Bytes Programming Languages Computer Programming

Is a pointer always 4 bytes?

12 Answers

John L. Miller, Veteran programmer in C, C++, Java, Assembler, and pseudocode.
Answered Mar 6 2016 · Author has **2.3k** answers and **32m** answer views

No, a C / C++ pointer is not always four bytes.

In the normal case, the size of the pointer is determined by the architecture of the platform your compiler is running on. For example, a pointer on a 64-bit system will be 64-bits, which is 8 bytes. Having said that, pointers to intrinsic data and to objects should always be the same size on a given architecture.

There's a special case for C++ depending upon the *kind* of pointer. Specifically, whether it's a pointer to an object or to a non-static class method. I'm a little fuzzier on this and my knowledge is mostly based on C++ 98, so don't treat it as gospel. I *believe* that pointers to non-static functions in classes consist of two pointers, the class and the function. This would require twice as many bytes as a pointer to an intrinsic data type or object.

Related Questions More Answers Below

Why is the pointer size 2 bytes in C?

Why is pointer 8 bytes in 64-bit systems?

In C language, the integer takes 2 bytes for a 32-bit compiler and 4 bytes for a 64-bit compiler. The float always takes 4 bytes. The characte...

Why is int 4 bytes and char is 2 bytes?

Related Questions

Why is the pointer size 2 bytes in C?

Why is pointer 8 bytes in 64-bit systems?

In C language, the integer takes 2 bytes for a 32-bit compiler and 4 bytes for a 64-bit compiler. The float always takes 4 bytes. The characte...

Why is int 4 bytes and char is 2 bytes?

How it is possible for pointer to store address of more than 2 bytes?

Can I store 8 bytes of information into 4 bytes and reverse it back?

Is it possible to make a pointer to point to each bit in byte not byte by byte?

Why is the double pointer size 8 bytes while the pointer size is 4 bytes?

Why are pointers used in C/C++?

What is the value of a byte?

How it is possible for pointer to store address of more than 2 bytes?

Richard Rombouts, former Freelance C++ Software Engineer (2008-2019)
Answered Mar 19, 2016 · Author has **2.1k** answers and **581.8k** answer views

No. It depends on your operating system. Then hopefully you have the right compiler for your operating system. Nowadays, pointers are either 32 or 64 bit on most systems, but I wonder if this was the case in old Turbo C or Turbo Pascal. At that time they did not even have flat memory, but used segments. Sometimes you could get away with 16 bits for pointing in a 64KB block, at other times you needed to make the segment explicit. I don't recall what sizeof(some_type*) would have returned in those days. Pointers came in different sizes. Probably the "normal" pointer was 16 bits, while the "far pointer" was 32 bits.

You should try to write code that doesn't depend on the size of pointers, or else use sizeof().

Then remember a pointer is a concept, which is much wider applicable than for programming alone. We probably have kind of pointers in our head also, and I never needed to know their size. It's just a relatively boring implementation detail. Better focus on concepts if you want to understand programs or write good programs (correct, readable, easy to maintain).

Then your compiler is absolutely right. A pointer to void, int, long, long long, double, struct blablah, char* mystring, a function pointer etc, all have the same size. Remember, a pointer is just an address, not the contents itself. So the pointer only points to the data (hence the name), it doesn't contain the data. The number of bits needed for a pointer depends on the amount of memory you uniquely can identify. For 4GB, which is 2^2 * 1024 * 1024 * 1024 = 2^(2+10+10+10) = 2^32, you need 32 bits, which is the log (base 2) of 4GB. Then on modern systems, your data probably needs to be properly aligned: signed/unsigned short int's for instance on 16 bit boundaries, integers possibly on 4 byte boundaries (can be 64 on your system), doubles on 8 byte boundaries etc. Usually, sizeof(type) tells you how to align well. But I can think of one possible exception: You have 80 bits doubles also. These can go by the name of extended doubles, maybe long doubles. They take 10 bytes in memory, but I could

understand if you need to align them on 16 byte addresses. Computers love powers of 2.

1.7k views

David Hayden, 30 years of professional programming.
Answered Mar 9, 2016 · Author has **1k** answers and **1.3m** answer views

Sometimes different pointers are different sizes in the same program. The 8086 used segmented memory where memory was accessed via a 16 bit segment and a 16 bit offset. The physical address was computed as (segment << 4) + offset. Most pointers were just a 16 bit offset, but you could also specify a 32 bit segment-and-offset pointer.

Notice that more than 1 segment/offset pair can yield the same physical address. That means you had to be careful when comparing pointers.

I think the most unusual addressing scheme I've seen is the Saturn processor (and its predecessors) used in HP calculators until recently. Each address specified a 4-bit nibble instead of an 8-bit byte. The size of an address grew over the years, but the Saturn used a 20 bit address. Data was nibble-addressable because numbers were stored in BCD with one digit per nibble.

1.4k views · View 1 Upvoter

Related Questions

- Can I store 8 bytes of information into 4 bytes and reverse it back?
- Is it possible to make a pointer to point to each bit in byte not byte by byte?
- Why is the double pointer size 8 bytes while the pointer size is 4 bytes?

Why are pointers used in C/C++?

What is the value of a byte?

Who can help me with printing out an integer byte by byte using a pointer in C programming?

What kind of data types can offset a stack pointer by 7 bytes in x86?

What are some tips for someone who has a hard time using pointers, arrays of pointers, and pointers to pointers?

Are all variables pointers?

Is the value of a pointer type always an address?