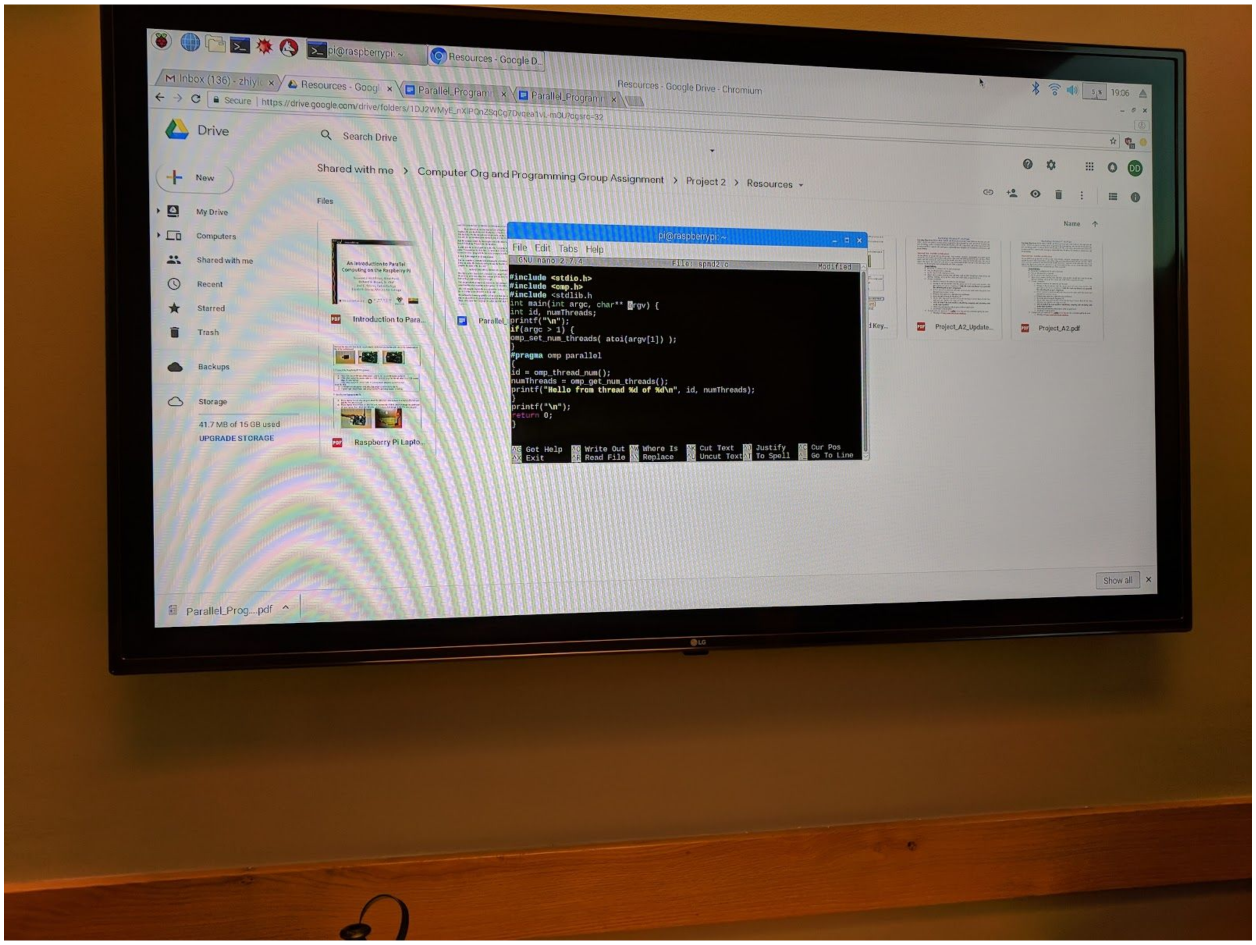
Raspberry pi Code



```
GNU nano 2.7.4                File: spmd2.c                    Modified

#include <stdio.h>
#include <omp.h>
#include <stdlib.h
int main(int argc, char** argv) {
int id, numThreads;
printf("\n");
if(argc > 1) {
omp_set_num_threads( atoi(argv[1]) );
}
#pragma omp parallel
{
id = omp_thread_num();
numThreads = omp_get_num_threads();
printf("Hello from thread %d of %d\n", id, numThreads);
}
printf("\n");
return 0;
}

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Here is the copied code that was given from the project description. In this code we're supposed to exam "spmd2.c". It is a C program that uses special additions to the language that make it easy to run a portion of your program on multiple threads on the different cores of a multicore machine. We complete the task with the following actions.

We first created the file by typing the following codes in a terminal window:

