**COLLEGE NAME, CITY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**LAB MANUAL**

**Name:** OPERATING SYSTEM

**Course Code:**

**Course:** B.TECH

**Session:**

**Submitted to Submitted by**

**LIST OF EXPERIMENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lab No.** | **Topics to be covered** | **Date of Practical** | **Faculty Signature** |
| 1 | Write a Program to implement FCFS CPU scheduling algorithm. |  |  |
| 2 | Write a Program to implement SJF CPU scheduling algorithm. |  |  |
| 3 | Write a Program to implement Priority CPU Scheduling algorithm. |  |  |
| 4 | Write a Program to implement Round Robin CPU scheduling algorithm. |  |  |
| 5 | Write a program to implement Deadlock. |  |  |
| 6 | Write a Program to implement classical inter process communication problem (producer consumer). |  |  |
| 7 | Write a Program to implement classical inter process communication problem (Reader Writers). |  |  |
| 8 | Write a Program to implement classical inter process communication problem (Dining Philosophers). |  |  |
| 9 | Write a Program to implement FIFO page replacement algorithm. |  |  |
| 10 | Write a Program to implement LRU page replacement algorithm |  |  |

**EXPERIMENT 1**

**AIM**: Program to implement FCFS CPU scheduling algorithm.

**Algorithm:**

1. Start the process

2. Get the number of processes to be inserted

3. Get the value for burst time of each process from the user

4. Having allocated the burst time(bt) for individual processes , Start with the first process from it’s initial position let other process to be in queue

5. Calculate the waiting time(wt) and turnaround time(tat) as

Wt(pi) = wt(pi-1) + tat(pi-1) (i.e wt of current process = wt of previous process + tat of previous process)

tat(pi) = wt(pi) + bt(pi) (i.e tat of current process = wt of current process + bt of current process)

6. Calculate the total and average waiting time and turnaround time

7. Display the values

8. Stop the process

**Program:**

**EXPERIMENT 2**

**Algorithm:**

1. Start the process

2. Get the number of processes to be inserted

3. Sort the processes according to the burst time and allocate the one with shortest burst to execute first

4. If two processes have same burst length then FCFS scheduling algorithm is used

5. Calculate the total and average waiting time and turnaround time

6. Display the values

7. Stop the process

**Program:**

**EXPERIMENT 3**

**AIM:** Program to implement Priority CPU Scheduling algorithm.

 **Algorithm:**

1. Start the process

2. Get the number of processes to be inserted

3. Get the corresponding priority of processes

4. Sort the processes according to the priority and allocate the one with highest priority to execute first

5. If two process have same priority then FCFS scheduling algorithm is used

6. Calculate the total and average waiting time and turnaround time

7. Display the values

8. Stop the process

**Program:**

**EXPERIMENT 4**

**AIM:** Program to implement Round Robin CPU scheduling algorithm.

**Algorithm:**

1. Start the process

2. Get the number of elements to be inserted

3. Get the value for burst time for individual processes

4. Get the value for time quantum

5. Make the CPU scheduler go around the ready queue allocating CPU to each process for the time interval specified

6. Make the CPU scheduler pick the first process and set time to interrupt after quantum. And after it's expiry dispatch the process

7. If the process has burst time less than the time quantum then the process is released by the CPU

8. If the process has burst time greater than time quantum then it is interrupted by the OS and the process is put to the tail of ready queue and the schedule selects next process from head of the queue

