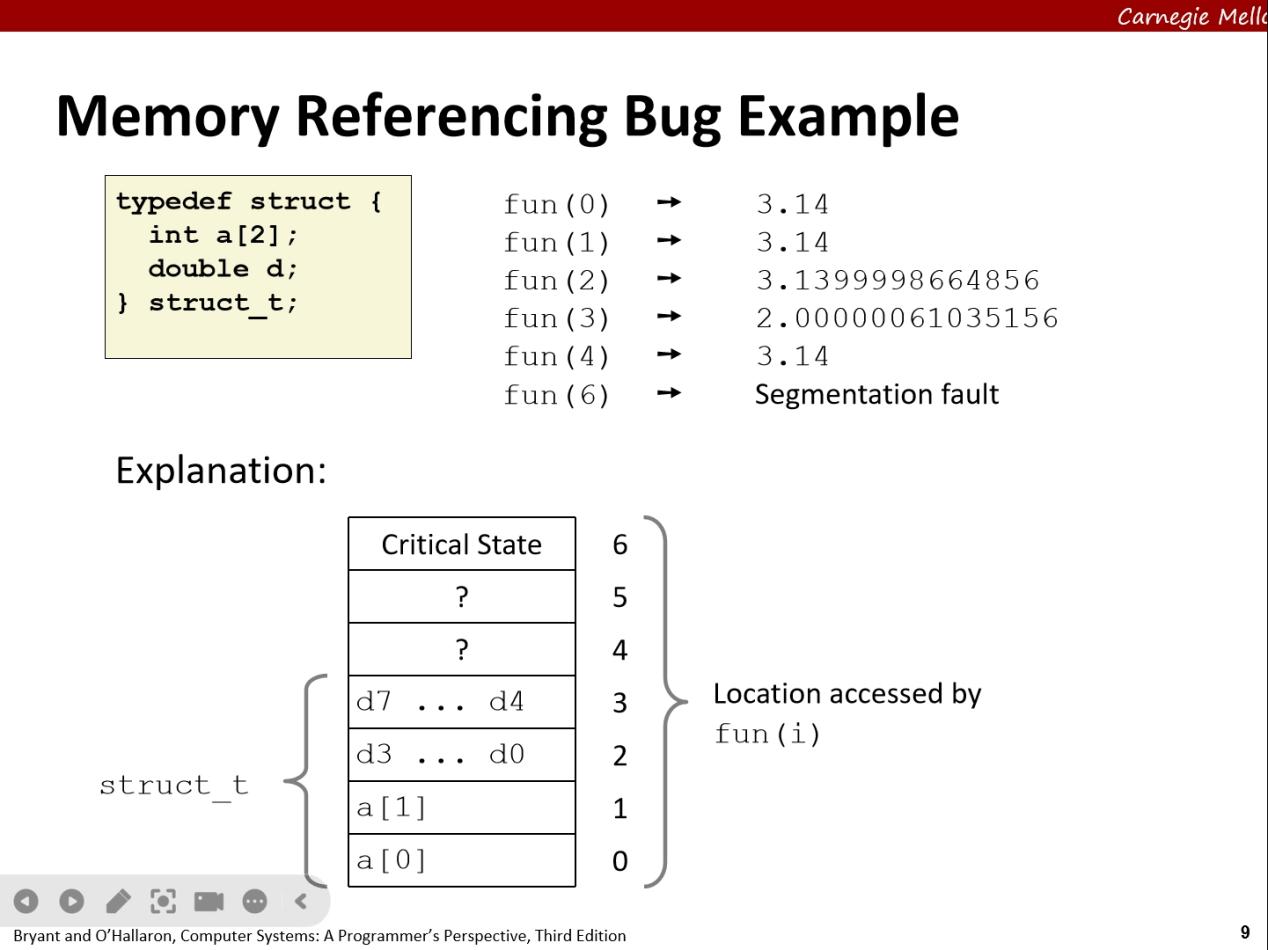
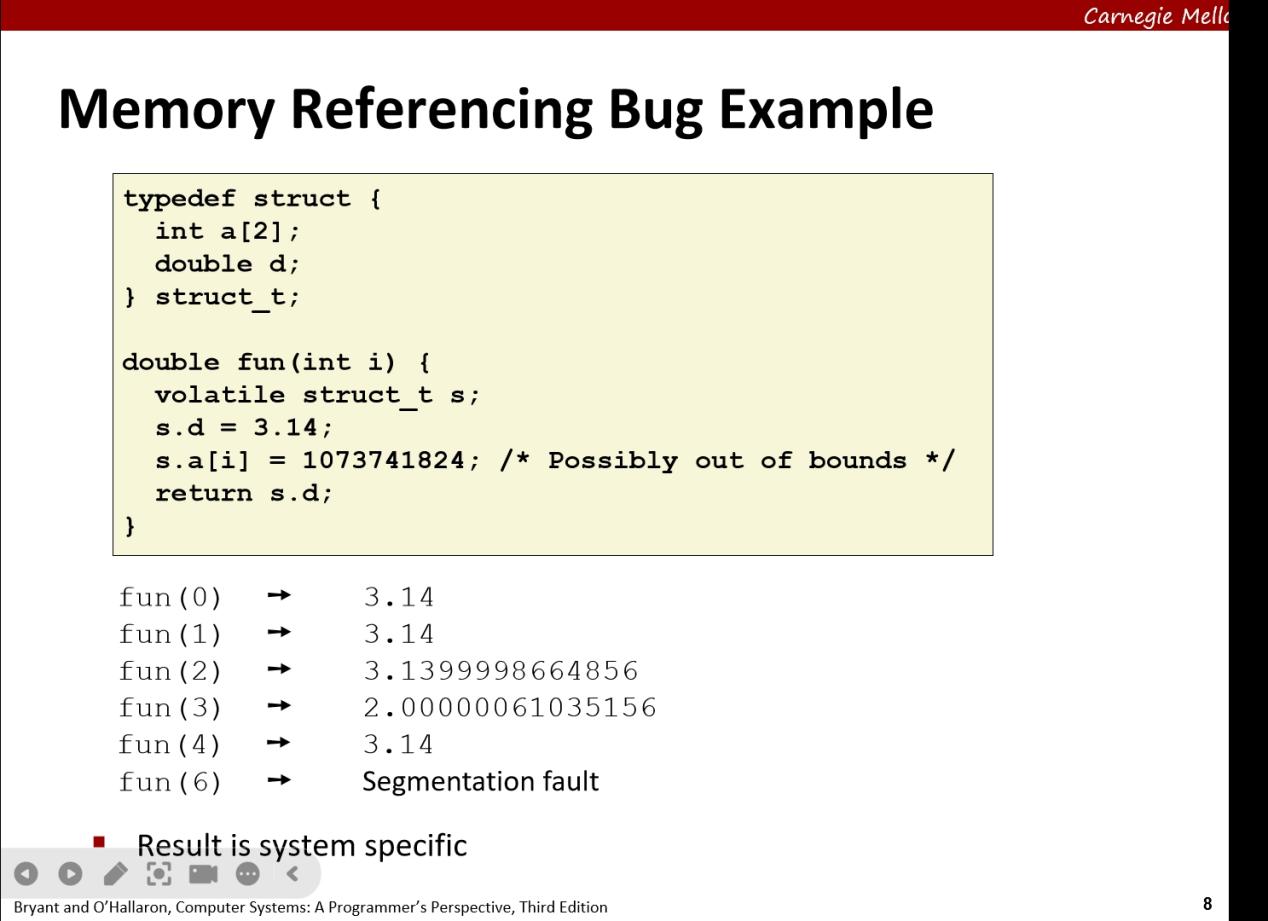
# 第一节学习



