## sizeof()

- Compile-time operator*
- Compute the size of operand in **bytes** (8 bits = 1 byte)
  - Can compute size of **type** or **variable**
- May give different output depending on machine!
  - aarch64/arm64/x86_64 sizeof(int*) == 8
  - arm32/i386 sizeof(int*) == 4 (Usually in older computers)

## sizeof()

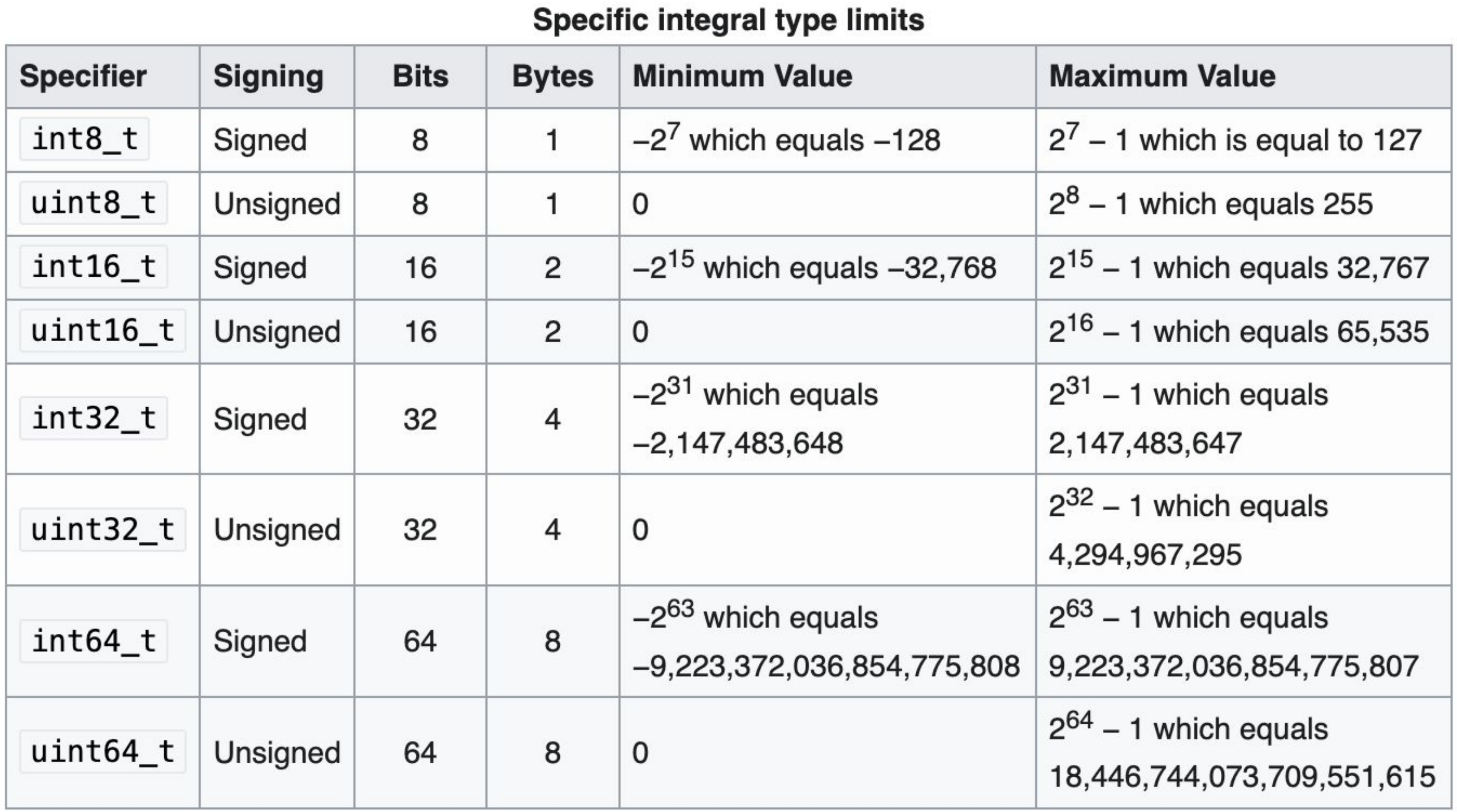
```
sizeof(char)

sizeof(double)

sizeof(float)

sizeof(int *)

sizeof(char *)
```

