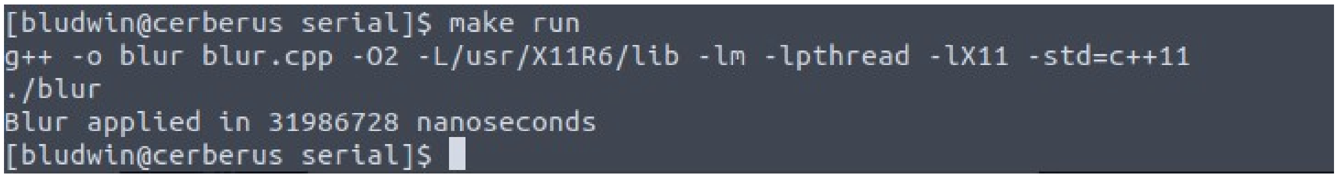
Lab Assignment 1

The serial and parallel implementations are both located in their respective folders in the /ludwin/lab_assignment_1 directory. Both sets of code can be compiled and ran using the command shown below.

To run:
make run

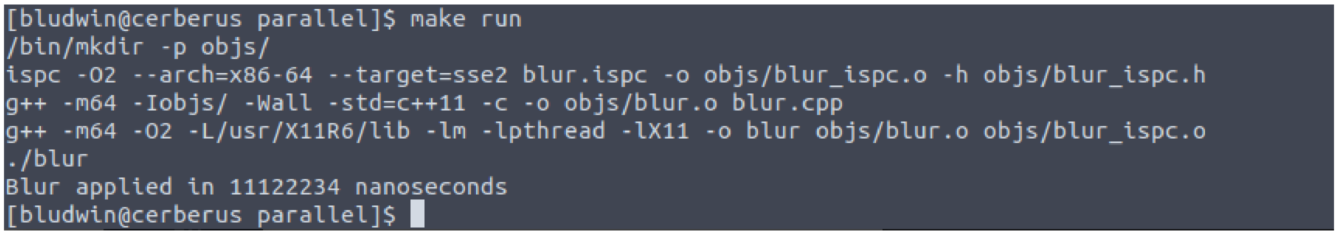
The timing information will be printed after the compilation is completed.

Serial timing:



```
[bludwin@cerberus serial]$ make run
g++ -o blur blur.cpp -O2 -L/usr/X11R6/lib -lm -lpthread -lX11 -std=c++11
./blur
Blur applied in 31986728 nanoseconds
[bludwin@cerberus serial]$ 
```

Parallel timing:



```
[bludwin@cerberus parallel]$ make run
/bin/mkdir -p objs/
ispc -O2 --arch=x86-64 --target=sse2 blur.ispc -o objs/blur_ispc.o -h objs/blur_ispc.h
g++ -m64 -Iobjs/ -Wall -std=c++11 -c -o objs/blur.o blur.cpp
g++ -m64 -O2 -L/usr/X11R6/lib -lm -lpthread -lX11 -o blur objs/blur.o objs/blur_ispc.o
./blur
Blur applied in 11122234 nanoseconds
[bludwin@cerberus parallel]$ 
```

I was able to achieve a ~3x speed up when comparing my parallel implementation compared to the serial version. At first I modified the R, G, and B into their respective _mod arrays one at a time and was still able to achieve above 2x speed up. However, after testing I was able to achieve a higher speed up by modifying my ispc function to modify each array at the same time without multiple function calls. By reading the pixels into a buffer larger than the original image, I was able to avoid if statements that handle edge cases in my code. This way I was able to index through the valid pixel locations without having to include any if statements that reduce the efficiency of ispc.