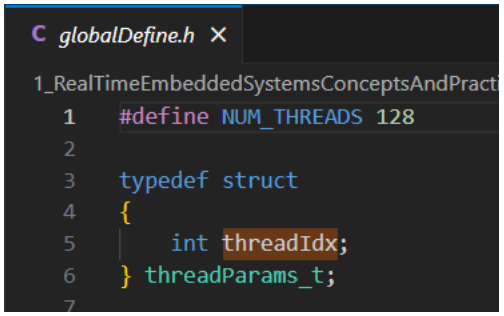
# Multiple Threads

- The code has 2 files:
  pthread.c: logic to implement the threads.
- globalDefines.h: file with all generic definitions.

- Pthread file descripction:
- From line 1 to 7 are all file included.
- Line 10 an 11 are the thread and thread parameters

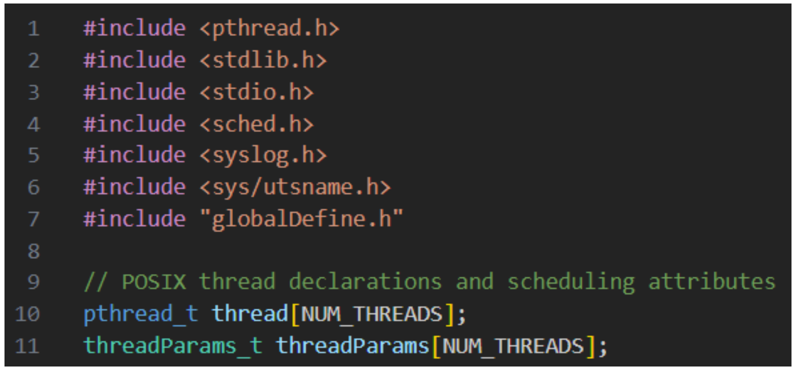


globalDefine.h

1_RealTimeEmbeddedSystemsConceptsAndPract

```c
#define NUM_THREADS 128

typedef struct
{
    int threadIdx;
} threadParams_t;
```

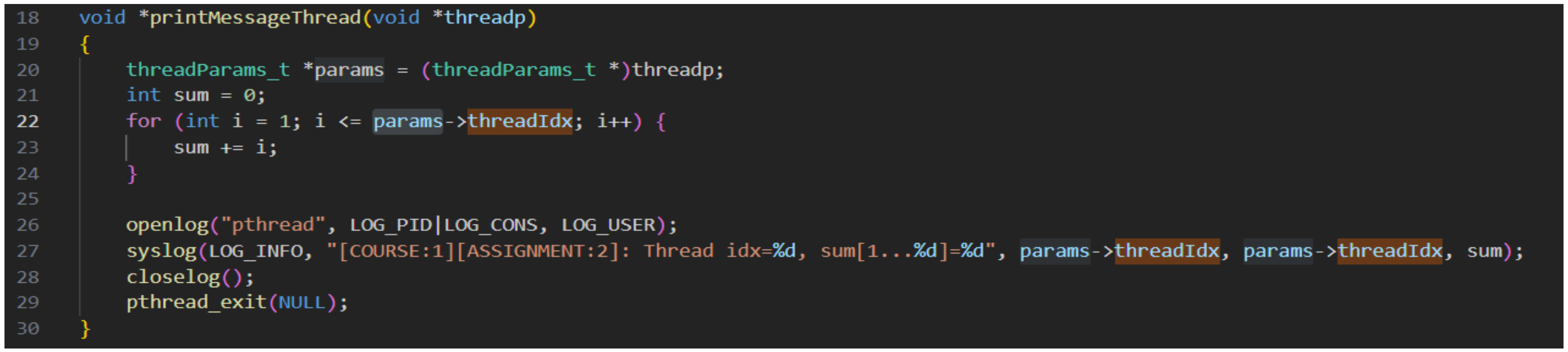


```c
1   #include <pthread.h>
2   #include <stdlib.h>
3   #include <stdio.h>
4   #include <sched.h>
5   #include <syslog.h>
6   #include <sys/utsname.h>
7   #include "globalDefine.h"
8
9   // POSIX thread declarations and scheduling attributes
10  pthread_t thread[NUM_THREADS];
11  threadParams_t threadParams[NUM_THREADS];
```

# Multiple Threads

From line 18 to 30 is the “printMessageThread” function implementation that has a threadp void pointer, in this implementation de argument is  needed to know the thread index.

