CMPT 300 Assignment 2

Name: Ruijia Mao
SFU ID: 301295769

1 Code Explanation

The answer to exercise 1 is in the comment of h_timer.c.

2 Measure Methods and Results

In order to measure the time precisely without the overhead of the initialization of counter and for loop. I first loop for 100000 times and initialize two counters each time in for loop and perform the operation between two counters and then I just loop loop for 100000 times and initialize two counters each time in for loop and do nothing between the two counters. Then I subtract the two results and divide the difference by the number of iteration.

2.1 Function Call

The time for a minimum function call is 0.1 ns in a csil machine and 0.4 - 0.6 ns in my laptop.

I first call function $clock_gettime()$ with a CLOCK_PROCESS_CPUTIME_ID counter to get the start time. And use a for loop to test the empty function call for 100000 times. And then call function $clock_gettime()$ eith a CLOCK_PROCESS_CPUTIME_ID counter to get the stop time. Result is computed by subtraction of the start and stop.

And in order to reduce the overhead of initialization of clock and for loop, I also call function $clock_gettime()$ twice and do an idle for loop for 100000 times.

2.2 System Call

The time for a minimum function call is 2-3 ns in a csil machine and my laptop.

I first call function $clock_gettime()$ with a CLOCK_PROCESS_CPUTIME_ID counter to get the start time. And use a for loop to test the $getpid()$ which is the simplest system call for 100000 times. And then call function $clock_gettime()$ eith a CLOCK_PROCESS_CPUTIME_ID counter to get the stop time. Result is computed by subtraction of the start and stop.

And in order to reduce the overhead of initialization of clock and for loop, I also call function $clock_gettime()$ twice and do an idle for loop for 100000 times.

2.3 Process Context Switch

The time for a minimum function call is 1000-2000 ns in my laptop.

I first use $sched_setaffinity()$ to set the running CPU to be one single CPU. Then, I call function $clock_gettime()$ with a CLOCK_MONOTONIC counter to get the start time. I use $shed_yield()$ to complete context switch for 100000 times. And then I call function $clock_gettime()$ with a CLOCK_MONOTONIC counter to get the stop time. Result is computed by subtraction of the start and stop.

And in order to reduce the overhead of initialization of clock and for loop, I also call function $clock_gettime()$ twice and do an idle for loop for 100000 times.

2.4 Thread Context Switch

The time for a minimum function call is 1000-2000 ns in my laptop.

I first use $sched_setaffinity()$ to set the running CPU to be one single CPU. Then, I call function $clock_gettime()$ with a CLOCK_PROCESS_CPUTIME_ID counter to get the start time. I use functions $pthread_{m}utex_{l}ock()$, $pthread_{c}ond_{w}ait()$ and $pthread_{c}ond_{s}ignal()$ to implement the method described in a2.pdf to complete context switch for 100000 times. And then I call function $clock_gettime()$ with a CLOCK_PROCESS_CPUTIME_ID counter to get the stop time. Result is computed by subtraction of the start and stop.

And in order to reduce the overhead of initialization of clock and for loop, I do an for loop for 100000 times and call function $clock_gettime()$ twice in the for loop.