Documentation

Notes -

- **Files -** *(Tabulated Results)*`Report.pdf , Steps.txt, Steps.pdf, Honor-Code.pdf`
- **We have 2 lock implementations in the source package - Filter & Bakery**
- **We have added the performance measurement-related code in the Test Files namely BakeryTest.java, FilterTest.java & LockTest.java (which uses Reentrant Lock).**
- **We have our driver code in Main.java in our Test Package (as we needed access to performance measurement-related code from the test files).**
- **We will run *just* this Main.java according to the instructions given below.**
- **Specs - JDK 11**
- **In the file `netbeans-12.0/netbeans/etc/netbeans.conf` make the following change - `netbeans_jdkhome="/usr/lib/jvm/java-11-openjdk`**

Steps

1. Unzip / untar project named "`Mutex-Assignment`" and upload it in the rlogin home directory.
2. Login to rlogin interactive desktop and open netbeans (with the configs mentioned above in the 'Notes' section) & open the project in it.
3. Open terminal and run the following `taskset` command as given below before running our project to allocate 0 to 16 cores for java process -
   a. `taskset -cpa 0-16 $(pidof java)`
4. In the Test Package right click on and run the `Main.java` - there will be relevant output in std out & will generate three text files which contain the **times of executions** {1, 2, 4, 6, 8, .. 16} threads for the Three lock algorithms for different COUNT values in the format "`N:time-in-nano-sec`"
   a. `BakeryOutput.txt`
   b. `FilterOutput.txt`
   c. `JavaLockOutput.txt`
5. The project directory also contains a python script (`result.py`) which reads the above `.txt` files and calculates the stats of the performance measurements - to run this script
   a. Open terminal
   b. Navigate to the project directory - $ `cd Mutex-Assignemnt`
   c. And run the command - $ `python result.py`
6. The above step will calculate and print the required stats on the command line itself (we have also attached a screen capture of the results in our `Report.pdf` file)

(Next Page ...)

Short Explanation -

Our BakeryTest.java , FilterTest.java, LockTest.java - have a Array declared called `asList` which contains pairs of {N : COUNT} eg {4 : 5120} when we run 4 threads and test if the counter which is incremented in the critical section matches with the alloted given COUNT.
We have multiple pair corresponding to every value of N
for eg. {4 : 5120}, {4 : 10240}, {4 : 15360}
And we calculate the average execution time of critical section against number of threads N.

We also take an overall mean and standard deviation using the python script and compare the values as tabulated in the Results.pdf file.

In our main file - we execute call the Test Classes.