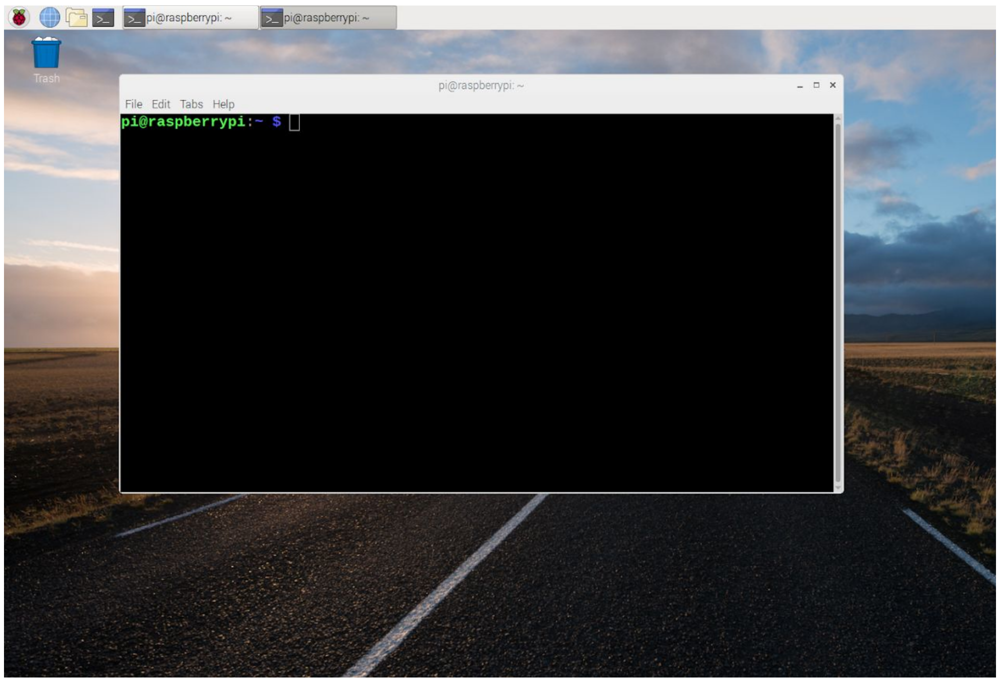
Parallel Programming Basics: Drug Design and DNA in Parallel:
Executed by Shilp Patel

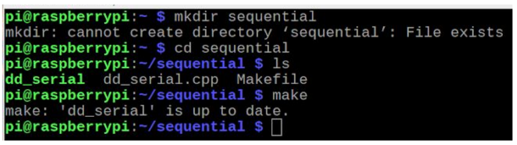
Compilation:

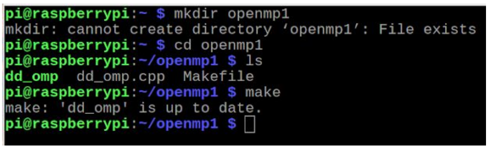
Start terminal:

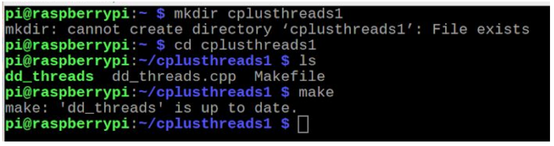


```
pi@raspberrypi:~ $ 
```

Here I opened terminal window by choosing its icon in the menu bar at the top
So in this project, the most difficult part was compiling dd_omp and dd_threads. To do that I download TBB library, PIP and OpenCV.
I used this following steps provided by this sites to install TBB:
First I installed PIP for easier approach towards installing OpenCV using steps provided by
https://www.pyimagesearch.com/2018/09/19/pip-install-opencv/
Second I installed OpenCV using steps provided by
https://circuitdigest.com/microcontroller-projects/how-to-install-python-opencv-on-raspberry-pi
Third I installed TBB Library using steps and packages provided by
https://packages.debian.org/experimental/libtbb-dev
https://github.com/abhiTronix/TBB_Raspberry_pi
https://www.theimpossiblecode.com/blog/intel-tbb-on-raspberry-pi/
The following screenshots are taken after I had finally installed and tried out the library correctly. It took me about 15 hrs to figure it out but in the end it was worth it.

