**­­C++ Interview questions**

1. C++ Core concepts
   1. Constructor and destructors
   2. Operator overloading and function overloading
   3. Virtual class and friend class
   4. Polymorphism
   5. Virtual functions
   6. Virtual destructors
   7. Abstract classes
   8. File operations
   9. Constant, mutable
   10. Templates
   11. RTTI
2. STL
   1. Vector
   2. List
   3. Map
   4. Hash table
   5. unordered\_map, unordered\_set
3. Class diagram for your project
4. Design Patterns, Provide use cases
   1. Creational
      1. Singleton design pattern
      2. Abstract factory
   2. Structural
      1. Proxy and Adaptor
      2. Composite
   3. Behavioral
      1. Observer
      2. visitor
5. Multi-threading
   1. Semaphore
   2. Mutex

**Mutex**: Mutual exclusion, for synchronization of two processes, any of this two process can have KEY (MUTEX) to proceed there work, implemented as locking mechanism.

**Semaphore**: Generalization of mutex, implemented as signaling mechanism

**Binary semaphore**:

**Main Difference between mutex and semaphore**

Mutex is Locking Mechanism.

Semaphore is Signaling mechanism.

**Recursive and Non recursive mutex:**

**Shared Mutex**

**Read/Write mutex**

**Spin locks**

**C++ Mutex types**

1. **std::mutex**
2. **std::timed\_mutex**
3. **std::recursive\_mutex**
4. **std::recursive\_timed\_mutex**
5. **std::shared\_timed\_mutex**

**C++ Lock types**

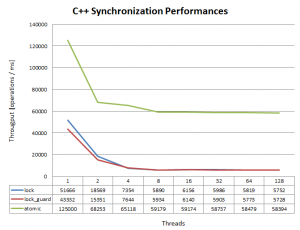
1. **std::lock\_guard<>**
2. **std::unique\_lock<>**
3. **std::shared\_lock<>**

**Performance comparisons**

A single std::mutex with calls to lock() and unlock()

A single std::mutex locked with std::lock\_guard

An atomic reference on the integer



Ref : <http://baptiste-wicht.com/posts/2012/04/c11-concurrency-tutorial-advanced-locking-and-condition-variables.html>

C++ /OOPs interview Questions

1. What is OOP?

* Philosophy of design and implementation, which is based on concept of object and classes.
* Programming language Model, organized around object
* Historically programming has been viewed as logical procedure
* It is not procedural programming.

1. What are building blocks of OOP?
   1. Classes and objects
   2. Encapsulation
   3. Data Abstraction
   4. Inheritance
   5. Polymorphism
2. Explain classes and objects in details.
   1. Concept

* It is just a blueprint
* Defines basic characteristics and behaviors of similar kind of data.
  1. Give one/two theoretical example (simple bookish example)
  2. Give one/two classical example (Practical / real example)
  3. Give one/two example from your project (Actual example on which you worked)

1. Explain Encapsulation in details.
   1. Concept

* Data hiding
  1. Give one/two theoretical example (simple bookish example)
  2. Give one/two classical example (Practical / real example)
  3. Give one/two example from your project (Actual example on which you worked)

1. Explain abstraction in details.
   1. Concept

* Exposing only essential details and hiding other irrelevant details.
* Reduces code complexity
  1. Give one/two theoretical example (simple bookish example)
  2. Give one/two classical example (Practical / real example)
  3. Give one/two example from your project (Actual example on which you worked)

1. Explain Inheritance in details.
   1. Concept

* Derive new type of objects from existing objects.
  1. Give one/two theoretical example (simple bookish example)
  2. Give one/two classical example (Practical / real example)
  3. Give one/two example from your project (Actual example on which you worked)

