

Forum:[Hack Forums](#)

Thread: [Multi-tasking & Multi-threading for Win OS - Q & A's \(Part-5\)](#)

?si?qo? - December 27, 2008, 7:20 am

- Part One: <https://hackforums.net/showthread.php?tid=39070>
- Part Two: <https://hackforums.net/showthread.php?tid=40287>
- Part Three: <https://hackforums.net/showthread.php?tid=41526>
- Part Four: <https://hackforums.net/showthread.php?tid=41528>

I will be talking about multi-threading concepts based on Windows OS...It is important to understand the concepts before you go to implementation!

Q: What is fork()?

A: please look at the tutorial on the txt file called fork_tutorial.txt.

fork() is only available in nix systems, not Windows. Windows have CreateProcess. CreateProcess is very different from fork. They kind of relies on the same concept, that is creating several processes, but the implementation between fork() and CreateProcess() is completley different based on what experts said.

fork() is used for multi-processing. fork() gives you the ability to open the program in several windows and use them all at the same time, e.g. you open two or three new browsers and use them. That is what fork lets you do. Gives your application the ability to executed several times at once, such opening several windows at the same time of the same application. Another example, MSN is not designed to be multi-processed, since you only can open one MSN messenger window at a time.

To learn how to make your application multi-threaded in *nix & know what it is, please visit

<http://docs.sun.com/app/docs/doc/802-1949>
<http://www.computersciencelab.com/MultithreadingTut1.htm>
google!

Below we are going to talk about the concept of creating processes and threads under windows, not Unix.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: What the heck is processes and threads (single thread & multi-threads)?

A: A process, in the simplest terms, is an executing program.
One or more threads run in the context of the process. Each process is started with a single thread, often called the primary thread, but can create additional threads from any of its threads.

For example, you open chatting application. This application gives you the ability to chat with more than one person in PM's. You open a chat box with james and another chat box with mike, at the same time you can do many other things on the chat application.

when you open the chatting application, you have made a process into your machine.

Every process gives you the ability to handle one thread at a time, which means once you open the chatting application, you can do one thing only until you finish it, then move to the next thing, e.g. you can only chat with one person at a time. You can't do anything else, until you close the chatting box of that person & then move to the next thing you want to do, e.g. chat with the another one. This is kind of application is called single threaded. But when you can do several things at the same time on that chatting application, this chatting application is called an application with multi-thread abilities.

In short,

the executed chatting application is called process.
the activities that happens on the applications called threads.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*

Q: What is multitasking?

A: multi-task refers to a capability of the OS, namely to run multiple programs at the same time, and taking care of giving each of those programs a fair share of CPU time (note that most modern OSs are multi-task OSs).

In other words, is a method by which multiple tasks, also known as processes, share common processing resources such as a CPU. With a multitasking OS, you can simultaneously run multiple applications. Multitasking refers to the ability of the OS to quickly switch between each computing task to give the impression the different applications are executing multiple actions simultaneously.

NOTE: Because each time slice is small (approximately 20 milliseconds), multiple threads appear to be executing at the same time. You must use caution when using multiple threads in an application, because system performance can decrease if there are too many threads.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*

Q: Can you tell me more about multi-tasking & multi-threading?

A: multitasking is a heavyweight process and runs in a different address space so context switch or intercommunication between processes is much expensive, e.g. on Windows you can run microsoft word, MSN, firefox, recording audio, etc all at the same time. Every of this application, which is known as a process, has its own address space, and since all these processes share the same CPU, CPU has to deal with them equally by sharing the time slice amongst them which is approximately 20 milliseconds & this gives the OS the ability to quickly switch between each computing task to give the impression that the different applications are executing multiple actions simultaneously (at the same time). Since every of this process runs in a different address space, so context switch or intercommunication between processes is much expensive. In other words, process are heavyweight tasks that require their own addressspace. Interprocess communication is very expensive and context switching from one process to another procees is costly since they are running in different address spaces.

Remember: every process can have multi-threads too... So with multitask you can have several processes and each of these processes can be multi-threads...

multi-tasking consideration:

The system consumes memory for the "context information" required by processes and threads. Therefore, the number of processes and threads that can be created is limited by available memory. If there are too many threads going around, most of them will not be able to make significant progress, since all of them use a shared processing resource which is CPU & CPU do time slices and share the time amongst them accordingly. If most of the current threads are in one process, threads in other processes are scheduled less frequently.

Multithreading requires less overhead than multitasking. Multithreading threads are lightweight process and can share same address space and inter-thread communication is less expensive than inter-process communication.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: What kind of "context information" required by processes and threads?

A: for sake of simplicity, i can say the thread context includes the thread's set of machine registers, the kernel stack, etc...

process context has a virtual address space, executable code, open handles to system objects, a security context, a unique process identifier, environment variables, etc.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: "If most of the current threads are in one process, threads in other processes are scheduled less frequently." What the heck! can you schedule a thread?

A: Yes, the scheduler determines which thread runs next using scheduling priorities.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: What is round-robin scheduling fashion?

A: well, google is your friend! Anyway, here is the answer:

Round-robin (RR) is one of the simplest scheduling algorithms for processes in an operating system, which assigns time slices to each process in equal portions and in order, handling all processes without priority. It is one of the oldest, simplest, fairest and most widely used scheduling algorithms, designed especially for time-sharing systems. A small unit of time, called time slice or quantum, is defined. All runnable processes are kept in a circular queue. The CPU scheduler goes around this queue.

