CSE 333 - OPERATING SYSTEMS
Programming Assignment # 3
Report
Marmara University – 2014

Akif Batur - 150111854

Writer Threads: Merge files
Reader Threads: Merge buffers

Valid inputs:
1. ./merge -n 2 file1 file2 -o outputfile
2. ./merge -n 4 file1 file2 file3 file4 -o outputfile
3. ./merge -n 8 file1 file2 file3 file4 file5 file6 file7 file8 -o outputfile

Functions:
1.errorCheck: Checks if there is an error on usage.
2.fileCheck: Check if given files are exist on file system or not.
3.merge: Merge two given array into another one.
4.readAndMergeBuffers: Read two buffers and merge them into another array.
5.readAndMergeFiles: Read two given files and merge them into a buffer.
6.writer: Writer threads will execute this function to merging two given files.
7.reader: Reader threads will execute this function to merging two buffers that are created after execution of writer threads.

Step by Step Code:
1.Execute errorCheck function to detect if there is any error on usage.
2.Execute fileCheck function to check if the given files are exist on the file system or not.
3.If there is no error on first and second steps, create writer and reader threads.
4.Join threads.
5.Start execution of the threads.
6.Execute writer function with writer threads.
7.Merge two given files and create intermediate buffers. Reader threads will wait for the creation of the buffers.
8. After creating buffers, execute reader function with reader threads.
9. Merge two buffers and create the last buffer.
10. Last reader thread will write the last buffer into the given output file.

Writer Function:
When a writer thread execute writer function, it send its ID to the this function. Then we check the ID of the thread and execute readAndMergeFiles function with sending the ID of the thread to merge the given files. In the readAndMergeFiles function, we decide which files will be merged into which buffer by using the ID of the writer thread. Then we just read files, write them into two arrays, merge the arrays into a final array and write this array into a buffer. After return from writer function we set a global variable called threadN. This variable will be checked by a reader thread in the reader function.

Assume that the user enter this input:
```
./merge -n 8 file1 file2 file3 file4 file5 file6 file7 file8 -o outputfile
```

We will need to create 4 writer threads to merge files.
1.First writer thread will merge file1 and file2 and will set thread1 variable.
2.Second writer thread will merge file3 and file4 and will set thread2 variable.
3.Third writer thread will merge file5 and file6 and will set thread3 variable.
4.Fourth writer thread will merge file7 and file8 and will set thread4 variable.

Reader Function:
When a reader thread execute the reader function it send its ID to this function. Then each reader thread fall into a infinite loop to wait the buffers get ready to be read.

Assume that the user enter this input:
```
./merge -n 8 file1 file2 file3 file4 file5 file6 file7 file8 -o outputfile
```

We will need to create 3 reader threads to merge buffers.
1. First reader will wait for the thread1 and thread2 are set.
2. When the first reader get a change to be executed, it will merge buffer1 and buffer2 into buffer5 and set another variable called thread5. This indicates the fifth buffer is ready.
3. Second reader will wait for the thread3 and thread4 are set.
4. When the second reader get a change to be executed, it will merge buffer3 and buffer4 into buffer6 and set another variable called thread6. This indicates the sixt buffer is ready.
5. Therefore the third and the last reader thread will and must wait for the first and the second readers to be completed.
6. When the third reader get a change to be executed, it will read the buffer5 and buffer6 and merge them into a last array. Then the last thread will write this array to the given output file.
Differentiating the Situations
We are using a variable called stopRead. It is unset at the beginning.

Case 1:
./merge -n 2 file1 file2 -o outputfile stopRead = 2
We will need 1 writer thread to merge file1 and file2. This thread will be called “the first writer Marmara University - 2014thread”. Same writer thread will write the merged result to the output file. When the first writer thread
finish merging files it will check the stopRead variable. If it's equal to 2 that means understand that this is the first case and there will be no more writer and reader thread. So it can write the buffer1 into the given output file.

Case 2: ./merge -n 4 file1 file2 file3 file4 -o outputfile
We will need 2 writer threads and 1 reader thread. First writer thread will merge file1 and file2 into buffer1 and second writer thread will merge file3 and file4 into buffer2. Reader thread will merge the buffer1and buffer2 into buffer3. When the reader thread check stopRead variable after merging buffers, it will understand that the process is completed. It then will write the buffer3 into the given output file.

Case 3: ./merge -n 8 file1 file2 file3 file4 file5 file6 file7 file8 -o outputfile
We will need 4 writer threads and 3 reader threads. We have already told this case above in the Differentiating the Situations section of the report. There will be no need to check stopRead variable for this case because when the last reader is completed there will be no more threads that working. So the last thread will write the last buffer (which is buffer7) into the given output file.