```
nano spmd2.c
```

We then wrote the file with the following codes:

```
1.#include <stdio.h>
2.#include <omp.h>
3.#include <stdlib.h>
4.int main(int argc, char** argv) {
5.int id, numThreads;
6.printf("\n");
7.if (argc > 1) {
8.omp_set_num_threads( atoi(argv[1]) );
9.}
#pragma omp parallel
{
id = omp_get_thread_num();
numThreads = omp_get_num_threads();
printf("Hello from thread %d of %d\n", id, numThreads);
}
printf("\n");
return 0;
}
```

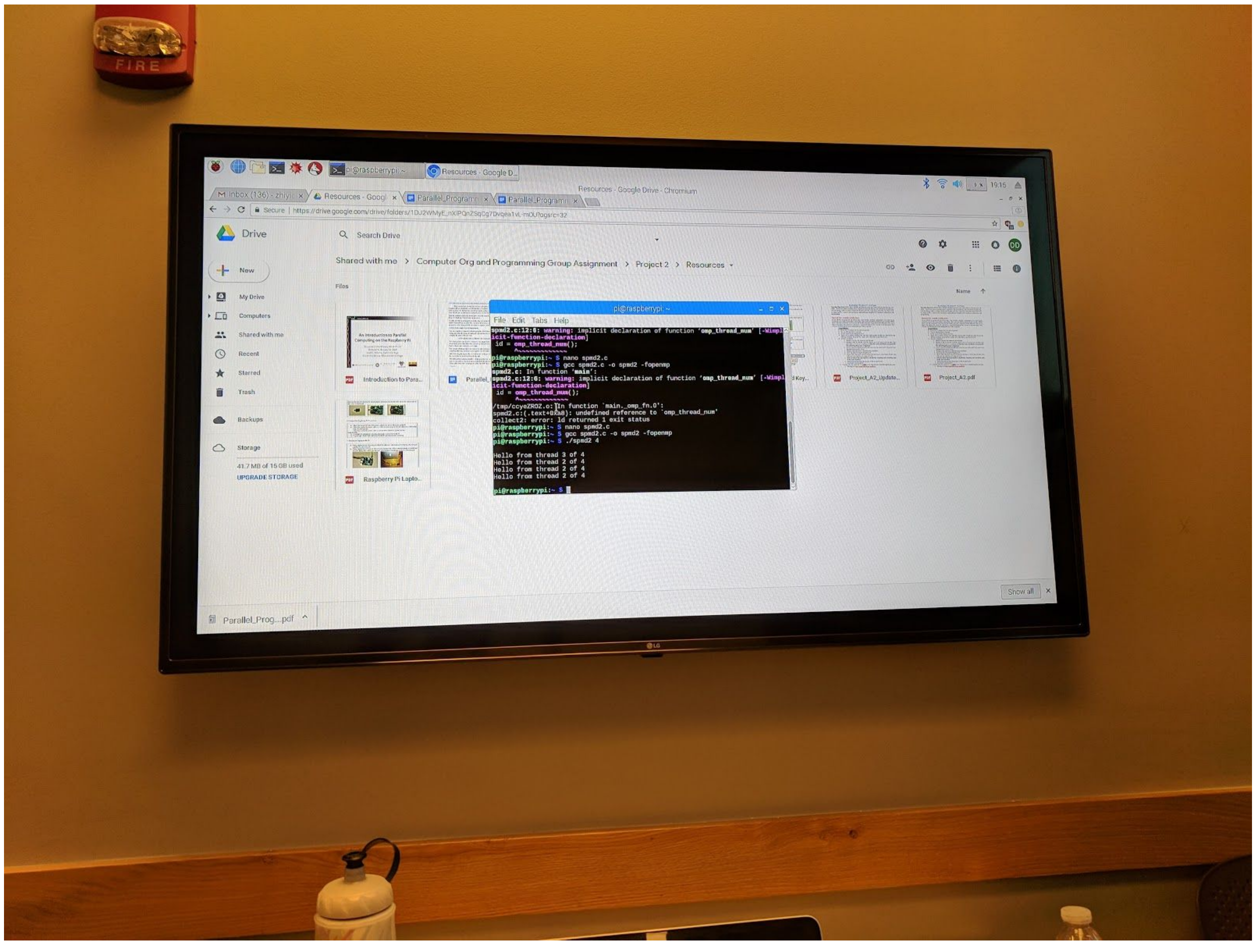
After saving and exiting from nano, we made the executable program by typing:

