```
CMPT 300 Assignment 3
Jiadi Luo
301354107
```

This report addresses some potential uncertainties related to the implementation of A3.

1. File Path

The program uses the file path "inputs/metadata.txt" by default. Feel free to modify the file paths in the code to match the actual file locations when testing.

• In *myChannels.c* program:

```
char* metadata_file_path = "inputs/metadata.txt";
char* output file path = "output.txt";
```

• In *metadata.txt* file:

```
Assignment3 > inputs > ≡ metadata.txt

1 2
2 input_files/file_1.txt
3 1
4 1
5 input_files/file_2.txt
6 0.5
7 0.5
8
```

2. Multithreading with pthread mutex t

The program utilizes multithreading with the help of *pthread_mutex_t* and related functions such as:

- pthread mutex lock,
- pthread mutex unlock,
- pthread mutex init, and
- pthread_mutex_destroy.

Reference in Piazza: https://piazza.com/class/lhaqnxludbb4e3/post/368

3. Locks

Three lock implementations (global lock, entry lock, and compare-and-swap lock) can be found under *write to output file* function:

- Global Lock (*lock config* == 1):
 - Purpose: Ensures exclusive access to the output channel by using a single global lock.
 - Usage: The program waits until the *global_lock* variable becomes 0, indicating that no other thread holds the lock. It then acquires the lock by incrementing *global_lock* to 1. After performing the required operations, it releases the lock by decrementing *global_lock* back to 0.

- Entry Lock ($lock\ config == 2$):
 - Purpose: Provides exclusive access to each entry in the output channel by using separate locks for each entry.
 - Usage: The program waits until the *entry_lock* variable becomes 0, indicating that the entry is not locked by another thread. It then acquires the lock by incrementing *entry_lock* to 1. After performing the necessary operations, it releases the lock by decrementing *entry_lock* back to 0.

- Compare-and-Swap Lock (lock config == 3):
 - Purpose: Uses a compare-and-swap mechanism to update entries in the output channel in a lock-free manner.
 - Usage: The program uses a *compare_and_swap* function to atomically check and update the *global_lock* variable. It repeatedly attempts to compare the value of *global_lock* with 0 and sets it to 1 only if the comparison succeeds. Once the operation is completed, it sets *global_lock* back to 0 to release the lock.

```
for (int j = 0; j < buffer float list length; j++) {</pre>
                        // compare and swap
245
                        while (compare and swap(&global lock, 0, 1) != 0);
246
           // compare and swap
           int compare and swap(int *value, int expected, int new value) {
      57
               int temp = *value;
      58
      59
               if (temp == expected) {
      60
                   *value = new value;
      61
               return temp;
      62
      63
```

4. Handling BOM and \r

The program assumes that the input files don't have a Byte Order Mark (BOM) and carriage return characters (\r) since it is specifically designed for a Linux system. This assumption is valid since Linux-based systems generally don't include BOM and treat newline characters (\n) as line terminators, instead of the carriage return and newline combination (\r\n) used in Windows systems.

Reference in Piazza: https://piazza.com/class/lhagnxludbb4e3/post/365

5. Logic of the Program

The program's logic involves the following steps:

• Get input from the user: buffer size, thread num, lock config:

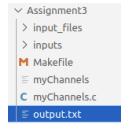
```
• jiadil@jiadil:~/Desktop/CMPT-300/Assignment3$ ./myChannels
Enter buffer size: 2
Enter number of threads: 2
Enter lock config (1: global lock, 2: entry lock, 3: compare and swap): 1
All threads finished
```

• If the user inputs the wrong thread_num, the program will print "file is not in multiple of threads" exit will error code 1:

```
    jiadil@jiadil:~/Desktop/CMPT-300/Assn3$ ./myChannels
    Enter buffer size: 2
    Enter number of threads: 5
    Enter lock config (1: global lock, 2: entry lock, 3: compare and swap): 1
    file is not in multiple of threads
```

• If the user inputs the wrong lock_config, the program will print "file is not in multiple of threads" exit will error code 1:

• If all inputs are correct, the program will process the input files and write the calculated results to the "output.txt" file. It is important to note that the output will not be displayed in the console. To view the results, please refer to the "output.txt" file:



```
Assignment3 > ≡ output.txt

1 6
2 5
3 5
4 9
5
```