SEMAPHORES

A semaphore is an integer variable that apart from initialisation is accessed only through two standard atomic operators.

In two processes, critical section solution can be possible using Peterson's Solution.

But an operating system is a n process system (many process system) so Semaphore solution is used here. Because Semaphore gives n process solution.

Semaphores are simple integer variables, as many people gets confusion as it is a separate data type or data structure so no, it's an integer variable only.

We use "int S" to show semaphore.

But don't be confuse we can use any thing like int a, int b etc to show semaphore.

Its seems easy to show Semaphore using "S" so we use "int S".

int S = 1; (this is known as initialisation as well as declaration)

Here initialisation of Semaphore shows that which problem you wants to solve by Semaphore.

We use "int S=1" so problem 1 will get solved in critical section.

And it's important to know that when we solve critical section problem using Semaphore we initialise S with 1.

If once we initialised it than we cant access it again. It can be accessed only by-

1) Wait (S) //Read as wait of S

2) Signal (S) // Read as signal of S

1) Wait(): this is a simple atomic operation, which reduces the value of S by 1. It means whatever the value of S, wait () will do a single decrement.

For example:

If S = 3,

Than wait() will just reduces it to S=2.

```
Wait (S)
{
While (S ≤ 0);
```

```
S = S-1;
}
```

See in above code while loop is a type of trap only which makes program to check again and again whether value of S is greater than zero.

If value of S is greater than zero, so wait(S) can make a single decrements in it.

2) Signal(): this is a simple atomic operation, which increases the value of S by 1. It means whatever the value of S, Signal () will do a single increment.

For example:

If S = 3,

Than Signal() will just increases it to S=4.

```
Signal (S)
{
S = S+1;

}
```

References:

1. Sebesta,"Concept of programming Language", Pearson Edu
2. Louden, "Programming Languages: Principles & Practices" , Cengage Learning
3. Tucker, "Programming Languages: Principles and paradigms ", Tata McGraw –Hill.
4. E Horowitz, "Programming Languages", 2nd Edition, Addison Wesley

Related Posts:

1. Sequence Control & Expression | PPL
2. PPL:Named Constants
3. Parse Tree | PPL | Prof. Jayesh Umre

4. Basic elements of Prolog
5. Loops | PPL | Prof. Jayesh Umre
6. Subprograms Parameter passing methods | PPL | Prof. Jayesh Umre
7. Programming Paradigms | PPL | Prof. Jayesh Umre
8. Subprograms Introduction | PPL | Prof. Jayesh Umre
9. Phases of Compiler | PPL | Prof. Jayesh Umre
10. Parse Tree | PPL
11. Influences on Language design | PPL | Prof. Jayesh Umre
12. Fundamentals of Subprograms | PPL | Prof. Jayesh Umre
13. Programming Paradigm
14. Influences on Language Design
15. Language Evaluation Criteria
16. OOP in C++ | PPL
17. OOP in C# | PPL
18. OOP in Java | PPL
19. PPL: Abstraction & Encapsulation
20. PPL: Introduction to 4GL
21. PPL: Variable Initialization
22. PPL: Conditional Statements
23. PPL: Array
24. PPL: Strong Typing
25. PPL: Coroutines
26. PPL: Exception Handler in C++
27. PPL: OOP in PHP
28. PPL: Character Data Type
29. PPL: Exceptions
30. PPL: Heap based storage management

31. PPL: Primitive Data Type
32. PPL: Data types
33. Programming Environments | PPL
34. Virtual Machine | PPL
35. PPL: Local referencing environments
36. Generic Subprograms
37. Local referencing environments | PPL | Prof. Jayesh Umre
38. Generic Subprograms | PPL | Prof. Jayesh Umre
39. PPL: Java Threads
40. PPL: Loops
41. PPL: Exception Handling
42. PPL: C# Threads
43. Pointer & Reference Type | PPL
44. Scope and lifetime of variable
45. Design issues for functions
46. Parameter passing methods
47. Fundamentals of sub-programs
48. Subprograms
49. Design issues of subprogram
50. Garbage Collection
51. Issues in Language Translation
52. PPL Previous years solved papers
53. Type Checking | PPL | Prof. Jayesh Umre
54. PPL RGPV May 2018 solved paper discussion| Prof. Jayesh Umre
55. PPL Viva Voce
56. PPL RGPV June 2017 Solved paper | Prof. Jayesh Umre
57. Concurrency

58. Basic elements of Prolog
59. Introduction and overview of Logic programming
60. Application of Logic programming
61. PPL: Influences on Language Design
62. Language Evaluation Criteria PPL
63. PPL: Sequence Control & Expression
64. PPL: Programming Environments
65. PPL: Virtual Machine
66. PPL: Programming Paradigm
67. PPL: Pointer & Reference Type
68. try-catch block in C++