**OPERATING SYSTEM (PRACTICALS) – FALL 2012**

**EXPERIMENT 7 – PETERSON’S ALGORITHM**

|  |  |  |  |
| --- | --- | --- | --- |
| **DATE: \_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | **Students Names: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | |
| **Marks Obtained: \_\_\_\_\_** | | **COURSE: BESE 16 \_\_\_\_\_** | |
| **Deadline: 1400 hrs 2nd Dec 2012** | | **Instructor: Engr. Umar Mahmud** | |
|  | **Instructions**   * This lab is to be performed by a syndicate of at most **TWO** students. Write your remarks next to the space provided. * Plagiarism is strictly forbidden. * Submit hard copy of the report before deadline. Marks will be deducted for late submissions. | |  |
| 1. | **Objectives:**   1. Critical Section 2. Peterson’s Algorithm | |  |
| 2. | **Time Required:** 3 hrs | |  |
| 3. | **Software Required:**   1. Java/C/C++/C# 2. Windows/Ubuntu | |  |
| 4. | Create two processes (producer and consumer) or functions that access the same shared variable. One process increments the variable while the other decrements it. The maximum value of the variable is 5 and the producer cannot produce anymore items if the variable is 5. The consumer cannot consume if the variable is zero. | |  |
| 5. | In the main function write a sequence of calls that simulate the producer consumer problem. Output the normal case, the case where the consumer should be blocked and the case where the producer should be blocked. Show the trace only.  \*write function calls.  and thier output.  three cases: normal, producer being blocked, consumer being blocked | | (2) |
| 6. | How does **race condition** occur in the cases of point 5? Explain and trace the case. | | (2) |
| 7. | **Critical Section:** Access to a shared item is considered critical and is carried out in critical section only. | |  |
| 8. | **Peterson’s Solution:** It provides a solution to the Critical Section.  Two processes share two variables:  int **turn**;  boolean **flag[2];**  //The variable **turn** indicates whose turn it is to enter the critical section  //The **flag** array is used to indicate if a process is ready to enter the critical section. //**flag[i]** = true implies that process **Pi** is ready! | |  |
| 9. | **Algorithm for Process Pi:**  do {  flag[i] = True;  turn = j;  while (flag[j] && turn = j)  //do nothing  //go to critical section i.e., access the shared item  flag[i] = FALSE;  // go to remainder section  } while (TRUE); | |  |
| 10. | Implement and show trace of Peterson’s Algorithm for two processes. Show output here. | | (3) |
| 11. | Implement and show trace of Peterson’s Algorithm for three processes. Show output here. | | (3) |