Every time a process pick out the CPU, a process context switch occurs, which adds overhead to the process execution time. As we have mentioned above context switching from one process to another process is costly since they are running in different address spaces.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

General Note: scheduling threads on a computer with multiple processors is similar to scheduling threads on a computer with a single processor. However, the scheduler has a pool of processors, so that it can schedule threads to run concurrently.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: What is a child process?

A: A child process is a process that is created by another process, called the parent process. For example, opening two or more windows of the same application, such as opening more than one internet explorer browser.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Question & Answers Regarding Some Parts Explained Above:-

Quote: "The system treats all threads with the same priority as equal. The system assigns time slices 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 rejects 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. Round-robin (RR) is one of the simplest scheduling algorithms for processes in an operating system, which assigns time slices to each process in equal portions and in order, handling all processes without priority."

>>If a higher-priority thread becomes available to run, the system rejects 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

Q1. the process will only run one thread at a time since the system takes the highest priority thread only, so all other threads will be rejected! We conclude with one thread running, similar to single-threading. Can you clarify please?

A1. At any time, on one processor, only one thread/process can execute. Multithreading is most helpful for multiprocessor systems.
Your understanding is correct ... if there is only one procesor, high pri thread will run .. others would wait in a queue.

>1>The system assigns time slices in a round-robin fashion to all threads with the highest priority.

>2>Round-robin (RR) is one of the simplest scheduling algorithms for processes in an operating system, which assigns time slices to each process in equal portions and in order, handling all processes without priority

Q2.In quote 2, round-robin algorithm handles all process without priority. In quote 1, it says the system assigns time slices in a round-robin fashion to all threads with the highest priority. RR algorithm handles all process without priority, but in case of threading RR cares about priority. Right?

A2. Typically if there are n priority levels, you would have n queues. OS would look for threads in highest priority queue. All of the threads in the queue would get processor in RR fashion. Once all of

these threads have finished, OS would look for process in next level. You can visualize it as priority queues with RR in each queue.

>> if there is only one processor, high pri thread will run .. others would wait in a queue

Q3. Yes, but one processor can have multi-threads & this quote is taken from multi-threading msdn guide. Here we are talking about a program that runs many activities at once. Its not fair, if i run 3 activities that keeps running, in a sudden a higher priority activity appears. All the 3 activities will be killed automatically, just to server the higher priority one. If thats the care, why do we even care to make a multi-threading source?

A3. Other 3 wont be killed ... just suspended and made to wait in a queue ... There is only one processor and it can execute only one control flow at a time. A higher pri implies that it is imp to execute this before anything else. Thats fair.

>>Typically if there are n priority levels, you would have n queues. OS would look for threads in highest priority queue.

Q4. levels range 0 to 31, so level 31 is the highest priority. Every level has a queue. Thus, a multi-threading application makes a queue for every level. Level 31 can have a queue of 10 threads, level 30 can have 4 threads, and so on. Level 31 has 10 threads - these 10 threads starts from 0 to 9, and system goes from thread 9 to 0. Once finished, system takes the next level queue, which is 30 & so on. I hope this right. Correct?

A4. Thats correct ... a slight clarification ... all threads in the same priority queue are given CPU slices in RR fashion until all of them have completed. Only then scheduler would move to the next level. Also, the queuing is not done by application - there is a system wide queue maintained by the OS. (so queue is done by OS).

"Each process is started with a single thread, often called the primary thread, but can create additional threads from any of its threads."

Q5. The primary thread is "int main()". Right? As without int main(), other threads wouldn't exist.

A5. correct. When you create an application. It runs within it's own primary thread. Every application has atleast 1 thread. When the primary thread terminates. All child-threads are destroyed and the application terminate.

"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.)"

Q6. Is this zero-page thread is created immediately after the a program is executed (process occurred)? Is every running process in a machine has its own a zero-page thread?

A6. The zero-page thread is a system thread - not per process. It's created @ boot time by the system. Not something you work with.

Q7. The priority level range from 0 to 31. If 0 is a system thread, then why don't we say a process

(program) have priority from 1 to 30! since 0 is a system thread. system thread is created when OS boots...any suggestions?

A7. Wouldn't you like to know about level 0? Even if you cant use it, it makes the picture complete.

Q8. Is base priority has number 0? Or base priority has to be specified by the developer?

A8. Base priority would not be 0 .. It would be some number ... there would be some default value for it and developers can specify priority using provided APIs

Q9. Every level can carry how many threads? 0 to 31 levels ...

A9. Implementation specific.

Q10. When is base priority occurs? When does it take place, why base priority is needed?

A10. Base priority is defined at the time of creation of thread. It is needed to determine what queue this newly created thread belongs to.

Q11. "These are known as ready threads". What are the not ready threads? Can you give an example of situation for a real life application please?

A11. Sleep, Wait, Finished are not-Ready threads. Often you will sleep a thread or Wait one to sync it with another thread.

Q12. "When a processor becomes available, the system performs a context switch." explain...

A12. Context-Switch means it'll load the register values, variable information, stack etc for that Thread to run on the CPU.

Multi-threaded development is exponentially harder than normal application development. And harder again when you move into inter-process communication to have applications spanning across CPUs. Mistakes are much more brutal, and debugging/profiling is alot harder into multi-thread space. Just be careful & take it slowly but surely...

Main source of threads: [http://msdn.microsoft.com/en-us/library/ms684841\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms684841(VS.85).aspx)

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Phew, this was some fairly deep stuff...Keep looking around for few more to come ?

PurpleHaze - December 27, 2008, 7:22 am
Whats this?

?si?qqo? - December 27, 2008, 7:26 am
(Dec 27, 2008 08:22 AM)PurpleHaze Wrote: [\[CITE\]](#) Whats this?
I just posted the thread! Read it so you know whats this...?

