Volume 5, Issue 11, November 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Experiment to Generate Thread Priority Map Table for Java Thread and Windows Operating System Thread

[1]Dr. Yogesh R Ghodasara, [2]Dr. G J Kamani, [3]Dr. Kamani Krunalkumar C, [4]Parag M Moteria
[1]Associate Professor, College of AIT, Anand Agricultural University, Anand, India
[2]Assistant Professor, College of Agricultural Engineering and Technology, Anand Agricultural University, Godhra, India
[3]Assistant Professor, Sheth M C Dairy Science College, Anand Agricultural University, Anand, India
[4]Ph.D. School of Computer Science, R K University, Rajkot, India

Abstract— Process scheduling policies and Thread scheduling policies play vital role in the performance of any multitasking operating system. When thread is executed by the operating system, operating system assigns various priority levels to give fair chance of execution. Computer programming languages like java also provides properties and functions to assign different priority levels to any thread created in java program. The java programming language supports ten thread priority levels. While Windows operating system supports thirty two thread priority levels. This paper describe an experiment carried out to find out relation between different thread priority levels given in the java programming language with the thread priority levels in windows operating system. The result of the experiment shows that which java thread priority level is equivalent to windows thread priority level.

Keywords— Multithreading, Priority, Scheduling, Transition States, Thread.

I. INTRODUCTION

As processor capabilities have increased, so have demands on performance, which has increased pressure on processor resources with maximum efficiency. Noticing the time that processors wasted running single tasks while waiting for certain events to complete, software developers began wondering if the processor could be doing some other work at the same time. To arrive at a solution, software architects began writing operating systems that supported running pieces of programs, called threads. Threads are small tasks that can run independently. Each thread gets its own time slice, so each thread represents one basic unit of processor utilization. Threads are organized into processes, which are composed of one or more threads. All threads in a process share access to the process resources [1].

Modern operating systems which allow multiple tasks to run simultaneously is called multitasking operating system. Similarly operating systems which allows multiple threads to run simultaneously is called multithreading operating system. Windows operating system is a multitasking and multithreading operating system. Being a multithreading operating system, windows follows thread base scheduling system in which highest priority thread will grab the CPU for execution.

II. THREAD STATE

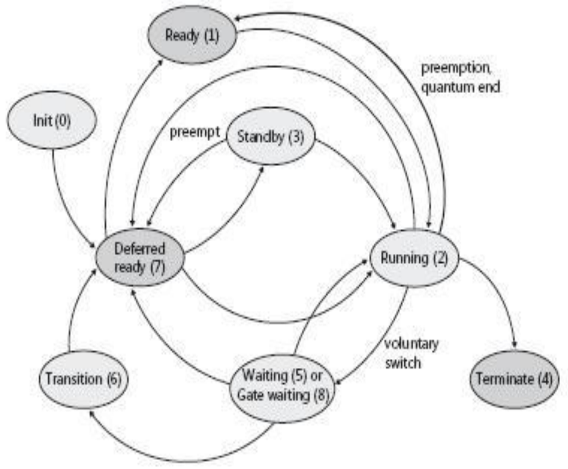


Figure 1: Thread States and Transitions

Once thread is created in the process, thread can be in one of the following state. The thread state decides current situation of a thread in the system. The thread can be in any one state at a time. The transition from one state to another state is possible during the execution of the thread. The thread states and transitions are shown in figure 1. [2]-[7]

■ Ready
A thread in the ready state is waiting for CPU to execute.
■ Deferred ready
A thread in the deferred ready state if selected to run on a specific processor but have not yet been scheduled.
■ Standby
A thread in the standby state has been selected to run next on a particular processor. A thread can be preempted out of the standby state before it ever executes if a higher priority thread becomes runnable before the standby thread begins execution.
■ Running
A thread is in execution.
■ Waiting
A thread can enter the wait state if waits for I/O event to finish.
■ Gate Waiting
When a thread does a wait on a gate dispatcher object, it enters the gate waiting state instead of the waiting state.
■ Transition
A thread enters the transition state if it is ready for execution but its kernel stack is paged out of memory.
■ Terminated
When a thread finishes execution, it enters the terminated state.
■ Initialized
This state is used internally while a thread is being created.

