

## 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:

```
12 char* metadata_file_path = "inputs/metadata.txt";
13 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.

```
71         if (lock_config == 1) {  
72             // waiting global lock  
73             while (global_lock == 1);  
74             global_lock = 1;
```

- 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.

```
118         for (int j = 0; j < buffer_float_list_length; j++) {  
119             // wait for entry lock  
120             while (entry_lock == 1);  
121             entry_lock = 1;
```

- 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.

```
154         for (int j = 0; j < buffer_float_list_length; j++) {  
155             // compare and swap  
156             while (compare_and_swap(&global_lock, 0, 1) != 0);
```

```
43     // compare and swap  
44     int compare_and_swap(int *value, int expected, int new_value) {  
45         int temp = *value;  
46         if (temp == expected) {  
47             *value = new_value;  
48         }  
49         return temp;  
50     }
```

#### 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/lhaqnxludbb4e3/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/Assn3$ ./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:

```
⊗ jiadil@jiadil:~/Desktop/CMPT-300/Assn3$ ./myChannels
Enter buffer size: 2
Enter number of threads: 2
Enter lock config (1: global lock, 2: entry lock, 3: compare and swap): 5
Invalid lock config!
```

- 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	Assignment3 > output.txt
> input_files	1 6
> inputs	2 5
Makefile	3 5
myChannels	4 9
myChannels.c	5
output.txt	