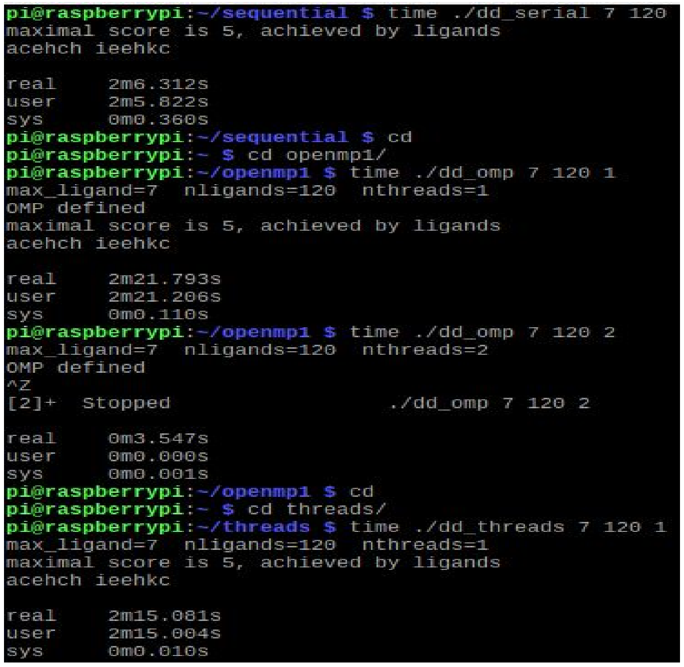
Sequential (also called Serial) solution:
I made directory named sequential.
Downloads and copied the dd_serial and Makefile provided by the professor, in to the directory. And to compile I used make command.
	
	`pi@raspberrypi:~ $ mkdir sequential`
	`mkdir: cannot create directory ‘sequential’: File exists`
	`pi@raspberrypi:~ $ cd sequential`
	`pi@raspberrypi:~/sequential $ ls`
	`dd_serial  dd_serial.cpp  Makefile`
	`pi@raspberrypi:~/sequential $ make`
	`make: 'dd_serial' is up to date.`
	`pi@raspberrypi:~/sequential $`

OpenMP solution:
I made directory named openmp1.
Downloaded and copied the dd_omp and Makefile provided by the professor, in to the directory. And to compile I used make command.
	
	`pi@raspberrypi:~ $ mkdir openmp1`
	`mkdir: cannot create directory ‘openmp1’: File exists`
	`pi@raspberrypi:~ $ cd openmp1`
	`pi@raspberrypi:~/openmp1 $ ls`
	`dd_omp  dd_omp.cpp  Makefile`
	`pi@raspberrypi:~/openmp1 $ make`
	`make: 'dd_omp' is up to date.`
	`pi@raspberrypi:~/openmp1 $`

C++11 standard threads solution:
I made directory named cplusthreads, while figuring out how to compile I had it named threads for easier approach.Downloaded and copied the dd_threads and Makefile provided by the professor, in to the directory. And to compile I used make command.
	
	`pi@raspberrypi:~ $ mkdir cplusthreads1`
	`mkdir: cannot create directory ‘cplusthreads1’: File exists`
	`pi@raspberrypi:~ $ cd cplusthreads1`
	`pi@raspberrypi:~/cplusthreads1 $ ls`
	`dd_threads  dd_threads.cpp  Makefile`
	`pi@raspberrypi:~/cplusthreads1 $ make`
	`make: 'dd_threads' is up to date.`
	`pi@raspberrypi:~/cplusthreads1 $`

Measure Run-Time:

