CSCE 313 Programming Assignment 3
Threading and Synchronization

Aldo Leon Marquez UIN: 326004699

The goal of the assignment is to introduce us to what threading is, how to do a basic implementation and the benefits and limits of it. To do so we build on top of the concepts, and code, from the previous assignment. We start with the server and client, and then proceed to split the workload into threads.

To First step to implement the assignment is to create a Bounded Buffer, a B.B allows us to use the concept of work stealing were the maximum fold-up is no longer dependent on the producer of data, rather, we have a pool of jobs that “workers” can take from, the B.B allows us break that foldup dependency and also prevent the pool from being to big, hence why its bounded.

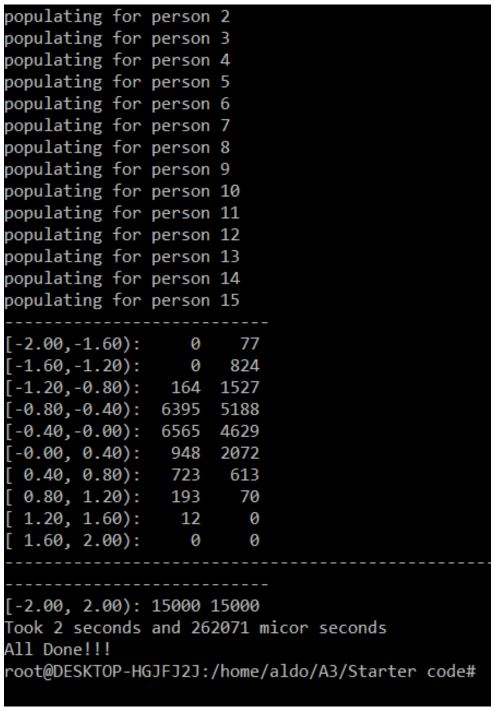
The base code for the B.B was provided, with the missing implementation of the push and pop int the queue , job pool. The important part of this to is to implement the regular push and pop features thread safely and also when they are supposed to be called, we use a mutex lock to prevent the critical section from being affected, and a wait for both push and pop to allow the thread to continue when available.

After that, we just start to thread the client, to do so we need Producer and Consumer threads, this threads are stored in a vector to ensure they do go out of scope. Each Consumer will have a histogram object to print at the end, and each Consumer will have a FIFO Channel. Once the vectors are correctly filled with the corresponding threads, its time to join and execute the functions of each.

The Consumer will execute the patient_funciton, a function that simply formats the server messages and pushes to the job pool, its also in charge of, when all of them are done, to send the QUIT_MESSAGE to the B.B so that the workers can end.

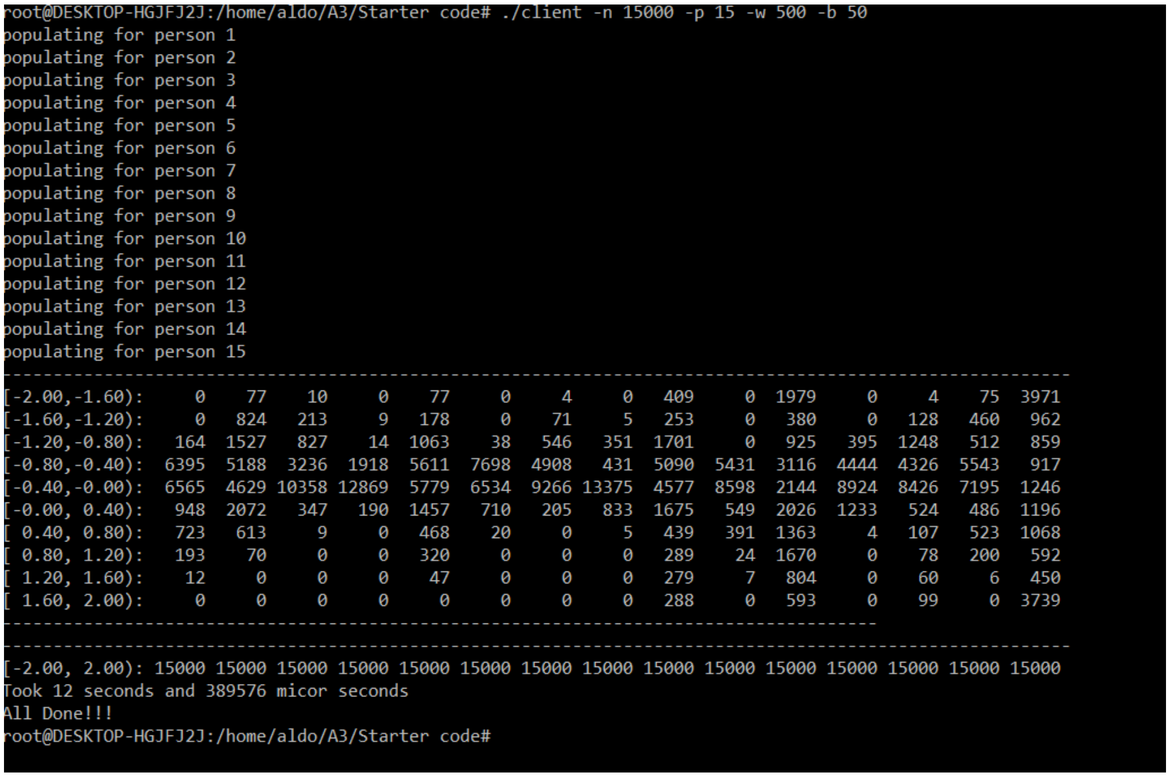
The Workers will execute the worker_function that creates the Channel for the worker, pops a job from the pool and repeats until the EXIT_MESSAGE is popped(note: it is important to push it back so other thread can get it), The function is also in charge of updating the Histogram and to handle both Data and File Msg, to handle those, an int is passed indicating the mode 0 or 1, it will then work the correct logic accordingly

