

Windows Programming Job Interview Questions And Answers



Interview Questions Answers

<https://interviewquestionsanswers.org/>

About Interview Questions Answers

Interview Questions Answers . ORG is an interview preparation guide of thousands of Job Interview Questions And Answers, Job Interviews are always stressful even for job seekers who have gone on countless interviews. The best way to reduce the stress is to be prepared for your job interview. Take the time to review the standard interview questions you will most likely be asked. These interview questions and answers on Windows Programming will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts.

If you find any **question or answer** is incorrect or incomplete then you can **submit your question or answer** directly with out any registration or login at our website. You just need to visit [Windows Programming Interview Questions And Answers](#) to add your answer click on the *Submit Your Answer* links on the website; with each question to post your answer, if you want to ask any question then you will have a link *Submit Your Question*; that's will add your question in Windows Programming category. To ensure quality, each submission is checked by our team, before it becomes live. This [Windows Programming Interview preparation PDF](#) was generated at **Wednesday 29th November, 2023**

You can follow us on FaceBook for latest Jobs, Updates and other interviews material.
www.facebook.com/InterviewQuestionsAnswers.Org

Follow us on Twitter for latest Jobs and interview preparation guides.
<https://twitter.com/InterviewQA>

If you need any further assistance or have queries regarding this document or its material or any of other inquiry, please do not hesitate to contact us.

Best Of Luck.

Interview Questions Answers.ORG Team
<https://InterviewQuestionsAnswers.ORG/Support@InterviewQuestionsAnswers.ORG>



Windows Programming Interview Questions And Answers Guide.

Question - 1:

What is a thread?

Ans:

A thread describes a path of execution within a process. Every time a process is initialized, the system creates a primary thread. This thread begins executing with the C/C++ run-time library's startup code, which in turn calls your entry-point function (`main` , `Wmain` , `WinMain` , or `WWinMain`) and continues executing until the entry-point function returns and the C/C++ run-time library's startup code calls `ExitProcess`

[View All Answers](#)

Question - 2:

What is Mutex Object and why it is used?

Ans:

A mutex object is a synchronization object whose state is set to signaled when it is not owned by any thread, and non-signaled when it is owned. For example, to prevent two threads from writing to shared memory at the same time, each thread waits for ownership of a mutex object before executing the code that accesses the memory. After writing to the shared memory, the thread releases the mutex object.

[View All Answers](#)

Question - 3:

How the handles are handled in the child process?

Ans:

The operating system creates the new child process but does not allow the child process to begin executing its code right away. Of course, the system creates a new, empty process handle table for the child process just as it would for any new process. But because you passed `TRUE` to `CreateProcess`'s `blInheritHandles` parameter, the system does one more thing: it walks the parent process's handle table, and for each entry it finds that contains a valid inheritable handle, the system copies the entry exactly into the child process's handle table. The entry is copied to the exact same position in the child process's handle table as in the parent's handle table.

[View All Answers](#)

Question - 4:

What about the usage count in the parent child process tables?

Ans:

The system increments the usage count of the kernel object because two processes are now using the object. For the kernel object to be destroyed, both the parent process and the child process must either call `CloseHandle` on the object or terminate.

[View All Answers](#)

Question - 5:

What do you mean by unnamed object?

Ans:

When you are creating the kernel objects with the help of API's like `CreateMutex`(, , , `pzname`). And the `Pzname` parameter is `NULL` , you are indicating to the system that you want to create an unnamed (anonymous) kernel object. When you create an unnamed object, you can share the object across processes by using either inheritance or `DuplicateHandle`

[View All Answers](#)

Question - 6:

What is the limit on per process for creating a thread?

Ans:

The number of threads a process can create is limited by the available virtual memory and depends on the default stack size



[View All Answers](#)

Question - 7:

What is Event Object and why it is used?

Ans:

Event is the thread synchronization object to set signaled state or non-signaled state.

[View All Answers](#)

Question - 8:

APIs for creating event and set and reset the events?

Ans:

CreateEvent- to create the event

OpenEvent - to open already created event

SetEvent - to set the event signaled state

ResetEvent - To set the Event To non-Signaled State

[View All Answers](#)

Question - 9:

How do I create a Mutex?

