Final Programming Report

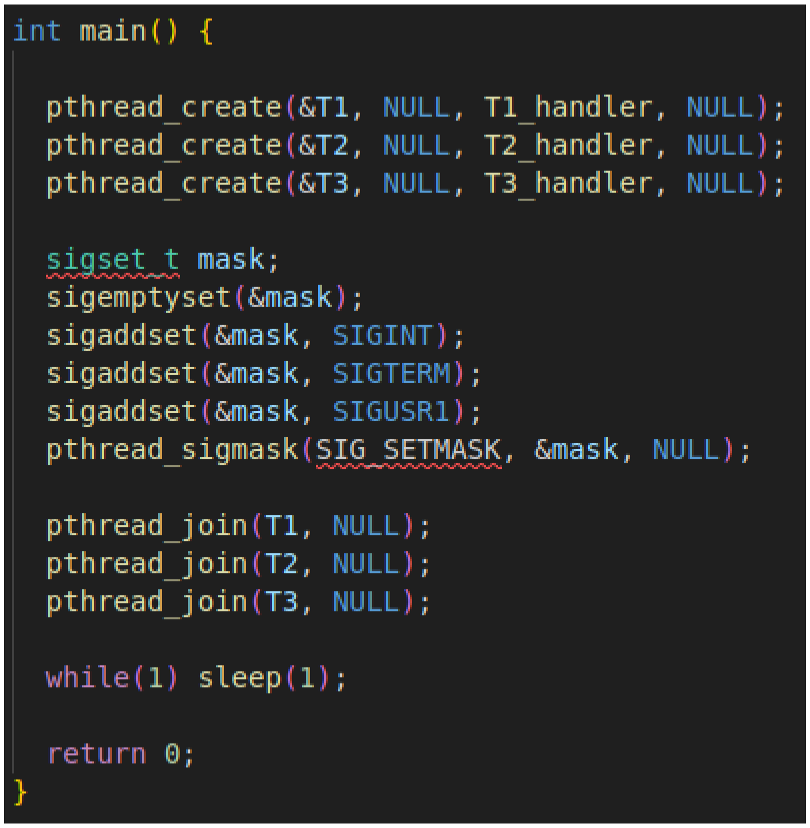
Team 13

NOTE: Use gmake (GNU make) to compile all files simultaneously.

Q1

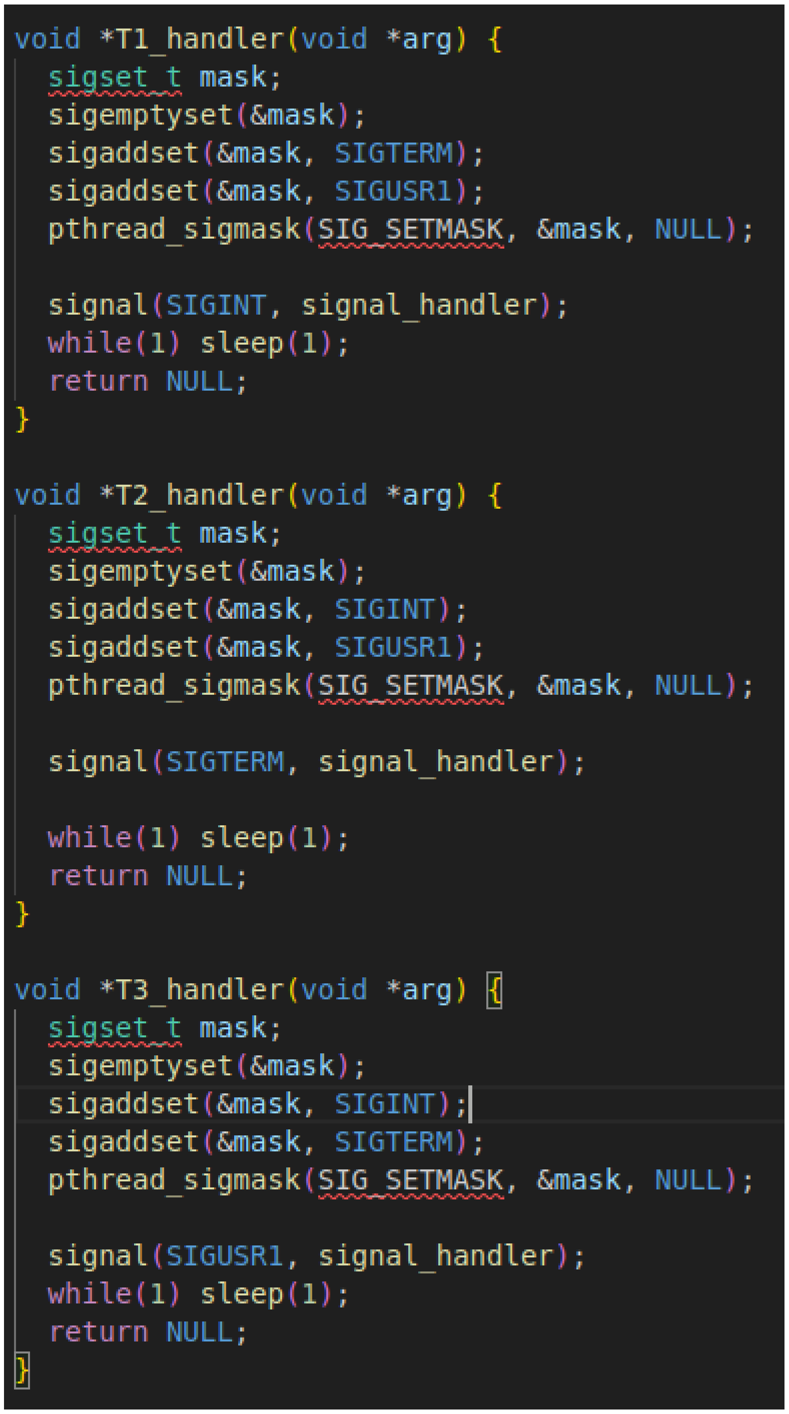
1. How to Run
   ./q1 &
   kill -USR1 %1
   kill -INT %1
   kill -TERM %1