The final implementation does all of the expected, with a few minor details. The images below show some examples of it



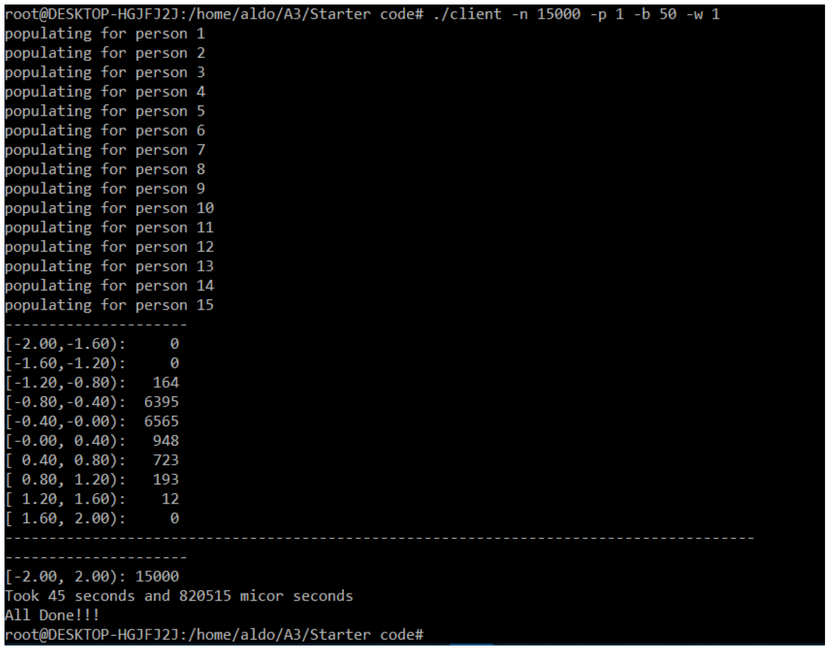
```
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
---------------------------
[-2.00,-1.60):     0    77
[-1.60,-1.20):     0   824
[-1.20,-0.80):   164  1527
[-0.80,-0.40):  6395  5188
[-0.40,-0.00):  6565  4629
[-0.00, 0.40):   948  2072
[ 0.40, 0.80):   723   613
[ 0.80, 1.20):   193    70
[ 1.20, 1.60):    12     0
[ 1.60, 2.00):     0     0
--------------------------------------------------
---------------------------
[-2.00, 2.00): 15000 15000
Took 2 seconds and 262071 micor seconds
All Done!!!
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code#
```