Part A



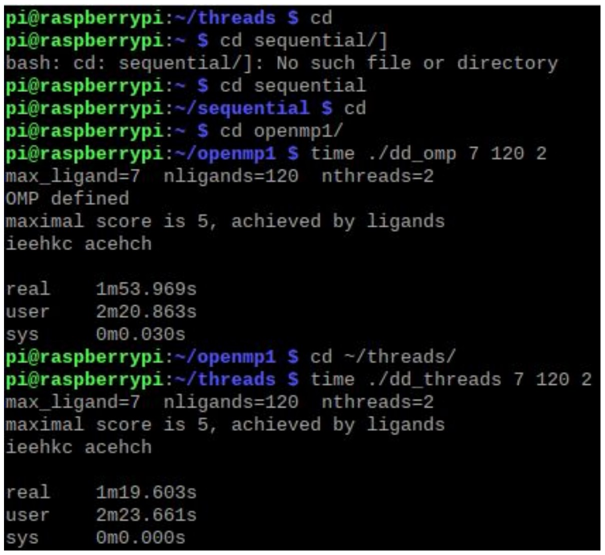
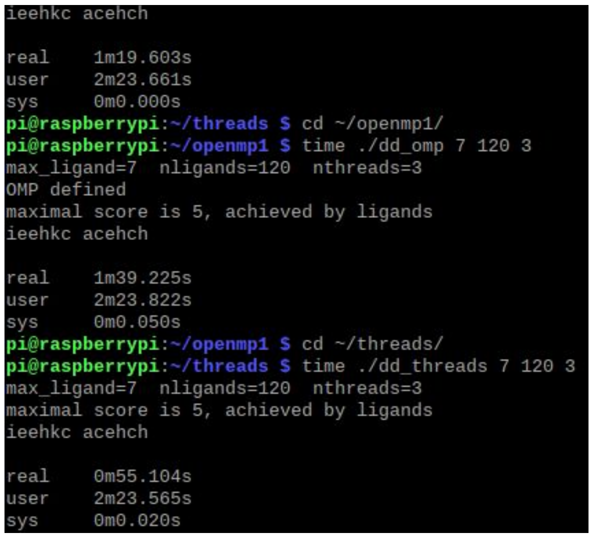
```
pi@raspberrypi:~/sequential $ time ./dd_serial 7 120
maximal score is 5, achieved by ligands
acehch ieehkc

real    2m6.312s
user    2m5.822s
sys     0m0.360s
pi@raspberrypi:~/sequential $ cd
pi@raspberrypi:~ $ cd openmp1/
pi@raspberrypi:~/openmp1 $ time ./dd_omp 7 120 1
max_ligand=7  nligands=120  nthreads=1
OMP defined
maximal score is 5, achieved by ligands
acehch ieehkc

real    2m21.793s
user    2m21.206s
sys     0m0.110s
pi@raspberrypi:~/openmp1 $ time ./dd_omp 7 120 2
max_ligand=7  nligands=120  nthreads=2
OMP defined
^Z
[2]+  Stopped                 ./dd_omp 7 120 2

real    0m3.547s
user    0m0.000s
sys     0m0.001s
pi@raspberrypi:~/openmp1 $ cd
pi@raspberrypi:~ $ cd threads/
pi@raspberrypi:~/threads $ time ./dd_threads 7 120 1
max_ligand=7  nligands=120  nthreads=1
maximal score is 5, achieved by ligands
acehch ieehkc

real    2m15.081s
user    2m15.004s
sys     0m0.010s
```

▣ Here I measured the running time of each implementation, using the time command and observing the real time for each solution. In the command I also included thread to be 1 and the default arguments for each value.

