C programming: casting a void pointer to an int?

Say I have a void* named ptr. How exactly should I go about using ptr to store an int? Is it enough to write

```
ptr = (void *)5;
```

If I want to save the number 5? Or do I have to malloc something to save it?

c pointers void

edited Oct 20 '11 at 20:27
Michael Küller
1,518 1 11 30

asked Oct 19 '11 at 21:28
Tim
1,111 5 18 36

2 Why, for the goodness sake, do you want to do this? – Griwes Oct 19 '11 at 21:30

2 What do you want to do? We need to see the bigger picture to help you. Now I can only guess that the code you posted is probably not doing what you would expect it to do. – Karel Petranek Oct 19 '11 at 21:31

3 Do you want a pointer to memory location 5 or to the number 5? There is a big difference between the two. – dbeer Oct 19 '11 at 21:36

6 Answers

You're casting `5` to be a void *pointer* and assigning it to `ptr` .

Now **ptr points at the memory address 0x5**

If that actually is what you're trying to do .. well, yeah, that works. You ... probably don't want to do that.

When you say "store an int" I'm going to guess you mean you want to actually store the integer value 5 in the memory pointed to by the `void*` . As long as there was enough memory allocated (`sizeof(int)`) you could do so with casting ...

```
void *ptr = malloc(sizeof(int));
*((int*)ptr) = 5;

printf("%d\n",*((int*)ptr));
```

edited Oct 19 '11 at 21:41

answered Oct 19 '11 at 21:36
Brian Roach
48.7k 4 63 96

2 I do this all the time, when working with a framework that stores "User Data" as a `void*` , but all I need is an int. – Mooing Duck Oct 19 '11 at 21:41

As long as someone after you doesn't accidentally dereference it later, I agree that works. It just makes me cringe a bit for that reason. – Brian Roach Oct 19 '11 at 21:50

And I knew there was something else that was bugging me - it's not portable. developer.gnome.org/glib/2.28/glib-Type-Conversion-Macros.html – Brian Roach Oct 19 '11 at 21:51

@MooingDuck Ah! That's exactly what I was running into. So it's a proper use case. Thanks. – Andy Oct 11 '14 at 19:57

Yes, what you have there will work just fine. Whether it's a good idea or not depends on what exactly you're trying to do, but there's no problem with that code as it is.

answered Oct 19 '11 at 21:30
Carl Norum
131k 16 224 316

That will work on all platforms/environments where `sizeof(void*) >= sizeof(int)` , which is probably most of them, but I think not all of them. You're not supposed to rely on it.

If you can you should use a union instead:

```
union {
    void *ptr;
    int i;
};
```

Then you can be sure there's space to fit either type of data and you don't need a cast. (Just don't try to dereference the pointer while its got non-pointer data in it.)

Alternatively, if the reason you're doing this is that you were using an int to store an address, you should instead use ~~`size_t`~~ **`intptr_t`** so that that's big enough to hold any pointer value on any platform.

edited Sep 16 '14 at 7:52

answered Oct 19 '11 at 21:40
Boann
20.3k 5 46 72

1 No, don't use `size_t` . It isn't guaranteed to be able to hold all possible addresses of pointers. `intptr_t` and `uintptr_t` are there for that since C99. In C90 you're out of luck. I can even give you a platform where `size_t` is 16 bits wide and pointers are 32 bits wide: x86 in real mode using memory model `HUGE` , `FAR` and `COMPACT` . `size_t` can be seen as the type used for addressing arrays, nothing more. – Patrick Schlüter Sep 16 '14 at 7:49

@tristopia Thank you! – Boann Sep 16 '14 at 7:52

Define "work".

It will certainly set `ptr` to the value of `5` , which means that it will point to memory location `0x5` , though it cannot be dereferenced without another cast because it is a typeless pointer.

If you later cast it to `something*` and dereference it you will probably just crash your program trying to access memory location `0x5` , if you're lucky.

I don't see the point of this at all, what are you trying to accomplish?

answered Oct 19 '11 at 21:43
Ed S.
78.9k 10 111 175

A pointer always points to a memory address. So if you want to save a variable with pointer, what you wanna save in that pointer is the memory address of your variable.

answered Oct 20 '11 at 9:34
Jo.Manurung
1

The `cast` is sufficient..................

edited Nov 18 '11 at 1:21
chown
29.7k 10 80 128

answered Oct 19 '11 at 21:30
James
6,041 1 8 26