Output for the Pdf Example



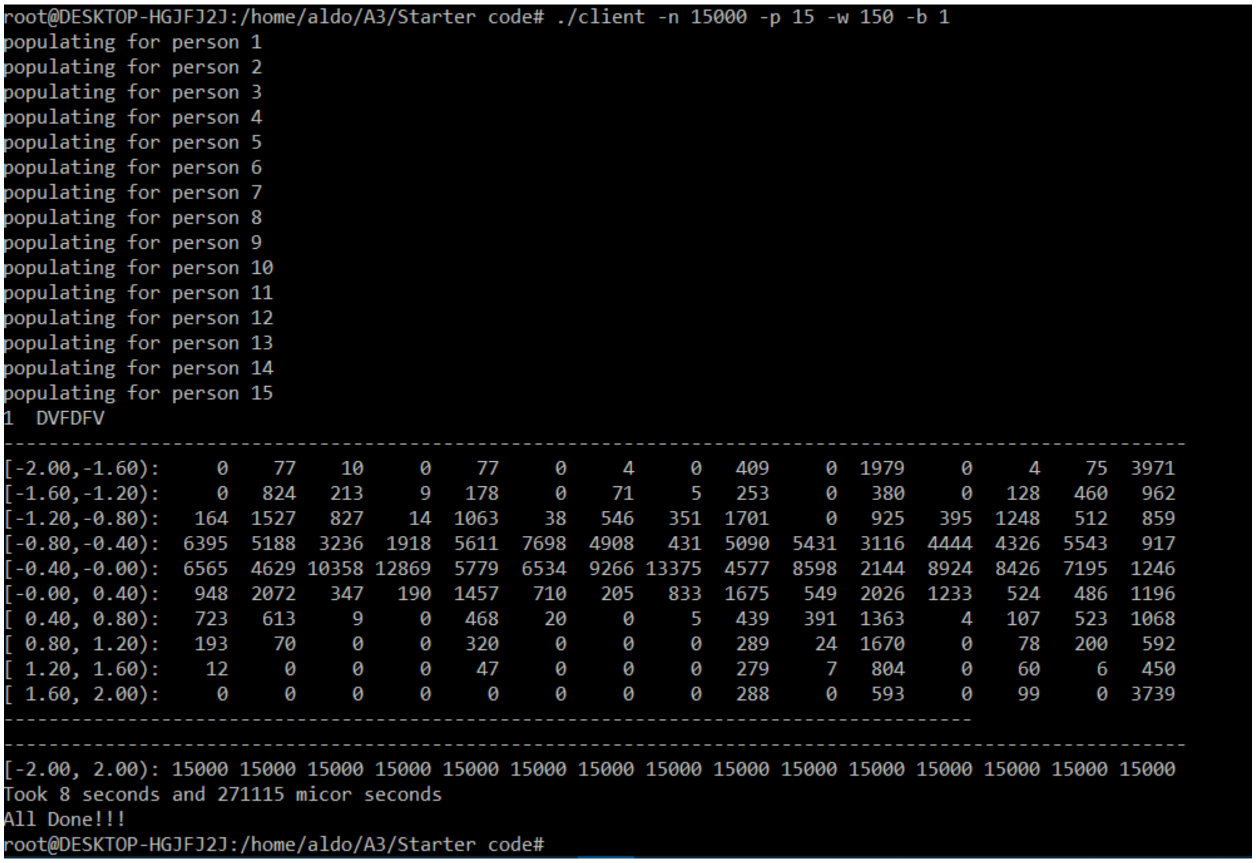
```
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# ./client -n 15000 -p 15 -w 500 -b 50
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
----------------------------------------------------------------------------------------------------------
[-2.00,-1.60):     0    77    10     0    77     0     4     0   409     0  1979     0     4    75  3971
[-1.60,-1.20):     0   824   213     9   178     0    71     5   253     0   380     0   128   460   962
[-1.20,-0.80):   164  1527   827    14  1063    38   546   351  1701     0   925   395  1248   512   859
[-0.80,-0.40):  6395  5188  3236  1918  5611  7698  4908   431  5090  5431  3116  4444  4326  5543   917
[-0.40,-0.00):  6565  4629 10358 12869  5779  6534  9266 13375  4577  8598  2144  8924  8426  7195  1246
[-0.00, 0.40):   948  2072   347   190  1457   710   205   833  1675   549  2026  1233   524   486  1196
[ 0.40, 0.80):   723   613     9     0   468    20     0     5   439   391  1363     4   107   523  1068
[ 0.80, 1.20):   193    70     0     0   320     0     0     0   289    24  1670     0    78   200   592
[ 1.20, 1.60):    12     0     0     0    47     0     0     0   279     7   804     0    60     6   450
[ 1.60, 2.00):     0     0     0     0     0     0     0     0   288     0   593     0    99     0  3739
--------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------
[-2.00, 2.00): 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000
Took 12 seconds and 389576 micor seconds
All Done!!!
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code#
```