Results:

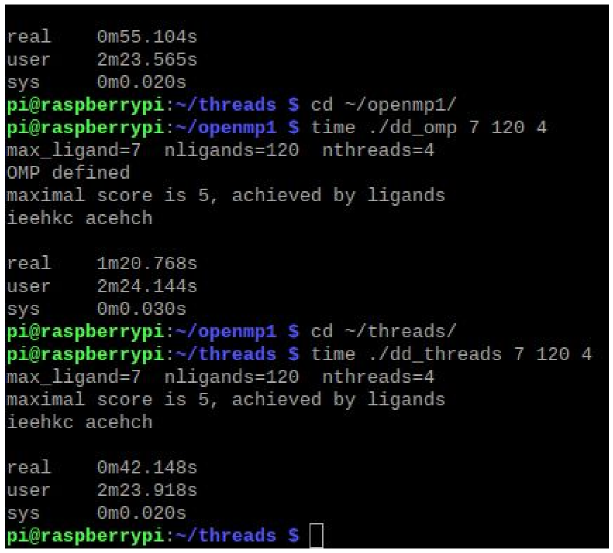
Implementation	Time(s)
dd_serial	126.312
dd_openmp	141.793
dd_threads	135.081

Part B:

2 Threads: Here I measured the running time of dd_omp and dd_threads implementation, using the time command and observing the real time for each solution. In the command I also included thread to be 2 and the default arguments for each value.	
3 Threads: Here I measured the running time of dd_omp and dd_threads implementation, using the time command and observing the real time for each solution. In the command I also included thread to be 3 and the default arguments for each value.	

```
pi@raspberrypi:~/threads $ cd
pi@raspberrypi:~ $ cd sequential/]
bash: cd: sequential/]: No such file or directory
pi@raspberrypi:~ $ cd sequential
pi@raspberrypi:~/sequential $ cd
pi@raspberrypi:~ $ cd openmp1/
pi@raspberrypi:~/openmp1 $ time ./dd_omp 7 120 2
max_ligand=7  nligands=120  nthreads=2
OMP defined
maximal score is 5, achieved by ligands
ieehkc acehch

real    1m53.969s
user    2m20.863s
sys     0m0.030s
pi@raspberrypi:~/openmp1 $ cd ~/threads/
pi@raspberrypi:~/threads $ time ./dd_threads 7 120 2
max_ligand=7  nligands=120  nthreads=2
maximal score is 5, achieved by ligands
ieehkc acehch

real    1m19.603s
user    2m23.661s
sys     0m0.000s
```

```
ieehkc acehch

real    1m19.603s
user    2m23.661s
sys     0m0.000s
pi@raspberrypi:~/threads $ cd ~/openmp1/
pi@raspberrypi:~/openmp1 $ time ./dd_omp 7 120 3
max_ligand=7  nligands=120  nthreads=3
OMP defined
maximal score is 5, achieved by ligands
ieehkc acehch

real    1m39.225s
user    2m23.822s
sys     0m0.050s
pi@raspberrypi:~/openmp1 $ cd ~/threads/
pi@raspberrypi:~/threads $ time ./dd_threads 7 120 3
max_ligand=7  nligands=120  nthreads=3
maximal score is 5, achieved by ligands
ieehkc acehch

real    0m55.104s
user    2m23.565s
sys     0m0.020s
```

4 Threads: Here I measured the running time of dd_omp and dd_threads implementation, using the time command and observing the real time for each solution. In the command I also included thread to be 4 and the default arguments for each value.	

```
real    0m55.104s
user    2m23.565s
sys     0m0.020s
pi@raspberrypi:~/threads $ cd ~/openmp1/
pi@raspberrypi:~/openmp1 $ time ./dd_omp 7 120 4
max_ligand=7  nligands=120  nthreads=4
OMP defined
maximal score is 5, achieved by ligands
ieehkc acehch

real    1m20.768s
user    2m24.144s
sys     0m0.030s
pi@raspberrypi:~/openmp1 $ cd ~/threads/
pi@raspberrypi:~/threads $ time ./dd_threads 7 120 4
max_ligand=7  nligands=120  nthreads=4
maximal score is 5, achieved by ligands
ieehkc acehch

real    0m42.148s
user    2m23.918s
sys     0m0.020s
pi@raspberrypi:~/threads $ 
```

Results:

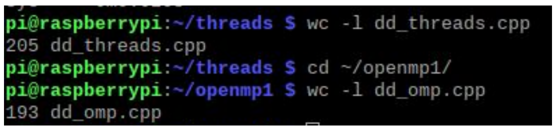
Implementation	Time(s) 2 Threads	Time(s) 3 Threads	Time(s) 4 Threads
dd_omp	113.969	99.225	80.768
dd_threads	79.603	55.104	42.148

Questions:

1. Which approach is fastest?

In part A dd_serial has the fastest approach; And in part B dd_thread with 4 threads has the fastest approach. The time depends on the numbers of threads that is used to run the program.