Ans:

A thread uses the CreateMutex function to create a mutex object. The creating thread can request immediate ownership of the mutex object and can also specify a name for the mutex object

[View All Answers](#)

Question - 10:

How do other threads own the mutex?

Ans:

A semaphore object is a synchronization object that maintains a count between zero and a specified maximum value. The count is decremented each time a thread completes a wait for the semaphore object and incremented each time a thread releases the semaphore. When the count reaches zero, no more threads can successfully wait for the semaphore object state to become signaled. The state of a semaphore is set to signaled when its count is greater than zero, and non-signaled when its count is zero. The semaphore object is useful in controlling a shared resource that can support a limited number of users. It acts as a gate that limits the number of threads sharing the resource to a specified maximum number. For example, an application might place a limit on the number of windows that it creates. It uses a semaphore with a maximum count equal to the window limit, decrementing the count whenever a window is created and incrementing it whenever a window is closed. The application specifies the semaphore object in call to one of the wait functions before each window is created. When the count is zero - indicating that the window limit has been reached - the wait function blocks execution of the window-creation code.

[View All Answers](#)

Question - 11:

How owns the Kernel Object?

Ans:

Kernel objects are owned by the kernel, not by a process

[View All Answers](#)

Question - 12:

Which is the data member common to all the kernel object and what is the use of it?

Ans:

The usage count is one of the data members common to all kernel object types

[View All Answers](#)

Question - 13:

What is the purpose of Process Handle Table?

Ans:

When a process is initialized, the system allocates a handle table for it. This handle table is used only for kernel objects, not for User objects or GDI objects. When a process first initializes, its handle table is empty. Then when a thread in the process calls a function that creates a kernel object, such as CreateFileMapping, the kernel allocates a block of memory for the object and initializes it; the kernel then scans the process's handle table for an empty entry

[View All Answers](#)

Question - 14:

What is handle?

Ans:

Handle value is actually the index into the process's handle table that identifies where the kernel object's information is stored.

[View All Answers](#)

**Question - 15:**

What happens when the CloseHandle(handle) is called?

Ans:

This function first checks the calling process's handle table to ensure that the index (handle) passed to it identifies an object that the process does in fact have access to. If the index is valid, the system gets the address of the kernel object's data structure and decrements the usage count member in the structure; if the count is zero, the kernel destroys the kernel object from memory.

[View All Answers](#)

Question - 16:

What is the need of process relative handles?

Ans:

The most important reason was robustness. If kernel object handles were system-wide values, one process could easily obtain the handle to an object that another process was using and wreak havoc on that process. Another reason for process-relative handles is security. Kernel objects are protected with security, and a process must request permission to manipulate an object before attempting to manipulate it. The creator of the object can prevent an unauthorized user from touching the object simply by denying access to it

[View All Answers](#)

Question - 17:

Why the entries in the parent process table and child table are same?

Ans:

It means that the handle value that identifies a kernel object is identical in both the parent and the child processes.

[View All Answers](#)

Question - 18:

What are Named Objects?

Ans:

Method available for sharing kernel objects across process boundaries is to name the objects. Below are the kernel named objects:

- 1) mutex,
- 2) Events,
- 3) semaphore,
- 4) waitableTimers,
- 5) file mapping,
- 6) job object.

There are APIs to create these objects with last parameter as the object name.

[View All Answers](#)

Question - 19:

What is DuplicateHandle (API)?

Ans:

Takes an entry in one process's handle table and makes a copy of the entry into another process's handle table

[View All Answers](#)

Question - 20:

What is Synchronization Objects?

Ans:

Synchronization objects are used to co-ordinate the execution of multiple threads. Which kernel objects are used for Thread Synchronization on different processes? - Event, Mutex, Semaphore

[View All Answers](#)

Question - 21:

What is signaled and non signaled state?

Ans:

An event in a signaled state means that it has the capacity to release the threads waiting for this event to be signaled. An event in a non-signaled state means that it will not release any thread that is waiting for this particular event. Example in our project: when a user clicks the image application icon double simultaneously. Two image application windows were created, so PAIG created an event and set it to non-signaled state. Then the image application will reset the event to signaled state, after this all the threads are released.

[View All Answers](#)

Question - 22:

How does the kernel object outlive the process that created it?