Full Histogram with default conditions



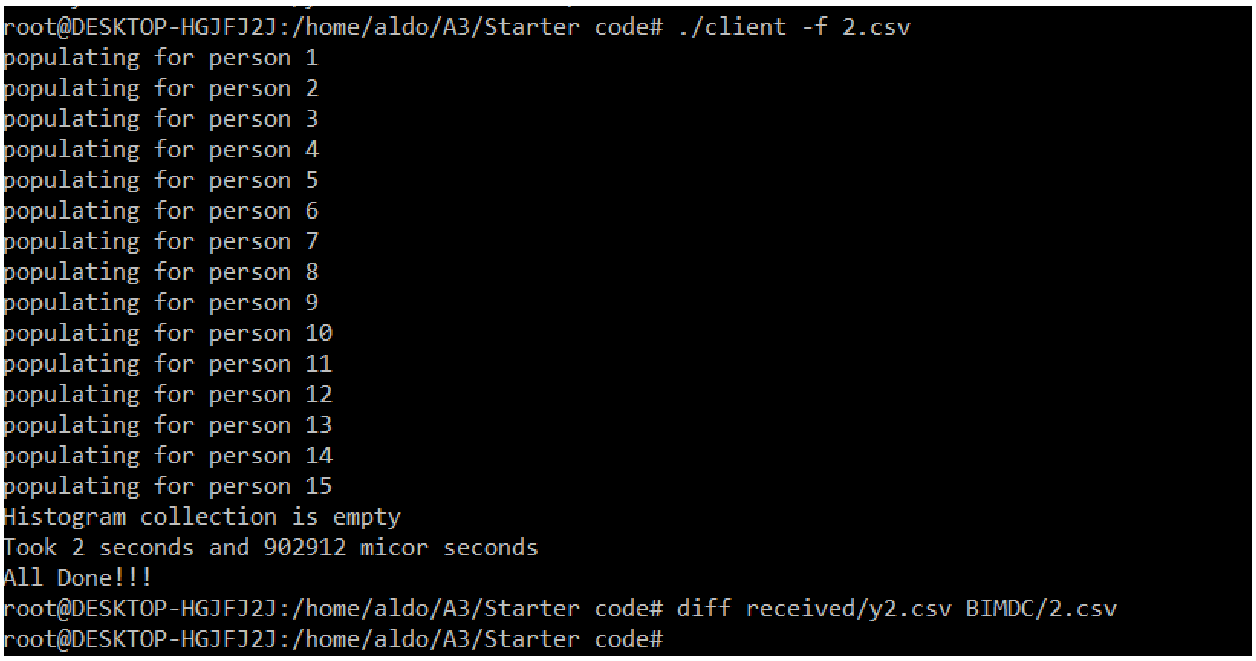
```
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# ./client -n 15000 -p 1 -b 50 -w 1
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
--------------------
[-2.00,-1.60):     0
[-1.60,-1.20):     0
[-1.20,-0.80):   164
[-0.80,-0.40):  6395
[-0.40,-0.00):  6565
[-0.00, 0.40):   948
[ 0.40, 0.80):   723
[ 0.80, 1.20):   193
[ 1.20, 1.60):    12
[ 1.60, 2.00):     0
-------------------------------------------------------------------------------------
--------------------
[-2.00, 2.00): 15000
Took 45 seconds and 820515 micor seconds
All Done!!!
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code#
```