```
gcc  spmd2.c -o spmd2 -fopenmp
```

And this should create a file called spmd2, which is the executable program. To run the program, we typed:

```
./spmd2 4
```

Dasdas



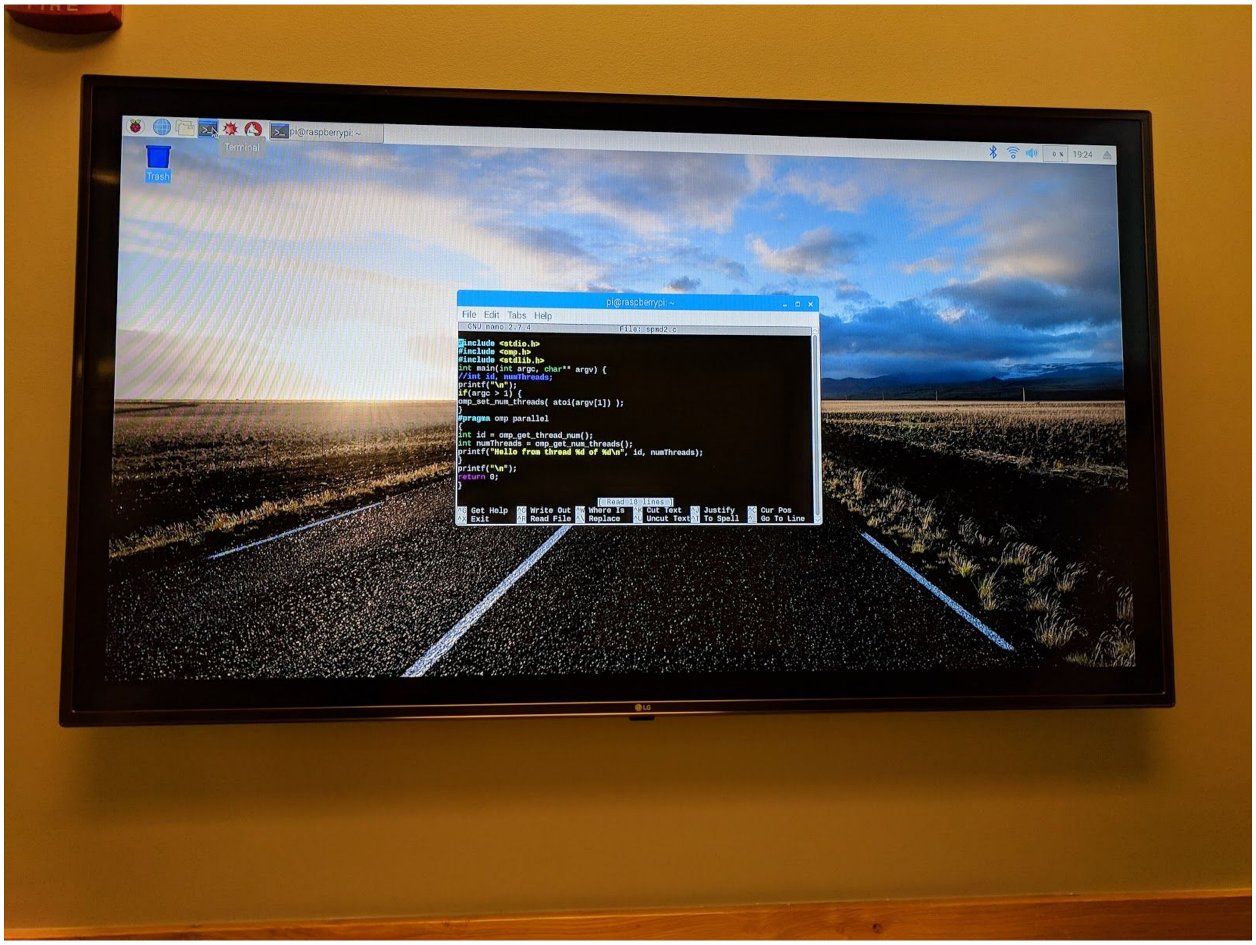
```
spmd2.c:12:6: warning: implicit declaration of function 'omp_thread_num' [-Wimpl
icit-function-declaration]
 id = omp_thread_num();
      ^~~~~~~~~~~~~~
pi@raspberrypi:~ $ nano spmd2.c
pi@raspberrypi:~ $ gcc spmd2.c -o spmd2 -fopenmp
spmd2.c: In function 'main':
spmd2.c:12:6: warning: implicit declaration of function 'omp_thread_num' [-Wimpl
icit-function-declaration]
 id = omp_thread_num();
      ^~~~~~~~~~~~~~
/tmp/ccyeZROZ.o: In function `main._omp_fn.0':
spmd2.c:(.text+0xa8): undefined reference to `omp_thread_num'
collect2: error: ld returned 1 exit status
pi@raspberrypi:~ $ nano spmd2.c
pi@raspberrypi:~ $ gcc spmd2.c -o spmd2 -fopenmp
pi@raspberrypi:~ $ ./spmd2 4