Ans:

If your process calls a function that creates a kernel object and then your process terminates, the kernel object is not necessarily destroyed. Under most circumstances, the object will be destroyed; but if another process is using the kernel object your process created, the kernel knows not to destroy the object until the other process has stopped using it



[View All Answers](#)

Question - 23:

How to identify the difference between the kernel object and user object?

Ans:

The easiest way to determine whether an object is a kernel object is to examine the function that creates the object. Almost all functions that create kernel objects have a parameter that allows you to specify security attribute information.

[View All Answers](#)

Question - 24:

Name few functions that create Kernel Objects?

Ans:

HANDLE CreateThread(...),HANDLE CreateFile(..),HANDLE CreateFileMapping(..)HANDLE CreateSemaphore(..)etcAll functions that create kernel objects return process-relative handles that can be used successfully by any and all threads that are running in the same process.

[View All Answers](#)

Question - 25:

What is a kernel object?

Ans:

Each kernel object is simply a memory block allocated by the kernel and is accessible only by the kernel. This memory block is a data structure whose members maintain information about the object. Some members (security descriptor, usage count, and so on) are the same across all object types, but most are specific to a particular object type. For example, a process object has a process ID, a base priority, and an exit code, whereas a file object has a byte offset, a sharing mode, and an open mode.

[View All Answers](#)

Question - 26:

You forget to call CloseHandle - will there be a memory leak?

Ans:

Well, yes and no. It is possible for a process to leak resources (such as kernel objects) while the process runs. However, when the process terminates, the operating system ensures that any and all resources used by the process are freed-this is guaranteed. For kernel objects, the system performs the following actions: When your process terminates, the system automatically scans the process's handle table. If the table has any valid entries (objects that you didn't close before terminating), the system closes these object handles for you. If the usage count of any of these objects goes to zero, the kernel destroys the object.

[View All Answers](#)

Question - 27:

How the handle helps in manipulating the kernel objects?

Ans:

Whenever you call a function that accepts a kernel object handle as an argument, you pass the value returned by one of the Create* functions. Internally, the function looks in your process's handle table to get the address of the kernel object you want to manipulate and then manipulates the object's data structure in a well-defined fashion.

[View All Answers](#)

Question - 28:

User can access these kernel objects structures?

Ans:

Kernel object data structures are accessible only by the kernel

[View All Answers](#)

Question - 29:

If we cannot alter these Kernel Object structures directly, how do our applications manipulate these kernel objects?

Ans:

The answer is that Windows offers a set of functions that manipulate these structures in well-defined ways. These kernel objects are always accessible via these functions. When you call a function that creates a kernel object, the function returns a handle that identifies the object.

[View All Answers](#)

Question - 30:

What are types of kernel objects?

Ans:

Several types of kernel objects, such as access token objects, event objects, file objects, file-mapping objects, I/O completion port objects, job objects, mailslot objects, mutex objects, pipe objects, process objects, semaphore objects, thread objects, and waitable timer objects.

[View All Answers](#)

Operating System Most Popular & Related Interview Guides

- 1 : [RTOS Interview Questions and Answers.](#)
- 2 : [Windows 7 Interview Questions and Answers.](#)
- 3 : [MAC Operating System Interview Questions and Answers.](#)
- 4 : [Disk Operating System \(DOS\) Interview Questions and Answers.](#)
- 5 : [Shell Scripting Interview Questions and Answers.](#)
- 6 : [Operating System \(OS\) Interview Questions and Answers.](#)
- 7 : [Solaris Interview Questions and Answers.](#)
- 8 : [Solaris Admin Interview Questions and Answers.](#)
- 9 : [VxWorks Interview Questions and Answers.](#)
- 10 : [OS Memory Management Interview Questions and Answers.](#)

Follow us on FaceBook

www.facebook.com/InterviewQuestionsAnswers.Org

Follow us on Twitter

<https://twitter.com/InterviewQA>

For any inquiry please do not hesitate to contact us.

Interview Questions Answers.ORG Team

[https://InterviewQuestionsAnswers.ORG/
support@InterviewQuestionsAnswers.ORG](https://InterviewQuestionsAnswers.ORG/support@InterviewQuestionsAnswers.ORG)