2. Design
   a. In the main function, we create three threads T1, T2, and T3 with pthread_create(3). Then, we block signals SIGINT, SIGTERM, and SIGUSR1 to make the main function not to handle any signals. In the end, use pthread_join(3) to start all threads.



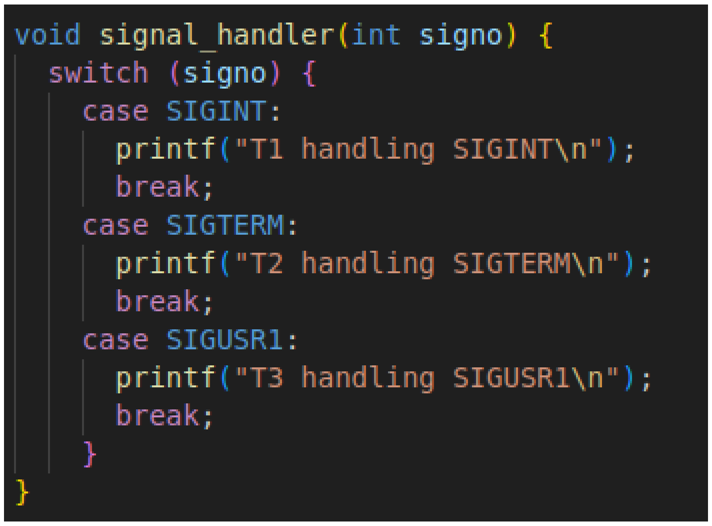
```c
int main() {

  pthread_create(&T1, NULL, T1_handler, NULL);
  pthread_create(&T2, NULL, T2_handler, NULL);
  pthread_create(&T3, NULL, T3_handler, NULL);

  sigset_t mask;
  sigemptyset(&mask);
  sigaddset(&mask, SIGINT);
  sigaddset(&mask, SIGTERM);
  sigaddset(&mask, SIGUSR1);
  pthread_sigmask(SIG_SETMASK, &mask, NULL);

  pthread_join(T1, NULL);
  pthread_join(T2, NULL);
  pthread_join(T3, NULL);

  while(1) sleep(1);

  return 0;
}
```

   b. In T1_handler, because thread T1 only needs to handle the SIGINT signal, we block the signal SIGTERM and SIGUSR1, and enroll the SIGINT signal.
   In T2_handler, because thread T2 only needs to handle the SIGTERM signal, we block the signal SIGINT and SIGUSR1, and enroll the SIGTERM signal.
   In T3_handler, because thread T3 only needs to handle the SIGUSR1 signal, we block the signal SIGINT and SIGTERM, and enroll the SIGUSR1 signal.



```c
void *T1_handler(void *arg) {
  sigset_t mask;
  sigemptyset(&mask);
  sigaddset(&mask, SIGTERM);
  sigaddset(&mask, SIGUSR1);
  pthread_sigmask(SIG_SETMASK, &mask, NULL);

  signal(SIGINT, signal_handler);
  while(1) sleep(1);
  return NULL;
}

void *T2_handler(void *arg) {
  sigset_t mask;
  sigemptyset(&mask);
  sigaddset(&mask, SIGINT);
  sigaddset(&mask, SIGUSR1);
  pthread_sigmask(SIG_SETMASK, &mask, NULL);

  signal(SIGTERM, signal_handler);

  while(1) sleep(1);
  return NULL;
}

void *T3_handler(void *arg) {
  sigset_t mask;
  sigemptyset(&mask);
  sigaddset(&mask, SIGINT);
  sigaddset(&mask, SIGTERM);
  pthread_sigmask(SIG_SETMASK, &mask, NULL);

  signal(SIGUSR1, signal_handler);
  while(1) sleep(1);
  return NULL;
}
```

c. In signal_hangler(), based on the signal number each thread receives, we print different results for the answer.