Proof for one worker only with only 2 patients

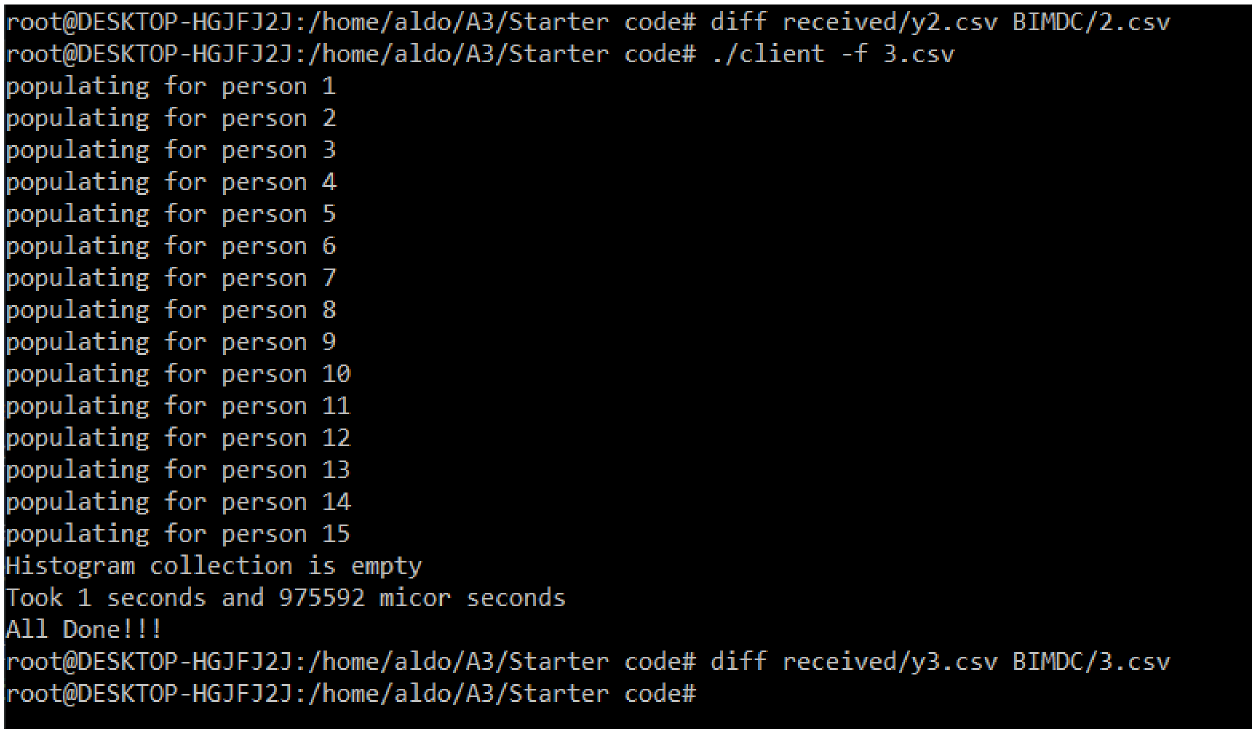


```
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# ./client -n 15000 -p 15 -w 150 -b 1
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
1  DVFDFV
------------------------------------------------------------------------------------------------------------
[-2.00,-1.60):     0    77    10     0    77     0     4     0   409     0  1979     0     4    75  3971
[-1.60,-1.20):     0   824   213     9   178     0    71     5   253     0   380     0   128   460   962
[-1.20,-0.80):   164  1527   827    14  1063    38   546   351  1701     0   925   395  1248   512   859
[-0.80,-0.40):  6395  5188  3236  1918  5611  7698  4908   431  5090  5431  3116  4444  4326  5543   917
[-0.40,-0.00):  6565  4629 10358 12869  5779  6534  9266 13375  4577  8598  2144  8924  8426  7195  1246
[-0.00, 0.40):   948  2072   347   190  1457   710   205   833  1675   549  2026  1233   524   486  1196
[ 0.40, 0.80):   723   613     9     0   468    20     0     5   439   391  1363     4   107   523  1068
[ 0.80, 1.20):   193    70     0     0   320     0     0     0   289    24  1670     0    78   200   592
[ 1.20, 1.60):    12     0     0     0    47     0     0     0   279     7   804     0    60     6   450
[ 1.60, 2.00):     0     0     0     0     0     0     0     0   288     0   593     0    99     0  3739
-------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
[-2.00, 2.00): 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000 15000
Took 8 seconds and 271115 micor seconds
All Done!!!
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code#
```

Full Histogram with buffer size 1

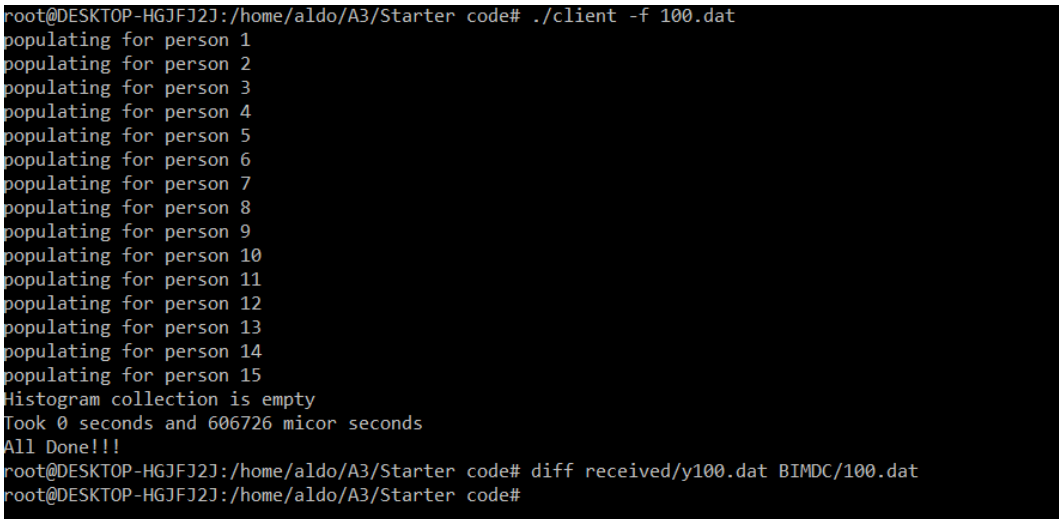


```
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# ./client -f 2.csv
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
Histogram collection is empty
Took 2 seconds and 902912 micor seconds
All Done!!!
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# diff received/y2.csv BIMDC/2.csv
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code#
```