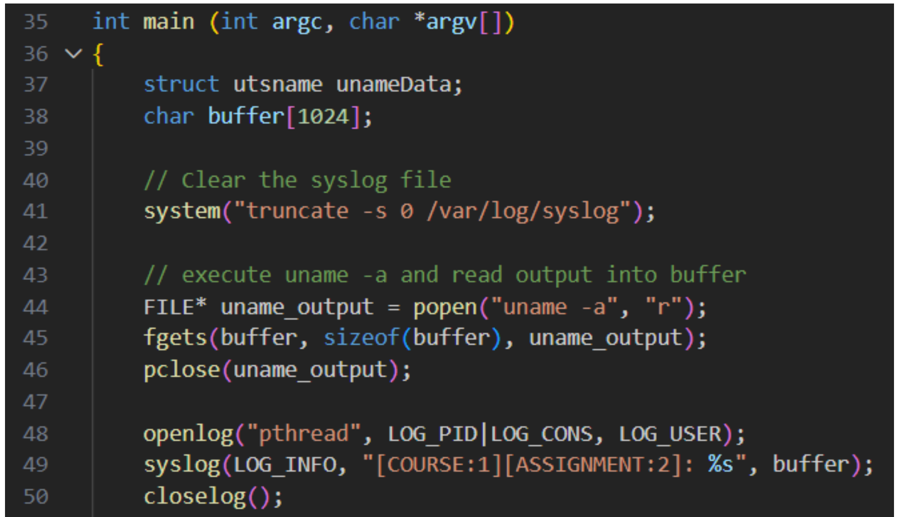
From line 22 to 24 a sum is done from 1 to the thread index, after thar from line26 to 28 the thread information is written in the syslog file.



```
18  void *printMessageThread(void *threadp)
19  {
20      threadParams_t *params = (threadParams_t *)threadp;
21      int sum = 0;
22      for (int i = 1; i <= params->threadIdx; i++) {
23          sum += i;
24      }
25
26      openlog("pthread", LOG_PID|LOG_CONS, LOG_USER);
27      syslog(LOG_INFO, "[COURSE:1][ASSIGNMENT:2]: Thread idx=%d, sum[1...%d]=%d", params->threadIdx, params->threadIdx, sum);
28      closelog();
29      pthread_exit(NULL);
30  }
```

# Multiple Threads

- From line 35 to 66 the main function is implemented
- Line 37 and 38 are the variables to write the “uname -a” information.
- Line 41 is to clean the syslog file.
- From line 44 to 50 we get the uname -a information and after that the information is written in the first syslog line file.

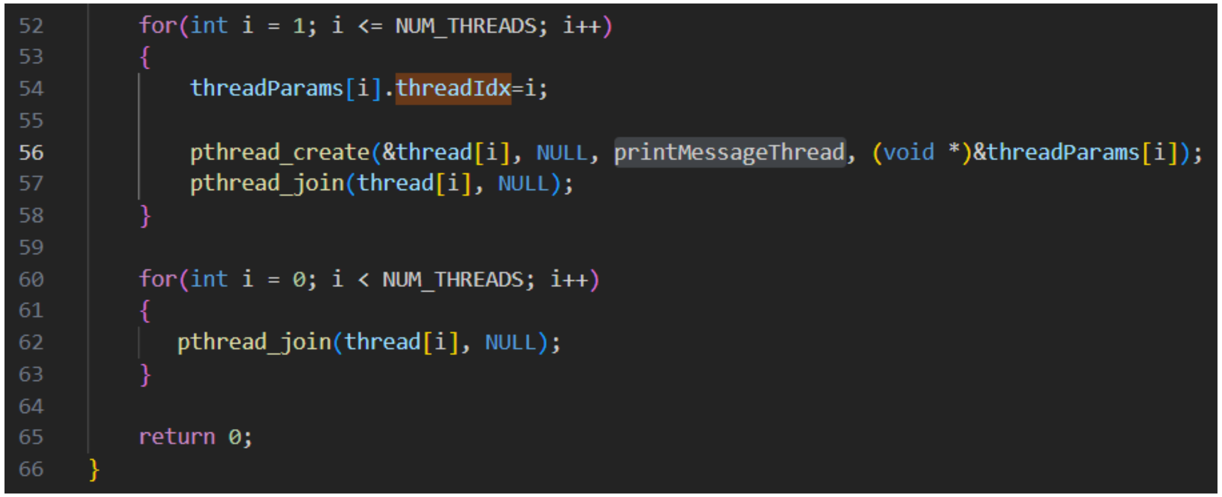


```
35  int main (int argc, char *argv[])
36  {
37      struct utsname unameData;
38      char buffer[1024];
39
40      // Clear the syslog file
41      system("truncate -s 0 /var/log/syslog");
42
43      // execute uname -a and read output into buffer
44      FILE* uname_output = popen("uname -a", "r");
45      fgets(buffer, sizeof(buffer), uname_output);
46      pclose(uname_output);
47
48      openlog("pthread", LOG_PID|LOG_CONS, LOG_USER);
49      syslog(LOG_INFO, "[COURSE:1][ASSIGNMENT:2]: %s", buffer);
50      closelog();
```

# Multiple Threads

From line 52 to 58 a for is implemented to crate the threads, to do that we use pthread_create with printMessageThread function in the argument to write the phreat information and sum index.

From line 60 to 63 the execute waits until all theads finish.



```
52      for(int i = 1; i <= NUM_THREADS; i++)
53      {
54          threadParams[i].threadIdx=i;
55
56          pthread_create(&thread[i], NULL, printMessageThread, (void *)&threadParams[i]);
57          pthread_join(thread[i], NULL);
58      }
59
60      for(int i = 0; i < NUM_THREADS; i++)
61      {
62          pthread_join(thread[i], NULL);
63      }
64
65      return 0;
66  }
```