9. Calculate the total and average waiting time and turnaround time

10. Display the results

11. Stop the process

**Program:**

**EXPERIMENT 5**

**AIM:** Program to implement Deadlock.

**Program:**

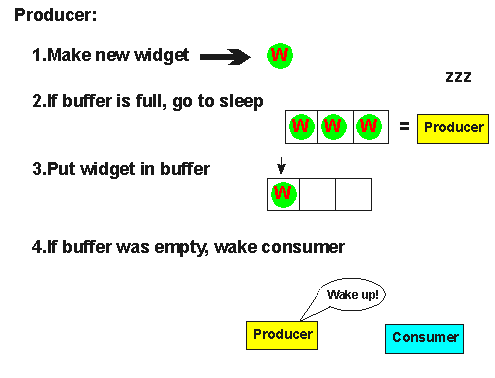
**EXPERIMENT 6**

**AIM:** Program to implement classical inter process communication problem (producer consumer)

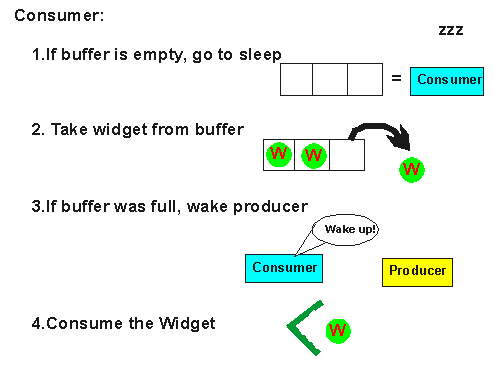
#### Algorithm:

The producer-consumer problem illustrates the need for synchronization in systems where many processes share a resource. In the problem, two processes share a fixed-size buffer. One process produces information and puts it in the buffer, while the other process consumes information from the buffer. These processes do not take turns accessing the buffer, they both work concurrently. Here in lies the problem. What happens if the producer tries to put an item into a full buffer? What happens if the consumer tries to take an item from an empty buffer?

In order to synchronize these processes, we will block the producer when the buffer is full, and we will block the consumer when the buffer is empty. So the two processes, Producer and Consumer, should work as follows:



1. The producer must first create a new widget.
2. Then, it checks to see if the buffer is full. If it is, the producer will put itself to sleep until the consumer wakes it up. A "wakeup" will come if the consumer finds the buffer empty.
3. Next, the producer puts the new widget in the buffer. If the producer goes to sleep in step (2), it will not wake up until the buffer is empty, so the buffer will never overflow.
4. Then, the producer checks to see if the buffer is empty. If it is, the producer assumes that the consumer is sleeping, an so it will wake the consumer. Keep in mind that between any of these steps, an interrupt might occur, allowing the consumer to run.



1. The consumer checks to see if the buffer is empty. If so, the consumer will put itself to sleep until the producer wakes it up. A "wakeup" will occur if the producer finds the buffer empty after it puts an item into the buffer.
2. Then, the consumer will remove a widget from the buffer. The consumer will never try to remove a widget from an empty buffer because it will not wake up until the buffer is full.
3. If the buffer was full before it removed the widget, the consumer will wake the producer.
4. Finally, the consumer will consume the widget. As was the case with the producer, an interrupt could occur between any of these steps, allowing the producer to run.

**Program:**

**EXPERIMENT 7**

**AIM:** Program to implement classical inter process communication problem (Reader Writers).

**Program:**

**EXPERIMENT 8**

**AIM:** Program to implement classical inter process communication problem (Dining Philosophers).

**Program:**

**EXPERIMENT 9**

**AIM:** Program to implement FIFO page replacement algorithm.

**ALGORITHM**

1. Start the process

2. Declare the size with respect to page length

3. Check the need of replacement from the page to memory

4. Check the need of replacement from old page to new page in memory

5. Forma queue to hold all pages

6. Insert the page require memory into the queue

7. Check for bad replacement and page fault

8. Get the number of processes to be inserted

9. Display the values

10. Stop the process

**Program:**

**EXPERIMENT 10**

**AIM:** Program to implement LRU page replacement algorithm.

**ALGORITHM**

1. Start the process

2. Declare the size

3. Get the number of pages to be inserted

4. Get the value

5. Declare counter and stack

6. Select the least recently used page by counter value

7. Stack them according the selection.

8.  Display the values

9. Stop the process

**Program:**