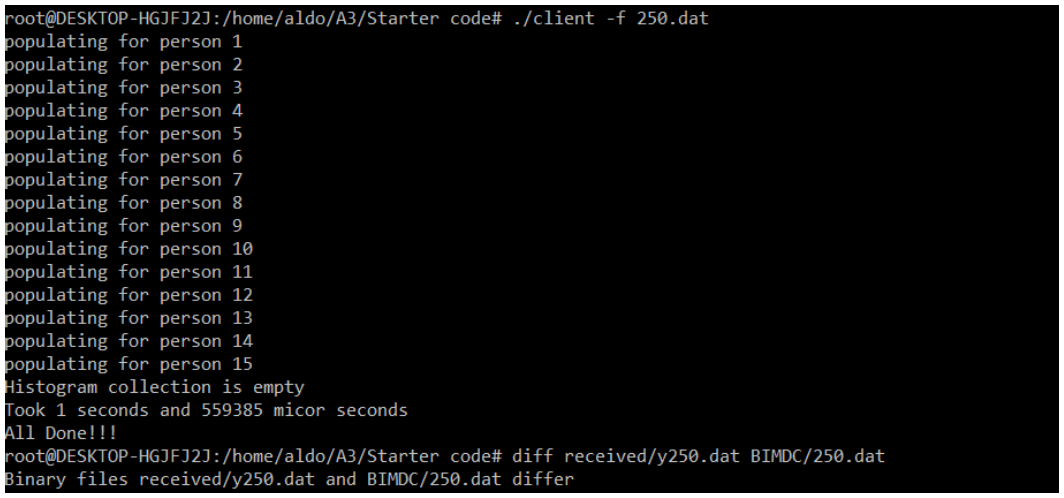


```
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# diff received/y2.csv BIMDC/2.csv
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# ./client -f 3.csv
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
Histogram collection is empty
Took 1 seconds and 975592 micor seconds
All Done!!!
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# diff received/y3.csv BIMDC/3.csv
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code#
```

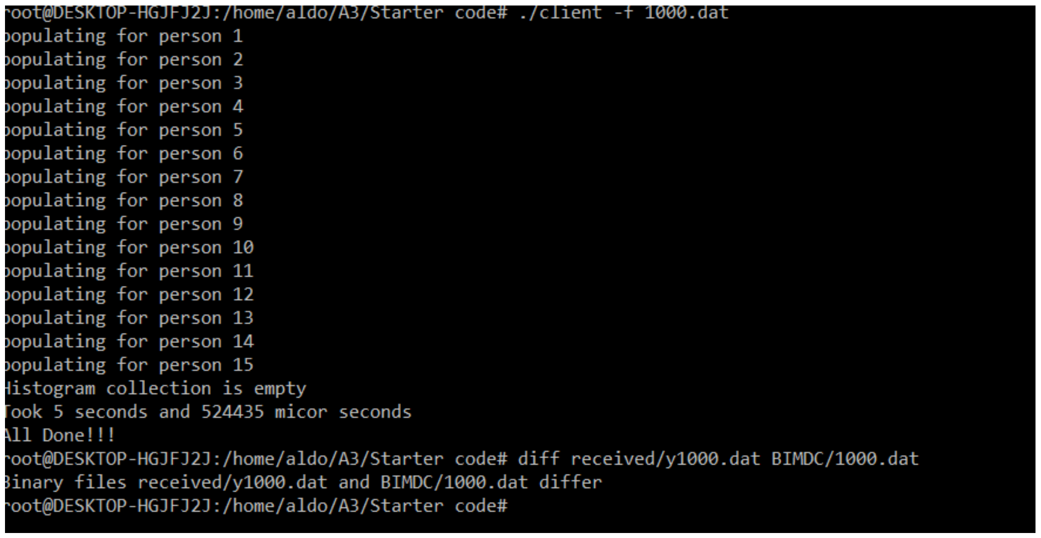
File Transfer examples, with diff check



```
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# ./client -f 100.dat
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
Histogram collection is empty
Took 0 seconds and 606726 micor seconds
All Done!!!
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# diff received/y100.dat BIMDC/100.dat
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code#
```

