i=0;i<tph;i++)
{
philname[i] = (i+1);
status[i]=1;
}
```

Part-2

- printf("How many are hungry : ");
- scanf("%d", &howhung);
- if(howhung==tph)
- {
- printf("\nAll are hungry..\nDead lock stage will occur");
- printf("\nExiting..");
- }
- else
- {
- for(i=0;i<howhung;i++)
- {
- printf("Enter philosopher %d position: ",(i+1));
- scanf("%d", &hu[i]);
- status[hu[i]]=2;
- }

Part-3

- do
- {
- printf("1.One can eat at a time\t2.Two can eat at a time\t3.Exit\nEnter your choice:");
- scanf("%d", &cho);
- switch(cho)
- {
- case 1: one();
- break;
- case 2: two();
- break;
- case 3: exit(0);
- default: printf("\nInvalid option..");
- }
- }while(1);
- }
- }

Part-4

- one()
- {
- int pos=0, x, i;
- printf("\nAllow one philosopher to eat at any time\n");
- for(i=0;i<howhung; i++, pos++)
- {
- printf("\nP %d is granted to eat", philname[hu[pos]]);
- for(x=pos;x<howhung;x++)
- printf("\nP %d is waiting", philname[hu[x]]);
- }
- }

Part-5


```
two()
{
int i, j, s=0, t, r, x;
printf("\n Allow two philosophers to eat at same time\n");
for(i=0;i<howhung;i++)
{
for(j=i+1;j<howhung;j++)
{
if(abs(hu[i]-hu[j])>=1&& abs(hu[i]-hu[j])!=4)
{
printf("\n\ncombination %d \n", (s+1));
t=hu[i];
r=hu[j];
s++;
printf("\nP %d and P %d are granted to eat", philname[hu[i]],
philname[hu[j]]);
for(x=0;x<howhung;x++)
{
if((hu[x]!=t)&&(hu[x]!=r))
printf("\nP %d is waiting", philname[hu[x]]);
}
}
}
}
}
```

RUN

- 1.One can eat at a time
- 2.Two can eat at a time
- 3.Exit Enter your choice: 1
- Allow one philosopher to eat at any time
- P 3 is granted to eat
- P 3 is waiting
- P 5 is waiting
- P 0 is waiting
- P 5 is granted to eat
- P 5 is waiting
- P 0 is waiting
- P 0 is granted to eat
- P 0 is waiting

1. One can eat at a time
2. Two can eat at a time
3. Exit Enter your choice: 2

Allo wo philosophers to eat at same time
combination 1
P 3 and P 5 are granted to eat
P 0 is waiting
combination 2
P 3 and P 0 are granted to eat
P 5 is waiting
combination 3
P 5 and P 0 are granted to eat
P 3 is waiting

1.One can eat at a time
2.Two can eat at a time
3.Exit
Enter your choice: 3