III. WINDOW SCHEDULING SCENARIOS

Windows implements a priority-driven, preemptive scheduling system and schedules at thread granularity level. That is the highest-priority runnable (ready) thread always gets the CPU for execution. All scheduling decisions are made strictly on a thread basis and no consideration is given to what process the thread belongs to. Windows scheduler schedules thread using following scheduling scenarios. [2]-[5][8]

A. Voluntary Switch

In voluntary switch, a thread might voluntarily relinquish use of the processor by entering a wait state by calling one of the Windows wait functions.

B. Preemption

In preemption, a lower-priority thread is preempted when a higher-priority thread becomes ready to run. This situation might occur for a couple of reasons:

- ■ A higher-priority thread's wait completes. The event that the other thread was waiting for has occurred.
- ■ A thread priority is increased or decreased.

C. Quantum End

When a thread is selected to run, it runs for an amount of time called a quantum. A quantum is the length of time a thread is allowed to run before another thread at the same priority level. When the running thread exhausts its CPU quantum, Windows must determine whether the thread's priority should be decremented and then whether another thread should be scheduled on the processor.

IV. THREAD PRIORITY LEVELS IN WINDOWS

To understand the thread-scheduling algorithms, you must first understand the priority levels that Windows uses. As illustrated in Figure 2, internally Windows uses 32 priority levels, ranging from 0 through 31. These values divide up as follows:[2]-[5][9]-[10]

- ■ Sixteen real-time levels (16 through 31)
- ■ Fifteen variable levels (1 through 15)
- ■ One system level (0)

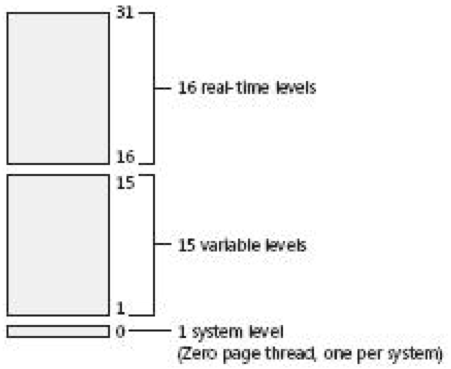


Figure 2: Thread Priority Levels in Windows

Threads are scheduled to run based on their scheduling priority. Each thread is assigned a scheduling priority. The priority levels range from zero (lowest priority) to 31 (highest priority). Only the zero-page thread can have a priority of zero. The zero-page thread is a system thread responsible for zeroing any free pages when there are no other threads that need to run.

The system treats all threads with the same priority as equal. The system assigns time slice(quantum) in a round-robin fashion to all threads with the highest priority. If none of these threads are ready to run, the system assigns time slices in a round-robin fashion to all threads with the next highest priority. If a higher-priority thread becomes available to run, the system ceases to execute the lower-priority thread without allowing it to finish using its time slice, and assigns a full time slice to the higher-priority thread.

The priority of each thread is determined by the following criteria:

- The priority class of its process.
- The priority level of the thread within the priority class of its process.

The priority class and priority level are combined to form the base priority of a thread. Each thread has a dynamic priority. This is the priority the scheduler uses to determine which thread to execute. Initially, a thread's dynamic priority is the same as its base priority. The system can boost and lower the dynamic priority, to ensure that it is responsive and that no threads are starved for processor time. The system does not boost the priority of threads with a base priority level between 16 and 31. Only threads with a base priority between 0 and 15 receive dynamic priority boosts. [8][11]-[19]

V. MULTITHREADING IN JAVA

Java provides built-in support for multithreaded programming.[20]-[22]

A. Thread States in Java

A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies. Following diagram shows complete life cycle of a thread in java.

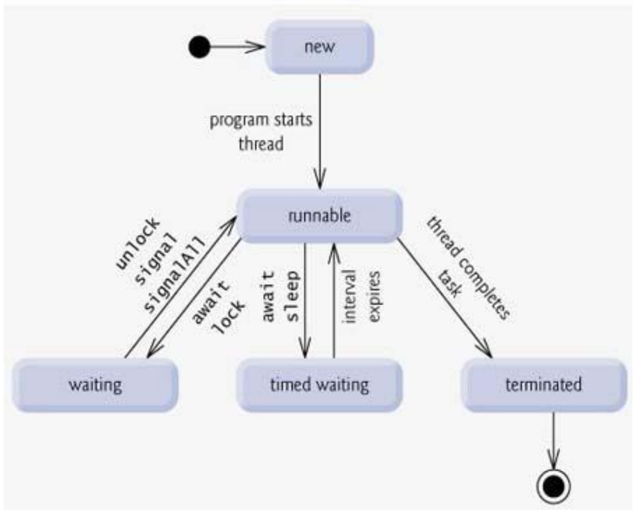


Figure 3: Thread Life Cycle

New: A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.
Runnable: After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.
Waiting: Sometimes a thread transitions to the waiting state while the thread waits for another thread to perform a task. The thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.
Timed waiting: A runnable thread can enter the timed waiting state for a specified interval of time. The thread in this state transition back to the runnable state when that time interval expires or when the event it is waiting for occurs.
Terminated: A runnable thread enters the terminated state when it completes its task or otherwise terminates.