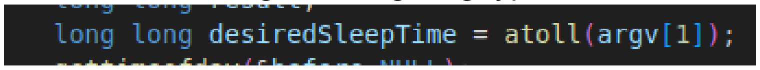
```c
void signal_handler(int signo) {
    switch (signo) {
        case SIGINT:
            printf("T1 handling SIGINT\n");
            break;
        case SIGTERM:
            printf("T2 handling SIGTERM\n");
            break;
        case SIGUSR1:
            printf("T3 handling SIGUSR1\n");
            break;
        }
}
```

d. We block the signals for each thread because when we give a command kill -USR1 %1, it will find the one which has the minimal id which is usually main function, so we block the signals which is no need to handle in each thread.

Q2

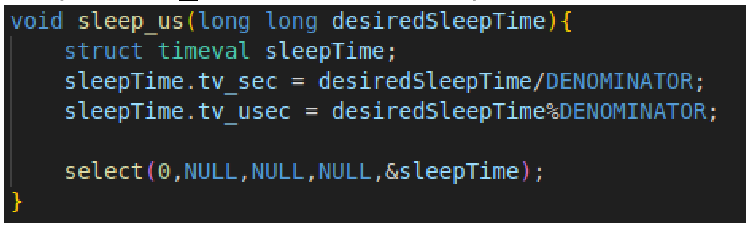
1. How to Run
   ./q2 <msec> Ex: ./q2 1000000

2. Design
   a. Use “struct timeval” from <sys/time.h> to store the desired sleep time.First, we get the input by argv[1] and use “atoll()” in <stdlib.h> to transform the string to a long long type in main function.



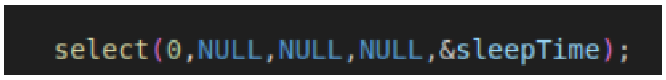
```c
    long long desiredSleepTime = atoll(argv[1]);
```

   Afterwards, we pass it in “sleep_us()” and set up “struct timeval sleepTime” by setting sleepTime.tv_sec = desiredSleepTime/1000000 and sleepTime.tv_usec = desiredSleepTime%1000000.



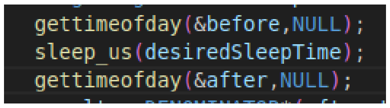
```c
void sleep_us(long long desiredSleepTime){
    struct timeval sleepTime;
    sleepTime.tv_sec = desiredSleepTime/DENOMINATOR;
    sleepTime.tv_usec = desiredSleepTime%DENOMINATOR;

    select(0,NULL,NULL,NULL,&sleepTime);
}
```

   b. Use select() from <sys/select.h> in “sleep_us()” to block the process until the time specified in “struct timeval sleepTime” expires.



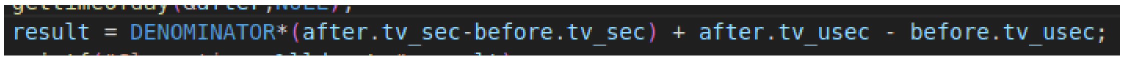
```c
    select(0,NULL,NULL,NULL,&sleepTime);
```

c. Get the time before calling “sleep_us()” in the main function by using “gettimeofday()” in <sys/time.h> .



```
gettimeofday(&before,NULL);
sleep_us(desiredSleepTime);
gettimeofday(&after,NULL);
```

d. Calculate the elapsed time using the “struct timeval before” and “struct timeval after”. After finishing the calculation, we store the value in “result” which is a long long type.



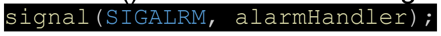
```
result = DENOMINATOR*(after.tv_sec-before.tv_sec) + after.tv_usec - before.tv_usec;
```

e. Print out the “result”. Since I use long long type parameters to prevent overflow, it might seem to be a longer time.

Q3

1. How to Run
   ./q3

2. Design
   a. Use alarm() from unistd.h to set the timer
   b. Use signal() from signal.h to handle the signals from the alarm
   c. Implement additional function alarmHandler(), this function is in charge of printing “Alarm!” and resetting the alarm after its finished.
   d. setAlarm() function is for setting the alarm to the parameter, and creating a signal



```
signal(SIGALRM, alarmHandler);
```

   e. clearAlarm() function is only in charge of resetting the alarm by using alarm(0)