Project 02: Test Code

Implementing a simple kernel-level thread

Due date
2025. 05. 28. 23:59

Notice

- **The project deadline has been extended.**
  - Submit your work to GitHub by **May 28th** at 11:59 PM.
  - Late submissions should be emailed by May 29th at 11:59 PM.
  - Project 3 will be released on the 29th.
- This assignment only requires that all provided test cases work correctly. (Don't give up!)
- The messages shown in the examples are just samples, so you can write them freely.
- The output may vary, and the numbers don't need to be exactly same.
- The results should be logically correct.
- If you get different results, write the reason on the wiki.

Tips

- Consider and implement what is shared between threads and what is not shared.
- Function addresses may appear as 0.

```
1
2   user/_thread_test:     file format elf64-littleriscv
3
4
5   Disassembly of section .text:
6
7   0000000000000000 <thread_basic>:
8      0: 1101                   addi  sp,sp,-32
9      2: ec06                   sd  ra,24(sp)
10     4: e822                   sd  s0,16(sp)
11     6: e426                   sd  s1,8(sp)
12     8: 1000                   addi  s0,sp,32
13     a: 84aa                   mv  s1,a0
14     c: 85aa                   mv  a1,a0
15     e: 00001517             auipc a0,0x1
16    12: f9250513             addi  a0,a0,-110 # fa0 <thread_join+0x30>
17    16: 561000ef             jal d76 <printf>
18    1a: c899                   beqz  s1,30 <thread_basic+0x30>
19    1c: 85a6                   mv  a1,s1
20    1e: 00001517             auipc a0,0x1
21    22: f9a50513             addi  a0,a0,-102 # fb8 <thread_join+0x48>
```

Test 1

- This test evaluates the basic functionality of the thread API and checks if memory is properly shared between threads.
- Thread 0 modifies a global variable, while other threads terminate immediately.
- Thread 0 should terminate last, and the main thread should be able to detect the modifications made by Thread 0.

```
[TEST#1]
Thread Thread 1 start
Thread 1 end
Thread 2 start
Thread 2 end
Thread 3 start
Thread 3 end
Thread 4 start
Thread 4 end
0 start
Thread 0 end
TEST#1 Passed
```

Test 2

- This test verifies that threads correctly receive two arguments and properly write to shared resources.
- The main thread also checks whether the threads' operations behaved as expected.

```
[TEST#2]
Thread 0 start, iter=0
Thread 0 end
Thread 1 start, iter=1000
Thread 1 end
Thread 2 start, iter=2000
Thread 2 end
Thread 3 start, iter=3000
Thread 3 end
Thread 4 start, iter=4000
Thread 4 end
TEST#2 Passed
```

Test 3

- This test verifies that threads handle fork correctly.
- After forking, the parent process should operate in the address space of the original process (main thread), while the child process should operate in a separate address space.
- The test detects an error if the child process shares the address space with the parent process.

```
[TEST#3]
Thread 0 start
Thread 1 start
Thread 2 start
Thread 3 start
Thread 4 start
Child of thread 0 start
Child of thread 1 start
Child of thread 2 start
Child of thread 3 start
Child of thread 4 start
Child of thread 0 end
Child of thread 1 end
Child of thread 2 end
Child of thread 3 end
Child of thread 4 end
Thread 0 end
Thread 1 end
Thread 2 end
Thread 3 end
Thread 4 end
TEST#3 Passed
```

Test 4

- This test verifies that threads handle sbrk correctly.
  - Using malloc internally calls sbrk.
- The test checks if there are no issues when other threads access memory allocated by one thread.
- The test also verifies that when multiple threads allocate their own memory, there are no duplicate allocations at overlapping addresses.

```
[TEST#4]
Thread 0 sbrk: old break = 0x0000000000015000
Thread 0 sbrk: increased break by 14000
new break = 0x0000000000029010
Thread 1 size = 0x0000000000029010
Thread 2 size = 0x0000000000029010
Thread 3 size = 0x0000000000029010
Thread 4 size = 0x0000000000029010
Thread 0 sbrk: free memory
Thread 0 end
Thread 1 end
Thread 2 end
Thread 3 end
Thread 4 end
TEST#4 Passed
```

Test 5

- This test verifies that threads handle kill correctly.
- When the main thread terminates, all threads should terminate.
- When any other thread terminates, only that thread should terminate.

```
[TEST#5]
Thread 0 start, pid 29
Thread 1 start, pid 29
Thread 2 start, pid 29
Thread 3 start, pid 29
Thread 4 start, pid 29
Thread 0 end
TEST#5 Passed
```

Test 6

- This test verifies that threads handle exec correctly.
- Thread 0 executes the `thread_fcn` program.
- When exec is called, all threads should terminate and a new image should be executed.

```
[TEST#6]
Thread 0 start
Thread 1 start
Thread 2 start
Thread 3 start
Thread 4 start
Executing...
Thread exec test 0
TEST#6 Passed
```

Result

```
$ thread_test

[TEST#1]
Thread 0 start
Thread 1 start
Thread 1 end
Thread 2 start
Thread 2 end
Thread 3 start
Thread 3 end
Thread 4 start
Thread 4 end
Thread 0 end
TEST#1 Passed

[TEST#2]
Thread 0 start, iter=0
Thread 0 end
Thread 1 start, iter=1000
Thread 1 end
Thread 2 start, iter=2000
Thread 2 end
Thread 3 start, iter=3000
Thread 3 end
Thread 4 start, iter=4000
Thread 4 end
TEST#2 Passed
```

```
[TEST#3]
Thread 0 start
Thread 1 start
Thread 2 start
Thread 3 start
Thread 4 start
Child of thread 0 start
Child of thread 1 start
Child of thread 2 start
Child of thread 3 start
Child of thread 4 start
Child of thread 0 end
Child of thread 1 end
Child of thread 2 end
Child of thread 3 end
Child of thread 4 end
Thread 0 end
Thread 1 end
Thread 2 end
Thread 3 end
Thread 4 end
TEST#3 Passed

[TEST#4]
Thread 0 sbrk: old break = 0x0000000000015000
Thread 0 sbrk: increased break by 14000
new break = 0x0000000000029010
Thread 1 size = 0x0000000000029010
Thread 2 size = 0x0000000000029010
Thread 3 size = 0x0000000000029010
Thread 4 size = 0x0000000000029010
Thread 0 sbrk: free memory
Thread 0 end
Thread 1 end
Thread 2 end
Thread 3 end
Thread 4 end
TEST#4 Passed
```

```
[TEST#5]
Thread 0 start, pid 29
Thread 1 start, pid 29
Thread 2 start, pid 29
Thread 3 start, pid 29
Thread 4 start, pid 29
Thread 0 end
TEST#5 Passed

[TEST#6]
Thread 0 start
Thread 1 start
Thread 2 start
Thread 3 start
Thread 4 start
Executing...
Thread exec test 0
TEST#6 Passed

All tests passed. Great job!!
$
```

Q & A