

Program 3

Haram Kwon

Implementation

Sync Queue and Queue Node

In order to implement SyncQueue, we need to implement QueueNode first because we are going to invoke QueueNode class in SyncQueue class.

For the QueueNode, we are going to simulate the Process with process id, and it will simply contains the pid into the vector. The methods that this functions has is to make process sleep and wake up.

For SyncQueue, we are using SyncNode, and we are going to simulate the monitor based on syncqueue. By this implementation, we are going to prevent the busy waiting on our system.

Test 3

In order to implement Test 3, we've noticed the behavior of TestThread3.class with instruction. In order to run the shell with TestThread3.class, we need to pass two parameter: "comp", or "disk". By this factor, in order to implement test3.java, we only need to figure out how to invoke TestThread3.class. As we've practice in program 1, we could invoke the command "SysLib.stringToArgs" and "Syslib.exec" syntax to send our two different command through the parameter.

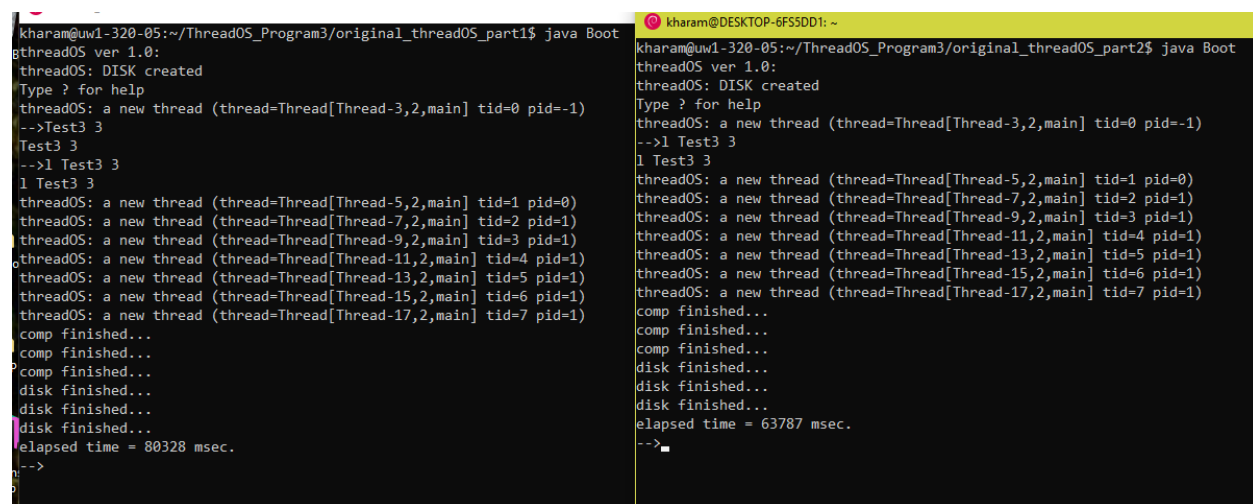
"Example of how to set the parameter."

```
String[] args1 = SysLib.stringToArgs( "TestThread3 comp" ); // computation  
intensive  
String[] args2 = SysLib.stringToArgs( "TestThread3 disk" ); // disk intensive
```

"Example how to execute the command"

```
SysLib.exec( args1 );  
SysLib.exec( args2 );
```

Performance



```
kharam@uw1-320-05:~/ThreadOS_Program3/original_threadOS_part1$ java Boot
gthreadOS ver 1.0:
threadOS: DISK created
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->Test3 3
Test3 3
-->l Test3 3
l Test3 3
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=1)
comp finished...
comp finished...
comp finished...
disk finished...
disk finished...
disk finished...
elapsed time = 80328 msec.
-->

kharam@DESKTOP-6FSSDD1: ~
kharam@uw1-320-05:~/ThreadOS_Program3/original_threadOS_part2$ java Boot
threadOS ver 1.0:
threadOS: DISK created
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Test3 3
l Test3 3
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=1)
comp finished...
comp finished...
comp finished...
disk finished...
disk finished...
disk finished...
elapsed time = 63787 msec.
-->
```

Figure 1: Screen shot of Part1 (left) Part2 (right) running Test3

Test3 on Kernel_1.java (Kernel.old) vs. Test3 on Kernel.java

There is slight performance improvement when we are using monitor instead of busy waiting (20000 msec = 20 sec). This is because the monitor approach improve the way of notifying the thread to wake up. Since part1 provides a way to search the next job from the disk in brute force, while the monitor make some process wake and work. It can save the time between another processor working, and entire processors are sleeping.