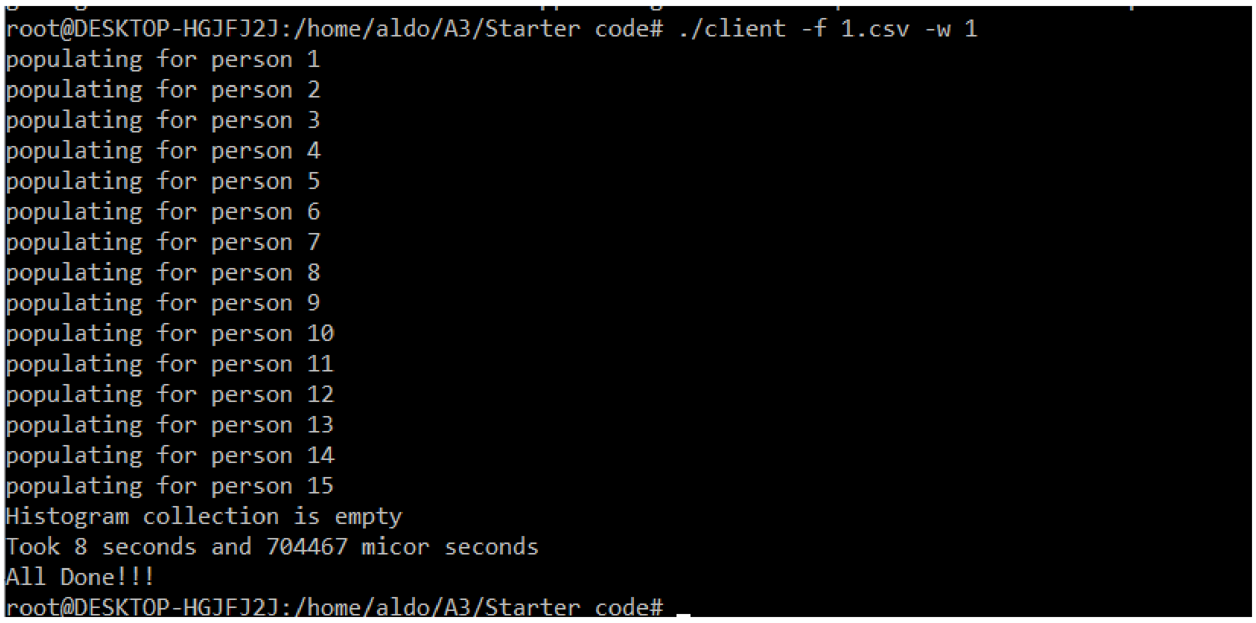


```
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# ./client -f 250.dat
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
Histogram collection is empty
Took 1 seconds and 559385 micor seconds
All Done!!!
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# diff received/y250.dat BIMDC/250.dat
Binary files received/y250.dat and BIMDC/250.dat differ
```



```
oot@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# ./client -f 1000.dat
opulating for person 1
opulating for person 2
opulating for person 3
opulating for person 4
opulating for person 5
opulating for person 6
opulating for person 7
opulating for person 8
opulating for person 9
opulating for person 10
opulating for person 11
opulating for person 12
opulating for person 13
opulating for person 14
opulating for person 15
istogram collection is empty
ook 5 seconds and 524435 micor seconds
ll Done!!!
oot@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# diff received/y1000.dat BIMDC/1000.dat
inary files received/y1000.dat and BIMDC/1000.dat differ
oot@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code#
```

.dat Files with diff Check, 100 perfect, 250 and 1000 differ, however they do show the correct size on the explorer and, are also empty as they should

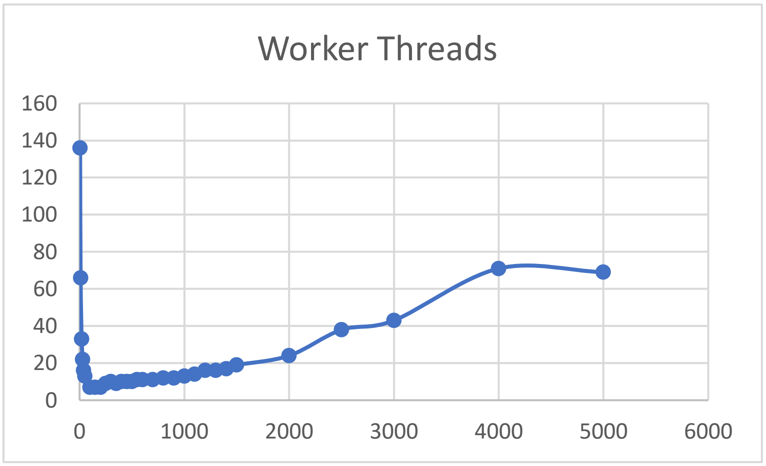


```
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# ./client -f 1.csv -w 1
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
Histogram collection is empty
Took 8 seconds and 704467 micor seconds
All Done!!!
root@DESKTOP-HGJFJ2J:/home/aldo/A3/Starter code# _
```

