Write a Semaphore solution for dining Philosopher's problem?

The dining philosophers problem is a classic concurrency problem in computer science, which involves a group of philosophers who share a circular table and some chopsticks. Each philosopher alternates between thinking and eating, but they can only eat when they have two chopsticks. If all philosophers try to grab a chopstick at the same time, they may deadlock and none of them will be able to eat.

One solution to the dining philosophers problem is to use semaphores, which are a type of synchronization primitive that allows multiple threads to access a shared resource while preventing race conditions and other synchronization problems.

Here's an example solution that uses semaphores:

```
// define the number of philosophers and chopsticks
#define NUM_PHILOSOPHERS 5
#define NUM_CHOPSTICKS NUM_PHILOSOPHERS

// create an array of semaphores to represent the chopsticks
Semaphore chopstick[NUM_CHOPSTICKS];

// create an array of threads to represent the philosophers
Thread philosopher[NUM_PHILOSOPHERS];

// define a function to simulate a philosopher's behavior
void philosopher_behavior(int id) {
    while (true) {
        // think for a while
        think();

        // pick up the left chopstick
        chopstick[id].wait();

        // pick up the right chopstick
```

```
        chopstick[(id + 1) % NUM_CHOPSTICKS].wait();

        // eat for a while
        eat();

        // put down the right chopstick
        chopstick[(id + 1) % NUM_CHOPSTICKS].signal();

        // put down the left chopstick
        chopstick[id].signal();
    }
}

// initialize the semaphores and start the threads
int main() {
    // initialize the chopstick semaphores
    for (int i = 0; i < NUM_CHOPSTICKS; i++) {
        chopstick[i] = Semaphore(1);
    }

    // start the philosopher threads
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        philosopher[i] = Thread(philosopher_behavior, i);
    }

    // wait for the philosopher threads to finish
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        philosopher[i].join();
    }

    return 0;
}
```

In this solution, we create an array of semaphores to represent the chopsticks, and initialize them with a count of 1 to indicate that they are available. We also create an array of threads

to represent the philosophers, and define a function philosopher_behavior() to simulate each philosopher's behavior.

In the philosopher_behavior() function, the philosopher alternates between thinking and eating. To eat, the philosopher must first acquire both the left and right chopsticks by calling the wait() method on the corresponding semaphores. Once the philosopher has both chopsticks, they can eat for a while, and then put down the chopsticks by calling the signal() method on the corresponding semaphores.

In the main() function, we initialize the semaphores and start the philosopher threads. Finally, we wait for the philosopher threads to finish by calling the join() method on each thread.

Related Posts:

15. SRTF shortest remaining time first

16. OS 4

17. OS 3

18. Os 2

19. Os 1

20. CBSE NET 2004 38

21. Cbse net 2004 37

22. Cbse net 2004

23. CBSE Net 2017

24. Ugc net 2017 solved

25. NET 4

26. NET 1

27. Net 28

28. Net 26

29. Net 50

30. Net 49

31. Net 48

32. Net 46

33. Net 44

34. Net 40

35. Net 39

36. GATE, Longest Remaining Time First Algorithm | Prof. Jayesh Umre

37. GATE SRTF | What is the total waiting time for process P2?

38. GATE Calculate Total Waiting Time SRTF algorithm | Prof. Jayesh Umre

39. Memory management

40. Concept of Threads

41. Process concept

42. Directory Structure OS

43. Contiguous disk space allocation method

44. File systems

45. Types of os

46. Evolution of os

47. Functions of os

48. Why is operating system a mandatory software?

49. Bankers algorithm problems

50. Diploma Linux Unit 3

51. RGPV Diploma Linnux Unit 2

52. Program to print string in reverse order

53. Program to implement while loop in Linux

54. Program to implement for loop using sequence keyword in Liux

55. Program to implement different types of increment in Linux

56. For loop without in keyword in Linux

57. Program to implement for loop using in keyword in Linux

58. Multiple Processor Scheduling

59. What do you mean by Virtual Memory? Write down its advantages?

60. Compare Paging and Segmentation?

61. What is Process Scheduling, CPU Scheduling, Disk Scheduling? Explain Short, Medium and Long term Scheduler?

62. Explain concept of a process with its components ?

63. Explain the following in brief Contiguous and Linked list allocation for implementing file system?

64. Explain various Disk scheduling algorithms with Illustrations ?

65. Define process and thread. What is PCB ? Explain its various entries with their usefulness ?

66. Discuss advantages and disadvantages of the Buffer cache ?

67. Explain different types of OS with examples of each ?

68. What is an Operating System? Write down its desirable characteristics ?

69. Define a deadlock ? Write down the conditions responsible for deadlock? How can we recover from deadlock ?

70. What are the various services provided by Operating system ?

71. What do you mean by PCB? Where is it used? What are its contents? Explain.

72. What is Binary and Counting semaphores ?

73. What is File? What are the different File attribute and operations?

74. What are System call? Explain briefly about various types of system call provided by an Operating System?

75. Describe necessary conditions for deadlocks situation to arise.

76. What are points to be consider in file system design? Explain linked list allocation in detail?

77. Consider the following page reference string:1,2,3,4,5,3,4,1,2,7,8,7,8,9,7,8,9,5,4,5. How many page faults would occur for the following replacement algorithm, assuming four frames:a) FIFOb) LRU

78. Explain CPU schedulers in operating system?

79. Write the different state of a process with the help of Process state deagram?

80. What is Mutex in operating system?

81. Explain Network operating system?

82. What do you mean by paging in operating system ?