这里注意对于内存机制的理解

1~c语言是不检查数组传进来的下标是否合法(除非你自己配置了)

2~对于结构体，程序在内存中的分配是连续的，上面的看起来是一个栈，其实在虚拟地址上各个元素上的地址是连续的。

即a[0]->a[1]->d

这里需要注意字节对齐的规则，就比如上面那个的示例

结构体中的元素通常会按顺序存储，但**需要注意字节对齐**。结构体中的成员 地址在虚拟地址空间内是连续的，但内存布局可能因对齐规则（alignment） 而影响。例如，double 类型的数据通常需要在8字节对齐的地址上存储， 所以编译器可能会在 int[2] 和 double 之间插入一些填充字节。

3~这里的a储存的是int型数据，在这里姑且视为占4字节，d储存double数据，姑且视为 8字节，需要注意这里是一个连续储存的情况，且注意数组通过中则运算符的计算情况， 简单可以看做是数组首地址的偏移，由于程序不检查下标是否越界，所以这里计算的话 传入2/3...时会访问一些其他数据，当传入2/3时，由于字长原因，其访问的内存会是 d的内存空间，故这里会出现逻辑外的修改。实际上是修改了d所在的储存空间的元素

4~由于对内存的修改，故传入2/3时程序输出会很奇怪

5~至于4时的情况，这是一个未定义行为，故这个情况不好分析

6~对于崩溃的6情况，可能就是因为这个访问逻辑导致程序访问了一个未知的内存并想尝试 写入不允许写入的导致了程序的崩溃（可能是程序储存函数的指针什么的）

数组越界可能会访问到未分配的内存或只读内存区域，导致段错误（segmentation fault）或程序崩溃。这通常是因为越界访问修改了受保护的内存区域。

7~当然，就算程序在6处还是没有崩溃，但是这里的行为还是未知的，你不知道它到底修改 的数据会被怎么用，也不知道这个会在你之后的程序中导致什么后果