& Produces Address at Which Expression is Stored

`&` the address operator

You have used address operator with **`scanf`**.

Address operator evaluates to
- **the address of an expression**
- (usually a variable).

```
char* cptr = "My favorite string";
```

For example, **`&cptr` evaluates to the address at which `cptr` is stored**.

& Only Usable with Expressions that Have Addresses

```
char* cptr = "My favorite string";
```

What about this one?

`&&cptr`

`&cptr` not known to be stored anywhere, so **expression above gives error**.

However, **`*(*(&cptr))`**

evaluates to 'M'.

Can Also Use Pointers to Pointers

```
char* cptr = "My favorite string";
```

What if we want to store **`&cptr`**?

What is the type?

Pointer to pointer to `char`.
(remember LDI/STI?)

So: **`char** cptr_ptr = &cptr;`**

And `**cptr` evaluates to what?

'M'

Understanding Pointers is Critical

How useful are pointers?

Rare to find anything but toy programs **that does not use pointers** (albeit hidden by many high-level languages).

How useful are pointers to pointers?

Useful in a wide range of applications; you will use them often (but as above, you may not know it).