1. Explain polymorphism in details.
   1. Concept
   2. Give one/two theoretical example (simple bookish example)
   3. Give one/two classical example (Practical / real example)
   4. Give one/two example from your project (Actual example on which you worked)
2. What is mean by Generalization?
   1. Concept
   2. Give one/two theoretical example (simple bookish example)
   3. Give one/two classical example (Practical / real example)
   4. Give one/two example from your project (Actual example on which you worked)
3. What is mean by Specialization?
   1. Concept
   2. Give one/two theoretical example (simple bookish example)
   3. Give one/two classical example (Practical / real example)
   4. Give one/two example from your project (Actual example on which you worked)
4. What is mean by Composition?
   1. Concept
   2. Give one/two theoretical example (simple bookish example)
   3. Give one/two classical example (Practical / real example)
   4. Give one/two example from your project (Actual example on which you worked)
5. What is mean by Association?
   1. Concept
   2. Give one/two theoretical example (simple bookish example)
   3. Give one/two classical example (Practical / real example)
   4. Give one/two example from your project (Actual example on which you worked)
6. What is mean by Aggregation?
   1. Concept
   2. Give one/two theoretical example (simple bookish example)
   3. Give one/two classical example (Practical / real example)
   4. Give one/two example from your project (Actual example on which you worked)
7. What are types of inheritance? Give practical example of each type.
8. What are types of polymorphism? Give practical example of each type.
9. What is copy constructor and copy assignment operator?
10. What is mean by shallow copy and deep copy? When to use which copy?
11. What are destructors? Why we need to use virtual destructor?
    1. Case when you have multiple inheritance
    2. Case when you multilevel inheritance
    3. Case when you have hybrid inheritance
    4. So each all above cases, which call destructors, you will define virtual?
12. What are pointers, what are references?
13. Explain following concepts along with example.
    1. Smart pointers
    2. Unique pointers
    3. Shared pointers
    4. Auto pointers
    5. Dangling pointers
    6. Void pointer in C vs void pointer in C++
14. Difference between malloc and new, free and delete, with example.
15. Allocate memory for 1d array, 2d array using new operator.
16. What is stack memory and heap memory?
17. What is stack unwinding?
18. Operator overloading examples
    1. Pre increment and post increment overloading
    2. New/delete operator overloading
    3. Operator overloading with help of friend functions.
    4. Conversion operator
    5. Conversion constructor
19. What are namespaces? What is koening lookup? How it is useful?
20. What are exception? How to handle exceptions? Have you used them any of your programs? Please explain with examples
21. What are templates?
    1. What are function templates
    2. What are class templates
    3. Write a program for class template
    4. Write a program for function template.
22. What are all string operations?
    1. Strrev
    2. Strcpy
    3. Strstr
    4. Substr
23. What is RTII?
24. Explain each cast.
25. What is RAII?

Multithreading and Multi Processing and OS Concepts.

1. What is process?

* It is an instance of program in execution. It has following major sections
* Data: Global and static variables, allocated and initialized prior execution of program.
* Text or Code: Compiled code, instructions or op-codes.
* BSS section: Uninitialized variables.
* Stack: local variables, function calls context.
* Heap: dynamic memory allocation, new, malloc, free delete etc.
* Process control block: contains information about each process.
* A process is just container for its threads.

1. What is thread?

* A basic unit of CPU utilization
* A light weight process.
* Multithreaded application have multiple threads within single process, each having their own program counter, stack and set of registers, but sharing common code, data and certain structures such as open files.
* All threads within a process share same address space and OS resources.

1. What is process control block? What are elements of process control block?

* Program counter and value of all program registers.
* Memory management information
* Accounting information such as time used, ID
* I/O status : list of open files, devices
* Scheduling data : priority
* Process state: New, ready, running, waiting, suspended, terminated.
* Linux PCB (task\_struct) has 106 fields.

1. What is thread control block? What are elements of thread control block?

* Program counter
* Registers.
* Linux TCB (thread\_struct) has 24 fields.
* A pointer to parent process.

1. What are different process states?

* In 5 state process model, following are the states
  + New
  + Running
  + Ready
  + Waiting
  + Terminated

1. What are similarities and differences between process and thread?

* Similarities
  + Both share CPU, only one thread active at a time.
  + Both can create child.
  + If one is block another can run.
* Differences
  + Threads are not independent of one another
  + All thread can access any address within task.
  + Threads are designed to assist one another.
  + Thread cannot have independent existence.

