Assignment 9

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/syscall.h>
#include <pthread.h>
#include <time.h>
#define MAXN 5
pthread_mutex_t mutex;
pthread_cond_t cond;
int start = 0;
void* func() {
    pthread_mutex_lock(&mutex);
    while(start == 0) {
        pthread_cond_wait(&cond, &mutex);
    }
    pthread_cond_signal(&cond);
    printf("Thread %llu running\n", pthread_self());
    pthread_mutex_unlock(&mutex);
    return NULL;
}
int main() {
    pthread_cond_init(&cond, NULL);
    pthread_mutex_init(&mutex, NULL);
    pthread_t threads[MAXN];
    for(int i = 0; i < MAXN; i++) {
        printf("Starting thread %d\n", i);
        if(pthread_create(&threads[i], NULL, func, NULL) != 0) {
            printf("Fail on create thread %d\n", i);
        }
    }
    start = 1;
    for(int i = 0; i < MAXN; i++) {
        if(pthread_join(threads[i], NULL) != 0) {
            printf("Fail on joining thread %d\n", i);
            exit(-1);
        }
    }
    pthread_cond_destroy(&cond);
    pthread_mutex_destroy(&mutex);
    return 0;
}
```

- Line 8
  Number of Threads.
- Line 28 to Line 34
  (1) Use pthread to create 5 threads and print “Starting thread i” for each thread before calling the thread-create function.
- Line 19
  (2) This is the barrier that uses synchronization primitives to block the threads and await the completion of the creation process for all threads.
- Line 20
  (3) Print “Thread # running” after calling the barrier.