2. Determine the number of lines in each file (use wc -l). How does the C++11 implementation compare to the OpenMP implementations?

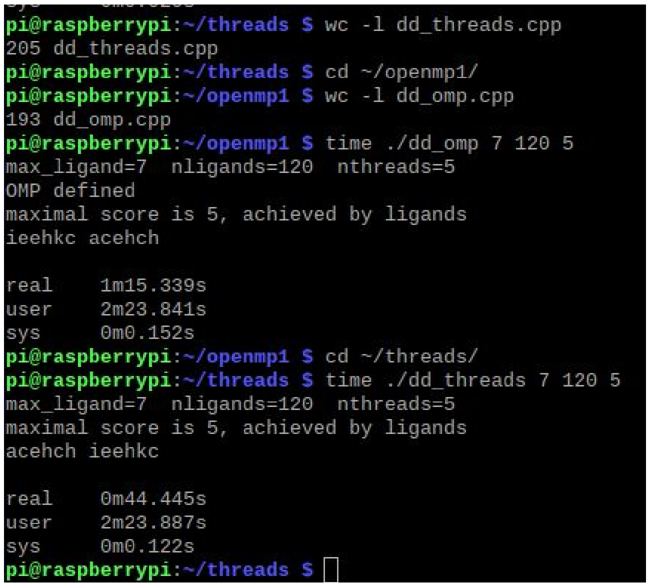


```
pi@raspberrypi:~/threads $ wc -l dd_threads.cpp
205 dd_threads.cpp
pi@raspberrypi:~/threads $ cd ~/openmp1/
pi@raspberrypi:~/openmp1 $ wc -l dd_omp.cpp
193 dd_omp.cpp
```

▣ As shown above in the screenshot provided, The word count of dd_omp is 193 and word count of dd_threads is 205. The difference between C++11 implementation and the OpenMP implementations is 12 word count difference and as a result dd_threads has faster approach than dd_omp.

3. Increase the number of threads to 5 threads. What is the runtime for each?

Implementation	Time(s) 5 Threads
dd_omp	75.339
dd_threads	44.445



```
pi@raspberrypi:~/threads $ wc -l dd_threads.cpp
205 dd_threads.cpp
pi@raspberrypi:~/threads $ cd ~/openmp1/
pi@raspberrypi:~/openmp1 $ wc -l dd_omp.cpp
193 dd_omp.cpp
pi@raspberrypi:~/openmp1 $ time ./dd_omp 7 120 5
max_ligand=7  nligands=120  nthreads=5
OMP defined
maximal score is 5, achieved by ligands
ieehkc acehch

real    1m15.339s
user    2m23.841s
sys     0m0.152s
pi@raspberrypi:~/openmp1 $ cd ~/threads/
pi@raspberrypi:~/threads $ time ./dd_threads 7 120 5
max_ligand=7  nligands=120  nthreads=5
maximal score is 5, achieved by ligands
acehch ieehkc

real    0m44.445s
user    2m23.887s
sys     0m0.122s
pi@raspberrypi:~/threads $
```

▣ Here I measured the running time of dd_omp and dd_threads implementation, using the time command and observing the real run time for each solution. In the command I used 5 thread and the default arguments for each value.

4. Increase the maximum ligand length to 7, and rerun each program. What is the run time for each?

Run time for max_ligand = 7 :

Implementation	Time(s)
dd_serial	126.312
dd_openmp	141.793
dd_thread	135.081