1. How to create Process in C program?

* Following are the ways to create new process in Linux.
  + fork:
  + clone:
  + popen:
  + system:
  + execve:
  + pipe:

1. How to create thread in C program?

* pthread\_create()

1. What is Zombie process? How to create zombie process? How to control it?

* A zombie or defunct process is a process that has completed execution but still has an entry in process table.
* This entry is still needed to allow the parent process to read its child’s exit status.
* Kill command has no effect on zombie process.
* When a process ends, all of the memory and resources associated with it are deallocated so they can be used by other processes. However, the process's entry in the process table remains. The parent can read the child's exit status by executing the wait system call, whereupon the zombie is removed.
* Parent process need to use wait call, or SIGCHLD, to get notification for child.

1. What is Orphan process? How to create orphan process? How to control it?

* An orphan process is a computer process whose parent process has finished or terminated, though it is running itself.
* It will be adopted by init system process. The reparenting operation occurs automatically.
* It can be intentionally or unintentionally.

1. What is Daemon process? How to create daemon process? How to control it?

* Daemon is orphan process created intentionally.
* It runs in background, it not in direct control of user interaction.
* In Linux, typically daemon names ends with d e.g. sshd, syslogd etc.

1. What are different segment?

* Text/Code/Instruction segments: It is read only, and initialized from program executable file.
* User data segment
  + Initialized data – string initialized from program executable.
  + Uninitialized data – global variables, these are set to 0
  + Malloc area or heap – Created by process
  + Stack – Auto variables, function parameters
* System data segment
  + Open file descriptors

1. What is BSS segment?

* It is **B**lock **S**tarted by **S**ymbol segment
* Uninitialized global and static variables.

1. In which segment Global variable stored?

* If global variable is initialized, it will be stored in initialized data section.
* If global variable is uninitialized, it will be stored in BSS segment.

1. In which segment dynamically allocated memory is stored?

* Dynamically allocated memory is stored in heap section.

1. What is Re-entrant code? What is thread-safe code? What are similarity and difference between re-entrant code and thread-safe code?

* In single threaded processes, only one flow of control exists, the code executed by these processes thus need not be re-entrant or thread-safe. In multithreaded programs, the same functions and the same resources may be accessed concurrently by several flows of control.
* To protect resource integrity, code written for multithreaded programs must be re-entrant and thread-safe.
* Re-entrant and thread safety are both related to the way that functions handle resources.
* Re-entrant and thread-safety are separate concepts
* A function can be either re-entrant, thread-safe, both or neither.
  + Re-entrant functions
    - Does not hold static data over successive calls, nor it returns pointers to static data.
    - All data is provided by caller function
    - A re-entrant function must not call non re-entrant functions.
  + Thread Safety
    - A thread safe function protects shared resources from concurrent access by locks.
    - Thread safety concerns only the implementation of a function and does not affect its external interface.
    - Any function that does not use static data or other shared resources is trivially thread safe.
    - Use of global data is thread unsafe.
    - Global data should be maintained per thread or encapsulated, so that its access can be serialized.
  + Converting non reentrant function to re-entrant
    - In many cases, non-reentrant function must be replaced with a modified interface to be re-entrant
    - Non-reentrant function cannot be used by multiple threads.
    - So, non-reentrant functions are impossible to be thread safe.
  + Making a function thread safe
    - Making a function re –entrant also makes them thread safe.
    - Locking shared resources
      * Functions that use static data or any other shared resources such as files or terminals must serialize the access to these resources by locks in order to be thread safe.

1. What is difference between logical and physical address space?

* An address generated by CPU is logical address or virtual address.
* Address actually available on memory unit is physical address.
* Virtual and physical address are same in compile time and load time address binding.
* Virtual and physical address differs in execution time address binding schemes.
* The set of all logical address generated by program is referred as logical address space.
* The set of all physical address corresponding to logical address is referred as physical address space.
* Run time mapping from virtual to physical address and vice versa is done by Memory Management unit (MMU).

