Wintermute

2022 Sep09 *16:00-17:30*

1. What is the use of anonymous namespace?

2. About `inline` and `constexpr`
- How to force compiler to do inline?
- How to force compile to do compile time `constexpr`?
- What is `constexpr`?

3. What is object slicing?
- Its about copy assignment between base and derived class

4. How does virtual function work?
- level 1 dereference : virtual table pointer in object pointing to virtual table in class
- level 2 dereference : read the corresponding function pointer in the virtual table

5. What are the different ways to do capture in lambda?
- capture a variable by value, capture all by value
- capture a variable by reference, capture all by reference
- capture `this` pointer

What is the disadvantage of capture by reference? Here is the case (provided by interviewer)

```
std::function<void()> f;
{
     int a = 123;
     f = [&a](){ std::cout << a; };
}
f(); // theoretically crash, but surprisingly, it works in online gcc
```

6. About hash table in STL
- How does it handle collision? *using link list rather than open address (according to what interviewer said)*
- What happen to iterator after rehashing? Is there anyway to remain valid after rehashing?
- If STL does use link list under the hood, how to make it cache friendly? *Using custom allocator, probably an array*

7. How is `std::string` implemented? What are the private members?
- it has a char pointer, a capacity and a size
- it has short string optimization for very short string

8. Single thread vs multi thread on reading multiple sockets
- when there are 1000 sockets, we need to read some them, how to do it?
- using one thread for each socket, each thread call synchronous read
- using one thread for all sockets, the thread call asynchronous read (with callbacks)
- what is the advantage of former method?
- when one of the thread reads a huge data and processes it
- that thread is not allowed to use up all CPU resource
- as scheduler allows preemption of other threads
- what is the advantage of latter method?
- no context switching
- this tradeoff / struggling is called single thread paradigm

9. What are the differences between process and thread?
- one process has unique env var, file descriptor table, signal table, shared by all threads
- one process has unique VAS, shared by all threads
- one thread has unique call stack

When two threads in same process allocate memory from heap, what are the possible hazard?
- sharing VAS means one thread may use more than the other
- false sharing > cache ping pong > solved by padding or alignment

10. Why shared pointer is not good?

- it uses heap
- it uses two allocations : one for manager, one for resource, can be avoided by factory `std::make_shared`
- it may cause problem when forming cycle of shared pointer
- it may not be thread safe (it depends)
- multi thread reference count is ok
- multi thread reset is not ok

11. Why shared pointer reference count …

- is incremented with `memory_order_relaxed`
- but decremented with `memory_order_release`?
- the former is obvious, no matter which thread increment first, no one is going to deallocate the resource
- the latter is suspicious (is it necessary), as there is only one thread getting zero count …

12. Given 1 million integers all within range 0 to 63, what is the fastest sorting?

13. What is the size of struct :

```
struct A { char m0; double m1; };
struct B { double m0; char m1; };
```

Both are 16 bytes. Consider an array of A or an array of B.

14. How to make no space in between members in a structure?

- by directive `#pragma pack(push, 1)`
- it is useful for reinterpret cast of a buffer into a struct, particular in datafeed, without copying

Given a struct to model a tick in datafeed, if the first data in the struct is integer, can I just perform :

```
reinterpret_cast<tick*>(pc)->tick_id = xxx;
```

No, we need to take care of the endian-ness. Is the endian-less OS-dependent or CPU-dependent? I said the former, yet interview said the latter. Please check.

15. Why system call is slower? How to make it fast? Yes kernel bypass technique.