Hello from thread 3 of 4
Hello from thread 2 of 4
Hello from thread 2 of 4
Hello from thread 2 of 4

pi@raspberrypi:~ $
```

The 4 is called a command-line argument that indicates how many threads to fork. Since Raspberry Pi has a 4-core processor, it is natural to try 4 threads. It is fine if the thread is shown not in a natural order (When the thread finishes can not be guaranteed), but the output should be different threads shown. As as shown in the picture above, some threads appeared multiple times.

Dasd



```
GNU nano 2.7.4                    File: spmd2.c

#include <stdio.h>
#include <omp.h>
#include <stdlib.h>
int main(int argc, char** argv) {
//int id, numThreads;
printf("\n");
if(argc > 1) {
omp_set_num_threads( atoi(argv[1]) );
}
#pragma omp parallel
{
int id = omp_get_thread_num();
int numThreads = omp_get_num_threads();
printf("Hello from thread %d of %d\n", id, numThreads);
}
printf("\n");
return 0;
}

                    [ Read 18 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

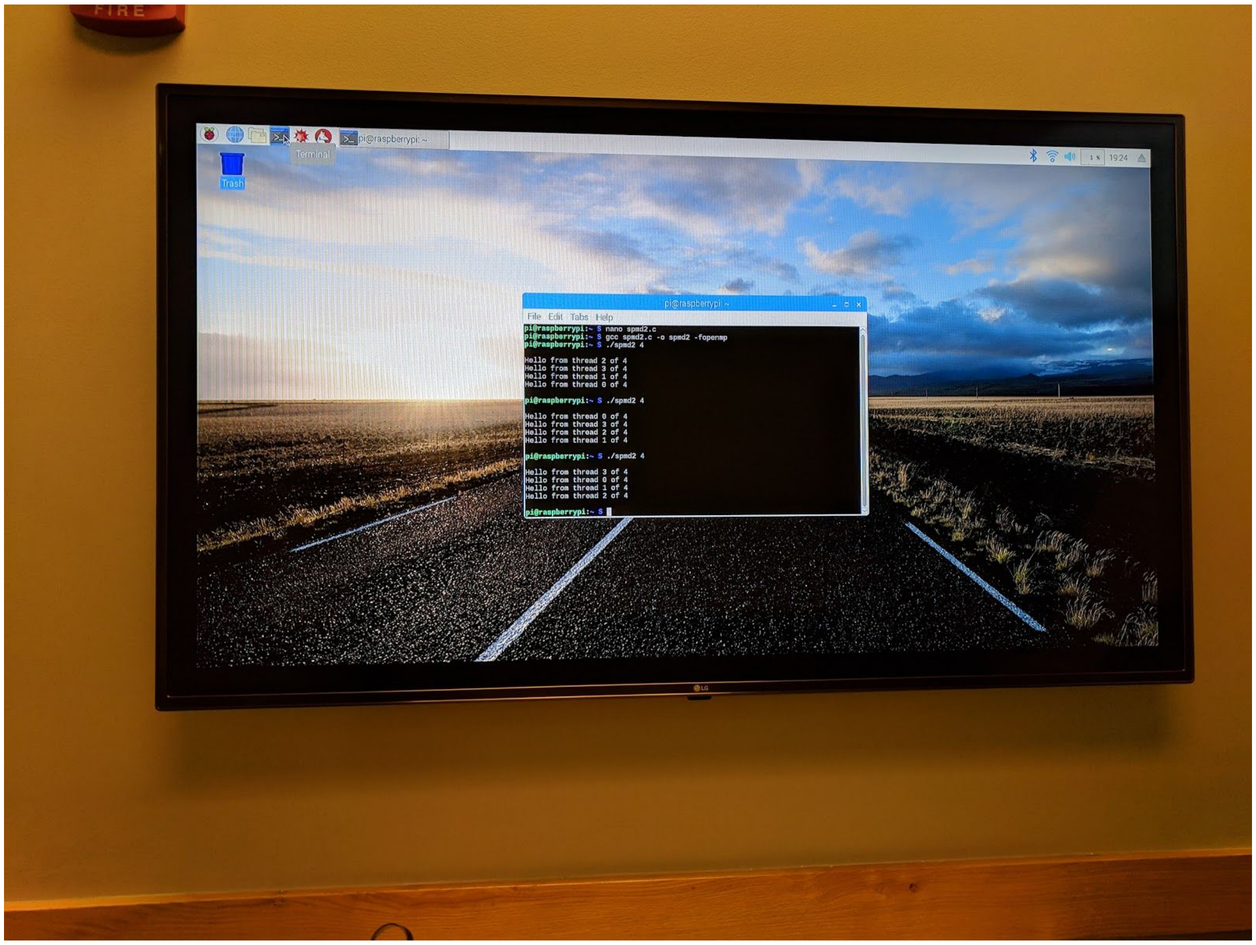
So we did the fix with the following codes:

5.//int id, numThreads;

12.int id = omp_get_thread_num();

13.int numThreads = omp_get_num_threads();

Asdasd



```
pi@raspberrypi:~ $ nano spmd2.c
pi@raspberrypi:~ $ gcc spmd2.c -o spmd2 -fopenmp
pi@raspberrypi:~ $ ./spmd2 4

Hello from thread 2 of 4
Hello from thread 3 of 4
Hello from thread 1 of 4
Hello from thread 0 of 4

pi@raspberrypi:~ $ ./spmd2 4

Hello from thread 0 of 4
Hello from thread 3 of 4
Hello from thread 2 of 4
Hello from thread 1 of 4

pi@raspberrypi:~ $ ./spmd2 4

Hello from thread 3 of 4
Hello from thread 0 of 4
Hello from thread 1 of 4
Hello from thread 2 of 4

pi@raspberrypi:~ $
```

And, the result is corrected. The fix worked by saying in the code that each thread will now have its own private copy of the variables named “id” and “numThread” (line 12&13).