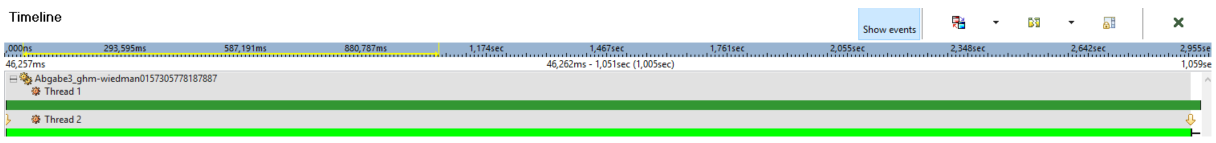
Abgabe 3

```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <pthread.h>
#include <sched.h>
#include <string.h>

#define MILLION  1E6


/*
 * Operates for one millisecond
 */
void oneSecMethod() {
  volatile int zero = 0, one = 1, sum;
  int i;
  for (i = 0; i < 50000; i++) {
    sum = zero + one;
  }
  for (i = 0; i < 50000; i++) {
    sum = zero + one;
  }
}

/*
 * Waste the time given in milliseconds
 */
void waste_msecs(unsigned int msecs) {
  int i;
  for (i = 0; i < msecs; i++) {
    oneSecMethod();
  }
}

/*
 * Waste one second and calculates the time waited more than expected
 */
void* function(void* arg) {
  struct timespec start, stop;
  int s, ms, i;

  for (i = 0; i < 10; i++) {
    if (clock_gettime(CLOCK_REALTIME, &start) == -1) {
      perror("clock gettime");
      return (void *) EXIT_FAILURE;
    }

    waste_msecs(1000);

    if (clock_gettime(CLOCK_REALTIME, &stop) == -1) {
      perror("clock gettime");
      return (void *) EXIT_FAILURE;
    }

    usleep(100000);

    s = (stop.tv_sec - start.tv_sec) * 1000;
    ms = (stop.tv_nsec - start.tv_nsec) / MILLION;

    printf("start: %d, n %lu \n", start.tv_sec, start.tv_nsec);
    printf("stop: %d, n %lu \n", stop.tv_sec, stop.tv_nsec);
    printf("Waited miliseconds: %d\n", s + ms);
  }
  return EXIT_SUCCESS;
}

/*
 * Create Thread with highest priority
 * Print priority of thread
 */
int main(int argc, char *argv[]) {
  pthread_t thread_one;
  pthread_attr_t attr;
  struct sched_param param;
  int err;

  param.sched_priority = sched_get_priority_max(SCHED_FIFO);

  pthread_attr_init(&attr);
  // main Thread soll auf diesen Thread warten, damit das Programm erst beendet wird, wenn der Thread mit seiner Berechnung fertig ist. Dadurch wird sichergestellt, dass das Programm immer genau so lange lauft, wie der Thread rechnet und nicht schon vorher beendet wird.
  pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
  pthread_attr_setschedparam(&attr, &param);

  err = pthread_attr_setinheritsched(&attr, PTHREAD_EXPLICIT_SCHED);

  if (err != 0) {
    printf("pthread_attr_setinheritsched: %s ", strerror(err));
  }

  err = pthread_create(&thread_one, &attr, &function, NULL);

  if (err != 0) {
    printf("pthread_create: %s ", strerror(err));
  }

  pthread_attr_getschedparam(&attr, &param);

  printf("Thread Priority: %d\n", param.sched_priority);

  err = pthread_join(thread_one, NULL);

  if (err != 0) {
    printf("pthread_create: %s ", strerror(err));
  }
```

```

    return EXIT_SUCCESS;
}
```



Wie in der Messung zu sehen ist, wird genau eine Sekunde gerechnet.