Emir Sahbegovic / Chase Anderson

Operating Systems CSci 5103

Programming Assignment 1

Assumptions

We assume that the input will be valid. For example, in the example where pi is calculated, the library will not check if points_per_thread is lower than the total thread count. When this kind of input occurs, the library will not fail or exit, but the output will be unpredictable.

Initially, we had assumed that certain elements of the templates and code snippets given to us were sufficient and did not need to be modified. This is especially true with the code snippets given to us from the demo and the hint about signaling. We eventually came to the conclusion that we had to change the way the virtual timer signal worked. This is because we had to use **sigaction()** to instruct the system to ignore the timer when interrupts are disabled. The code snippets given in the hint do not support this functionality, and so we had to rewrite the functions. We also removed the **startInterruptTimer()** function because it seemed unnecessary.

We also decided to add an additional "FINISHED" state to the enum in TCB. This more accurately represents the states that threads have in other libraries and helps us differentiate between finished and blocked threads.

Given that in the problem statement / project document it is stated that: "We don't require students' code to achieve production-level quality.", we have elected to omit garbage collection from the scope of our project. If we were tasked with publishing this library to the public we feel it would be imperative to account for memory ailments and ensure leaks would not cause damage.

Finally, we made a secondary constructor for the main thread. We understood that the main thread needed to have a TCB, but we weren't sure how to use the main TCB constructor to do this.

Test Cases

We didn't have enough time to create a script which will run test cases automatically. We have screenshots which show what each test case output looks like. The test cases can be compiled and executed by editing the Makefile on line 4.

```
1   CC = g++
2   CFLAGS = -lrt -g
3   DEPS = TCB.h uthread.h
4   OBJ = TCB.o uthread.o test_default.o
```

1. test_default quantum==1000usec is used to check basic operation of context switching, init, and create.

```
and08479@csel-kh1262-03:~/Documents/SENIOR/operating_systems/PA1/pa1$ ./uthread-demo 1000000 5
switching to thread 1
1 exiting...
1 waiting for tid
switching to thread 2
2 exiting...
switching to thread 3
switching to thread 4
4 exiting...
switching to thread 5
switching to thread 0
switching to thread 3
3 exiting...
3 waiting for tid
switching to thread 5
5 exiting...
switching to thread 0
total quantums: 10
main quantums: 3
thread 1 quantums: 1
thread 2 quantums: 1
thread 3 quantums: 2
thread 4 quantums: 1
thread 5 quantums: 2
Pi: 3.14238
```

2. test_susRes quantum==1000000 is used to check the operation of suspend and resume functions. Notice that thread 1 switches to thread 3, rather than thread 2.

```
and08479@csel-kh1262-03:~/Documents/SENIOR/operating_systems/PA1/pa1$ ./uthread-demo 100000000 5
switching to thread 1
1 exiting...
1 waiting for tid
switching to thread 3
3 exiting...
switching to thread 4
4 exiting...
switching to thread 5
5 exiting...
switching to thread 0
switching to thread 2
2 exiting...
2 waiting for tid
switching to thread 0
total quantums: 8
main quantums: 3
thread 1 quantums: 1
thread 2 quantums: 1
thread 3 quantums: 1
thread 4 quantums: 1
thread 5 quantums: 1
Pi: 3.14138
```

3. test_yield quantum==1000000usec is used to check the operation of yield. It uses suspend() and resume() to illustrate that thread 2 is not suspended immediately due to main yielding.

```
and08479@csel-kh1262-03:~/Documents/SENIOR/operating_systems/PA1/pa1$ ./uthread-demo 100000000 5
switching to thread 1
1 exiting...
switching to thread 2
2 exiting...
switching to thread 3
3 exiting...
switching to thread 4
4 exiting...
switching to thread 5
5 exiting...
switching to thread 0
total quantums: 7
main quantums: 2
thread 1 quantums: 1
thread 2 quantums: 1
thread 3 quantums: 1
thread 4 quantums: 1
thread 5 quantums: 1
Pi: 3.14153
```

4. We also checked to make sure that the library wasn't making too many threads.

```
and08479@csel-kh1262-03:~/Documents/SENIOR/operating_systems/PA1/pa1$ ./uthread-demo 1000 101
error: Too many threads created!
```

5. Finally, we will mention our satisfactory test of get_quantum(int tid) and get_total_quantum(). Within each of our screenshots we have shown the printout statements of these functions; however, in the interest of cleanliness we have omitted these printouts for the published library.

Answer To Question C

Input and output parameters were given to the function using a void pointer. A void pointer allows us to pass in any type of data we would like to. The library is not concerned with the data, we only need to make sure that the pointer makes it to the function so that the data can be used appropriately. This requires the user of the library to type cast their data to a void pointer when creating the thread, and then type cast back to the original data type when the thread is joined to main.

The context given to makecontext() is usually a copy of the context which calls getcontext(). This means that makecontext() has to modify the context to make it point to the given function. It also needs to create a new stack for this function.