B. Thread Priorities in Java

Every Java thread has a priority that helps the operating system to determine the order in which threads are scheduled. Java priorities are in the range between MIN_PRIORITY (a constant of 1) and MAX_PRIORITY (a constant of 10). By default, every thread is given priority NORM_PRIORITY (a constant of 5).[20]-[24]

Value	**Description**	**IntegerValue**
MIN_PRIORITY	Min. Priority	1
NORM_PRIORITY	Normal Priority	5
MAX_PRIORITY	Maximum Priority	10

Thus Java supports ten different priority levels. The Thread Class of java.lang package has a method called setPriority using which we can set different priority levels in Java program.

VI. EXPERIMENTAL DESIGN

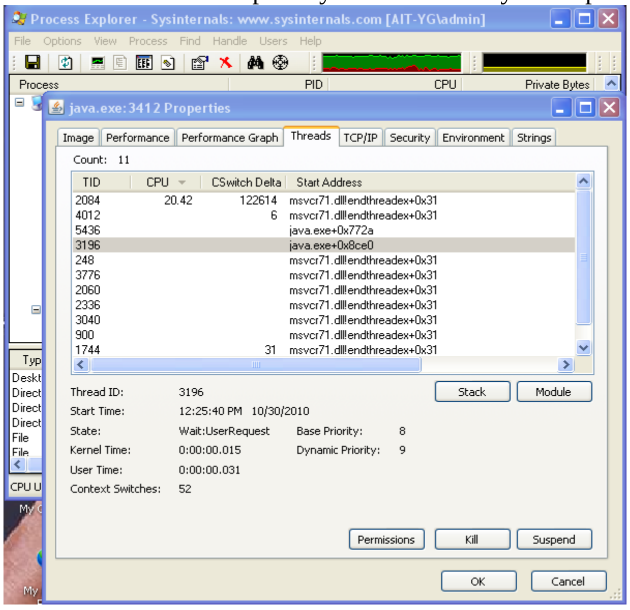
Java programming language has Thread class in java. lang package. By extending Thread class we can create thread in any java program. The Thread class has setPriority method which is inherited by subclass of Thread class. Using this method we can set different priority levels. Run the java program by setting different priorities in the subclass using setPriority method. View the output of each program run in the process explorer to get the relevant priority value of windows operating system thread.

An experiment can be design as under to generate thread priority map table for java thread and windows operating system thread.

Step 1:
Create a java sub class MyThread by extending Thread Class.
Step 2:
Override run method of subclass MyThread. Print "Hello World" infinite number of times.
Step 3:
Create a java class ThreadDemo. Create object of MyThread class in main method of ThreadDemo class.
Step 4:
Set priority level using setPriority method.
Step 5:
Compile and Run ThreadDemo program by setting different priority levels.
Step 6:
Check the priority of windows operating system for the priority set in the java program using Process Explorer Tool.

Process Explorer is an advanced process management utility. It will show you detailed information about a process including its icon, command-line, full image path, memory statistics, user account, security attributes, thread information etc. This tool is useful to display base priority and dynamic priority of windows operating system. In this experiment we will execute java program by setting different thread priority and view base priority and dynamic priority using this tool.

