Assignment 4 MTL458

October 2023

1 Reader Writer Problem

Implement multiple Readers-Writers problem using semaphores.

Your program should ensure bounded waiting for the writer. Assume that the directory with your code file has multiple text files that can be read or written to.

- The Assignment has to be done in C.
- You have to implement the reader and writer locks.
- Implement the bounded waiting strategy to limit the starvation for the writer. You can use any parameters of your liking, as long as there is no starvation.
- The program should run until

  ```
  exit
  ```

  command is given as an input.
- A separate thread should be created for each read and write operation.
- The command to read will be of the format:

  ```
  read asd.txt
  ```

  - On receiving this input, your program should make a new thread for each reader to read the specified file.
  - Also make it print a message containing the name of the file it read, the size of the file, and how many readers and writers were present. (How many readers and writers had their respective lock at that time).

- This should be printed after the reading is done but the lock has not been released by that specific thread.
- The format can be:

```
read asd.txt of x bytes with n readers and m writers present
```

- The command to write can be of the following format:

```
write 1 asd.txt efg.txt
or
write 2 asd.txt I am the text you need to add to asd.txt
```

- On receiving this input, your program should make new a thread for each writer.
  - If the command is write 1, then write the contents of the file efg.txt to the file asd.txt .
  - If the command is write 2, then write the string present after the name of the text file to the file asd.txt
  - Also make each writer print a message containing the name of the file it wrote to, the size of the write made, and how many readers and writers were present. (How can readers and writers had their locks at that time).
  - This should be printed after the writing is done but the lock has not been released. The format can be:

    ```
    writing to asd.txt added x bytes with n readers and m writers present
    ```

- An example input sequence can be:

```
read asd.txt
write 1 asd.txt sad.txt
read asd.txt
write 2 asd.txt 34324213
read asd.txt
read sad.txt
read asd.txt
write 1 asd.txt dfg.txt
read asd.txt
exit
```

- Each writer should add the contents to the file starting from a new line.

- The assignment will be evaluated based on the correct implementation of the reader writer locks, presence of no dead locks or starvation, the implementing of the bounded waiting and the correctness of the mutual exclusion.
- Submit the completed code file. Name it (Entry No 1)_(Entry No 2)_a.c (Note the a)

2 AVL Tree

- Implement the dynamic (insertion/deletion/contains) data structure AVL Tree in concurrent environment.
- Here the data structure is shared among multiple threads responsible for each of the operations: insertion, deletion, contains and in-order traversal.
- Compare the behaviour of this implementation with the traditional one in a report.
- For each insertion, you are supposed to make a separate thread. Same goes for deletion, contains and any in-order traversals.
- For insertion, deletion and contains, you will be given the value to insert or delete or accertain if it is in the tree.
- For in-order traversal, just print out the traversal in one line separated by space.
- At the end, also print the pre order traversal of the tree.
- Example input:

```
insert 3
insert 4
insert 2
delete 2
contains 3
in order
```

- Example output:

```
yes
3 4
3 4
```

- Name this code file as Name it (Entry No 1)_(Entry No 2)_b.c (Note the a)

- Zip the code files and the report in a folder named (Entry No 1)_(Entry No 2).
- The assignment will be checked for plagiarism against the internet and among your peers. Please don't copy, there will be strict penalties.