1. What is swapping?

* Swapping is a mechanism in which a process can be swapped temporarily out of main memory to a backing store and then brought into memory for continued execution.

1. What is fragmentation?

* As processes are loaded and removed from memory, free memory space is broken into little pieces. It happens after some time that processes cannot be allocated to memory blocks considering their small size and memory blocks remained unused. This problem is known as fragmentation.

1. What is external fragmentation?

* Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so it cannot be used. This is called as external fragmentation.

1. What is internal fragmentation?

* Memory blocks assigned to process are bigger. Some portion of memory is left unused as it cannot be used by other process.

1. What is paging?

* External fragmentation is avoided by using paging techniques.
* A paging is techniques in which physical memory is broken into blocks of same sizes called as pages. When a process is to be executed, its corresponding pages are loaded into available memory frames.

1. What is page thrashing?

* When a page is not required in main memory, it would be moved out of main memory, this is called page thrashing.

1. What are interrupts?
2. What is Deadlock?
3. What are deadlock detection algorithms?
4. What is starvation?
5. How to resolve Deadlock?
6. How to prevent Deadlock?
7. How you will decide, whether to use multithreading or multiprocessing for a program?

* Whenever a process has multiple tasks to perform independently of others, go for multithreaded application.
* When in program, if one task is blocked, other tasks need to proceed without blocking, go for multi-threaded application.

1. Provide advantages and disadvantages of multi-threading and multiprocessing.

* ADVANTAGES OF MULTITHREADING
  + **Responsiveness** - One thread may provide rapid response while other threads are blocked or slowed down doing intensive calculations.
  + **Resource sharing** - By default threads share common code, data, and other resources, which allows multiple tasks to be performed simultaneously in a single address space.
  + **Economy** - Creating and managing threads (and context switches between them) is much faster than performing the same tasks for processes.
  + **Utilization of multiprocessor architectures** - A single threaded process can only run on one CPU, no matter how many may be available, whereas the execution of a multi-threaded application may be split amongst available processors.

1. What are different inter process communication mechanisms?
2. What are different inter thread communication mechanisms?
3. What are Volatile variables?
4. What are different thread attributes?
5. What is concurrency?

* Concurrent means something that happens at the same time as something else.
* Concurrency is tendency for things to happen at the same time in a system.
* Concurrency is interleaving of processes in time to give the appearance of simultaneous execution. It differs from parallelism, which offers genuine simultaneous execution.

1. What is parallelism?
2. What is semaphore?
3. What is mutex?
4. How to achieve synchronization in following cases?
   1. Single read thread and single write thread.
   2. Single write thread and multiple read threads.
   3. Multiple write threads and multiple read threads.
5. What are different synchronization mechanisms? Explain each with detail example.
6. What are different types of semaphores?
7. What are different types of mutex?
8. What is difference between semaphore, mutex and binary semaphore?
9. How to know how much memory used by process?
10. What is difference between compiler and interpreter?
11. What is marshaling?
12. What is priority inversion?
13. What is context switching?
14. What is process spawning?
15. What is difference between socket and pipe?
16. What are types of scheduling?
17. Explain short, long and medium term scheduling.
18. What is busy waiting?
19. What are popular multi-processor thread-scheduling strategies? Explain following strategies.
20. Load sharing
21. Gang scheduling
22. Dedicated processor assignments
23. Dynamic scheduling
24. What are reasons for process suspensions/termination?

* Swapping, interactive user request, timing, parent process request.
* Normal completion
* Memory unavailable
* Protection error
* Privileged instruction
* Human intervention
* Parent request
* Exceed time limit
* Bounds violation
* Arithmetic error
* Invalid Instruction
* Parent termination
* I/O failure.

1. What is process migration?
2. Explain memory partitions, paging and segmentation.
3. What is monitor?
4. What is difference between semaphore and monitor?