Using above experiment design we can find that if we set different priority level in java programming language, associated windows priority level is available in process explorer as output. By setting ten different priorities in java program, we get ten different values for windows base priority and windows dynamic priority as shown in figure 4.



```
Count:  11

TID     CPU     CSwitch Delta   Start Address
2084    20.42   122614          msvcr71.dll!endthreadex+0x31
4012            6               msvcr71.dll!endthreadex+0x31
5436                            java.exe+0x772a
3196                            java.exe+0x8ce0
248                             msvcr71.dll!endthreadex+0x31
3776                            msvcr71.dll!endthreadex+0x31
2060                            msvcr71.dll!endthreadex+0x31
2336                            msvcr71.dll!endthreadex+0x31
3040                            msvcr71.dll!endthreadex+0x31
900                             msvcr71.dll!endthreadex+0x31
1744            31              msvcr71.dll!endthreadex+0x31

Thread ID:          3196
Start Time:         12:25:40 PM   10/30/2010
State:              Wait:UserRequest    Base Priority:      8
Kernel Time:        0:00:00.015         Dynamic Priority:   9
User Time:          0:00:00.031
Context Switches:   52
```

Figure 4: Process Explorer Output

VII. EXPERIMENTAL OUTPUT

The experiment gives following Thread Priority Map Table for Java Thread and Windows Operating System Thread.

Java Thread Priority Value	**Windows Base Priority Value**	**Windows Dynamic Priority Value**
1	6	6
2	6	6
3	7	7
4	7	7
5	8	8

6	8	8
7	9	9-10
8	9	9-10
9	10	10-11
10	10	10-11

VIII. SUMMARY AND CONCLUSIONS

The experimental result shows that the java thread priority range 1 to 10 is equal to windows base priority range 6 to 10. It also reveals that though Windows operating system gives 32 different priority levels, java language thread can set priority value between 6 to 10 base priorities of windows operating system.

REFERENCES

[1] Intel Hyper-Threading Technology, Technical User’s Guide. Available: http://cache-www.intel.com/cd/00/00/01/77/17705_htt_user_guide.pdf

[2] http://www.readbag.com/live-sysinternals-windowsinternals-windowsinternals-ch05

[3] http://www.readbag.com/download-microsoft-download-5-b-3-5b38800c-ba6e-4023-9078-6e9ce2383e65-c06x1116607

[4] Thread Scheduling API & their functions. Available: http://flylib.com/books/en/4.491.1.55/1/

[5] Overview of Windows 2000 Thread Scheduling. Available: http://flylib.com/books/en/3.169.1.54/1/

[6] Multithreading tutorials by Andey Krishnaji. Available: http://www.ijice.org/2014_10_09_archive.html

[7] http://www.researchgate.net/profile/Mohamed_El-Sherbeny/publication/26401698_Profit_Analysis_of_A_Two_Unit_Cold_Standby_System_with_Preventive_Maintenance_and_Random_Change_in_Units/links/54e2ed180cf2c3e7d2d4f1fc.pdf

[8] Thread Scheduling. Available: http://www.microsoft.com/mspress/books/sampchap/4354c.aspx

[9] Thread Priority. Available: http://www.fer.unizg.hr/_download/repository/scheduling_O(1).html

[10] Thread Scheduler, Thread Granularity and Priority of Thread. Available: http://stackoverflow.com/questions/656959/win32-thread-scheduling

[11] Process and Threads: Win32/WINDOWS API. Available: http://www.tenouk.com/ModuleT1.html

[12] Scheduling Priority Levels. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/ms685100(v=vs.85).aspx

[13] Different Thread Scheduler. Availale: http://superuser.com/questions/414604/difference-between-the-windows-and-linux-thread-scheduler

[14] Multitasking with Thread and Process. Available:http://www.grahamwideman.com/gw/tech/dataacq/procthread.htm

[15] Round Robin Scheduling. Available: http://cpp.knowcoding.com/help/round+robin+scheduling+algorithm.html

[16] The Windows Processes and Threads. Available: http://www.installsetupconfig.com/win32programming/windowsthreadsprocessapis7_1.html

[17] http://forums.codeguru.com/showthread.php?301665-Is-it-wise-to-set-a-priority-of-a-thread

[18] Priority Boosts. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/ms684828(v=vs.85).aspx

[19] Priority Boosts. Available: http://winapi.freetechsecrets.com/win32/WIN32Priority_Boosts.htm

[20] Java Multithreading. Available: http://www.tutorialspoint.com/java/java_multithreading.htm

[21] Java Multithreading. Available: http://divetojava.blogspot.in/

[22] Java Multithreading. Available: http://www.tutorialspoint.com/java/pdf/java_multithreading.pdf

[23] All about Java Threads. Available: http://www.niecdelhi.ac.in/uploads/Notes/btech/5sem/cse/Java%20Notes%201%20-%20Multithreading.pdf

[24] Thread Priority and Scheduling. Available: http://flylib.com/books/en/2.254.1.437/1/