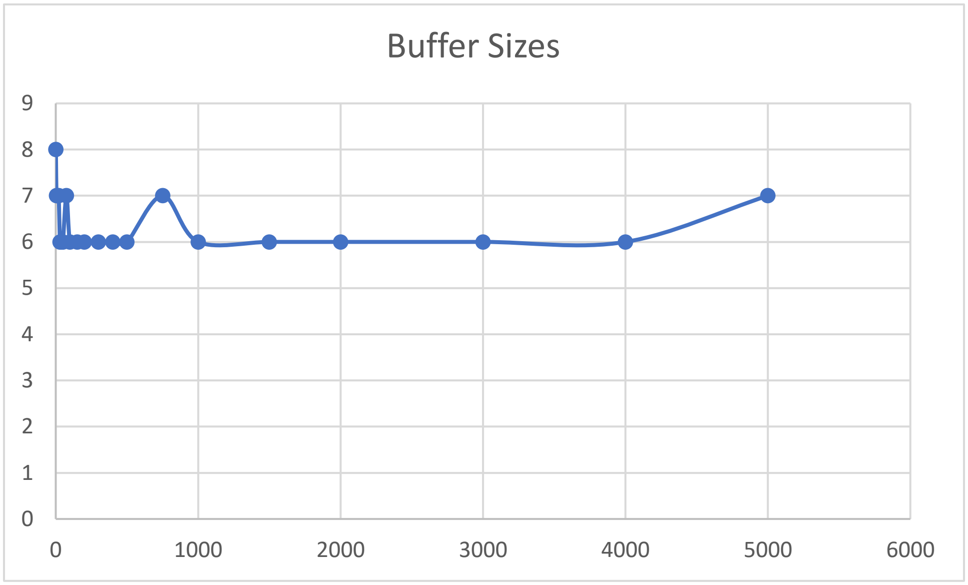
A file Transfer with just one worker

Performance and diminishing returns:

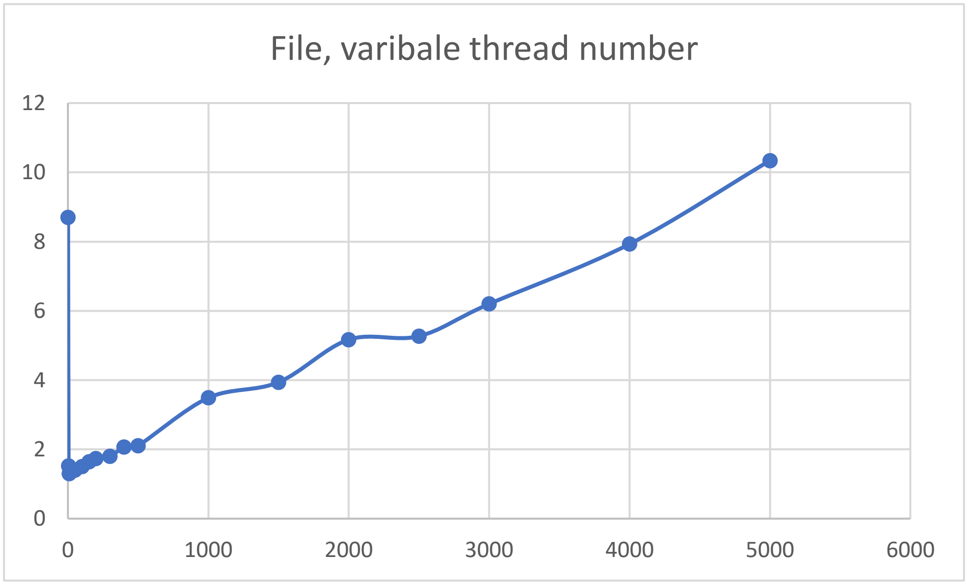
For the data transfers we can see a pretty clear performance improvement once we move of the 2 one-digit worker threads, a solid performance for the first few hundred, a clear slowdown just before the thousands and, a clear declined after a the first one thousand threads. This can be explained by context switching and file transfer speeds, As the number of threads grows, the context switching penalty starts to become apparent, with so many context switching it becomes pointless to thread, there is a clear sweet spot for this program, for 150 -300, then for 300 to 600 is almost identical and above that there’s a clear point of diminishing returns. The graph bellows contains the data just discussed



As for the buffer size I found no problem with increasing it, as long as it was above 40 the performance was pretty solid.



Finally, for the data transfers, we get the same behavior as we increase the number of worker threads, the performance slows down and becomes pointless to thread, for the exact same reason. For my code the sweet spot is around 10 worker threads.



So in conclusion the assignment was almost perfectly implemented aside from that .dat diff error, that I couldn’t find a solution both the file showed empty and with the correct size.