Assignment 4: Classical Problems of Synchronization

Goal of the Assignment:

- To study concept of Concurrent Threads. Thread Synchronization.
- To understand the race condition, achieving synchronization using semaphores
- Use semaphores to solve Reader Writer Problem.

Part A: Producer Consumer problem [Total: 8 Points]

Write a program for producer and consumer problem. Create a producer thread and a consumer thread, create a buffer (an array) and allocate memory. Use a shared variable to count the current size of buffer. The producer thread should write some data into the buffer and consumer thread should read the data from the buffer and flush the data item once read, the read data must be displayed. This write and read must be done mutually exclusive. When producer writes data in buffer the consumer must wait for a specified time. Similarly, when consumer reads the buffer, producer must wait. Shared variable counter value should be incremented once data is placed in buffer and decremented if data is flushed.

Sample Input/Output:

Producer thread: 1 2 3 4 5 6 7 8 9 10….(should write for 10 milliseconds)
Consumer thread: 1 2 3 4 5 6 7 8 9 10…(read all the data)
(flush all data….Buffer should be empty..)

Producer thread:...50 51 52 53 54 55….(should write for 10 milliseconds)
Consumer thread …. 50 51 52 53 54 55 56… (read all the data)
(flush all the data buffer should be empty)

Part B: Process Synchronization-Semaphores [Total: 8 Points]

Write a program using semaphore for synchronization between threads. A process can create two threads which have access to the common variable count whose initial value is 0. One thread will increment it till it reaches 20 and other thread should read each of the incremented value and print the value with the thread id.

Sample input / output:

when the user runs the executable file the program should print

$ Count is 0 and <thread id of the process that is incrementing the value>

$ Count is 0 and <thread id of the process that is reading and printing>

$ 1 and <thread id of the process that is incrementing the value>

$ 1 and <thread id of the process that is reading and printing>

$ 2 and <thread id of the process that is incrementing the value>

$ 2 and <thread id of the process that is reading and printing>

.

. .

.

$ 20 and <thread id of the process that is incrementing the value>

$ 20 and <thread id of the process that is reading and printing>

Part C: Reader Writer Problem [Total: 8 Points]

Write a program to solve the issues in reader writer problem. Reader and writer are two threads which try to access the shared variable for reading and writing respectively. When a reader thread has gained access to the shared variable then writer should wait and vice versa. There can be any number of readers reading the shared variable at the same time. At most only one writer should be allowed to write the shared variable at once. Solve this problem using semaphores and shared variables.

Simple Procedure

1. Declare two semaphores, reader_smph and writers_smph (usually integers they can take values 0 or 1) initialize all semaphores to 1.

2. Declare readers_count and a shared variable.

3. Create writer and reader threads, when a writer is grant access(writing) over the shared variable no reader should be allowed to read the variable (i.e. decrement the writers semaphore, writers_smph variable to 0). once writing is done writers_smph can be made as 1.

4. If a reader is reading the shared variable(increment readers count by 1 for each reader thread), there can be multiple readers reading at the same time. Writer must wait until every reader completes the read operation(if a reader leaves then decrement readers count. Only if readers count becomes 0 then writers_smph can be made as 0 and access is granted to the writer).

5. Print the shared variable value read by the readers.

6. Writer should increment the value of shared variable (eg. a[1]=a[1]+1) and print them.

7. Iterate the above actions.

Sample Input/output:

Writer count :1

Writer 1 - writing the variable:10

Writer count :0 (writer to wait for 10 ms)

Reader 1 Reading the variable : 10

Readers count :1

Reader 2 Reading the variable : 10

Readers count :2

Reader 1 left (Reader 1 waiting for 100ms to read the variable again) this might be random

Readers count :1

Reader 3 Reading the variable : 10

Readers count :2

Reader 4 Reading the variable : 10

Readers count :3

Reader 5 Reading the variable : 10

Readers count :4

Reader 2 left(Reader 2 waiting for 100ms)

Readers count :3

Reader 3 left(Reader 3 waiting for 100ms)

Readers count :2

Reader 4 left(Reader 4 waiting for 100ms)

Readers count :1

Reader 5 left(Reader 5 waiting for 100ms)

Readers count :0

Writer count :1

Writer 1 - writing the variable:11

Writer count :0(writer to wait for 10 ms)

.

.

.

Deliverables in a tar ball on GC:

- Submission Guidelines: Upload the assignment report, code files, README, etc in GC as a tar ball with file name as <your roll no>_<your name>.tar
- Readable Report [1 point for report quality] enumerating steps followed with screenshots for each of the important steps. Put the screenshots in the report for better clarity.

Check Web sources for more information