## sizeof()

```
sizeof(char)        // 1    (guaranteed)

sizeof(double)      // 8

sizeof(float)       // 4

sizeof(int *)       // 32bit: 4,  64bit: 8

sizeof(char *)      // 32bit: 4,  64bit: 8
sizeof(<type>*)     // 32bit: 4,  64bit: 8
```

| | ARM 32-bit Linux g++ 4.9 | ARM 64-bit Linux g++ 7.5.0 | x86 32-bit Linux g++ 4.8 | x64 64-bit Linux g++ 5.4 | x86 32-bit Windows 7 VisualStudio 2013 | x64 64-bit Windows 7 VisualStudio 2013 | x64 64-bit MacOS 10.15 clang 11.0.0 |
|---|---|---|---|---|---|---|---|
| **sizeof(bool)** | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **sizeof(char)** | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **sizeof(short)** | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **sizeof(int)** | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| **sizeof(long)** | 4 | 8 | 4 | 8 | 4 | 4 | 8 |
| **sizeof(long long)** | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| **sizeof(float)** | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| **sizeof(double)** | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| **sizeof(long double)** | 8 | 16 | 12 | 16 | 8 | 8 | 16 |
| **sizeof(size_t)** | 4 | 8 | 4 | 8 | 4 | 8 | 8 |
| **sizeof(void *)** | 4 | 8 | 4 | 8 | 4 | 8 | 8 |
| **sizeof(DWORD)** | n/a | n/a | n/a | n/a | 4 | 4 | n/a |

## `#include <stdint.h>  (C99+)`



**Specific integral type limits**

| Specifier | Signing | Bits | Bytes | Minimum Value | Maximum Value |
|---|---|---|---|---|---|
| `int8_t` | Signed | 8 | 1 | $-2^{7}$ which equals −128 | $2^{7} - 1$ which is equal to 127 |
| `uint8_t` | Unsigned | 8 | 1 | 0 | $2^{8} - 1$ which equals 255 |
| `int16_t` | Signed | 16 | 2 | $-2^{15}$ which equals −32,768 | $2^{15} - 1$ which equals 32,767 |
| `uint16_t` | Unsigned | 16 | 2 | 0 | $2^{16} - 1$ which equals 65,535 |
| `int32_t` | Signed | 32 | 4 | $-2^{31}$ which equals −2,147,483,648 | $2^{31} - 1$ which equals 2,147,483,647 |
| `uint32_t` | Unsigned | 32 | 4 | 0 | $2^{32} - 1$ which equals 4,294,967,295 |
| `int64_t` | Signed | 64 | 8 | $-2^{63}$ which equals −9,223,372,036,854,775,808 | $2^{63} - 1$ which equals 9,223,372,036,854,775,807 |
| `uint64_t` | Unsigned | 64 | 8 | 0 | $2^{64} - 1$ which equals 18,446,744,073,709,551,615 |

## sizeof()

```
sizeof(int)     // == 4 (bytes -- 32 bits)

int A[] = {1, 2, 3, 4, 5};
sizeof(A)       // == ?
```

## sizeof()

```
sizeof(int)     // == 4 (bytes -- 32 bits)

int A[] = {1, 2, 3, 4, 5};
sizeof(A)       // ==  20

sizeof(A) / sizeof(int)
                // == ?
```

## sizeof()

```
sizeof(int)    // == 4 (bytes -- 32 bits)

int A[] = {1, 2, 3, 4, 5};
sizeof(A)      // ==  20

sizeof(A) / sizeof(int)
               // == 5
```

Warning: this does **not** work for arrays passed to functions.  It only works where the array was defined.

## sizeof() in functions

```
int A[] = {1, 2, 3, 4};
printf(“%d\n”, sizeof(A)); // ?
double_nums(A, 4);
…

void double_nums(int A[], int n) {
   …
   printf(“%d\n”, sizeof(A)); // ?
}
```

## sizeof() in functions

```
int A[] = {1, 2, 3, 4};
printf(“%d\n”, sizeof(A)); // ?
double_nums(A, 4);
…

void double_nums(int A[], int n) {
   …
   printf(“%d\n”, sizeof(A)); // (program.c:12)
}
```

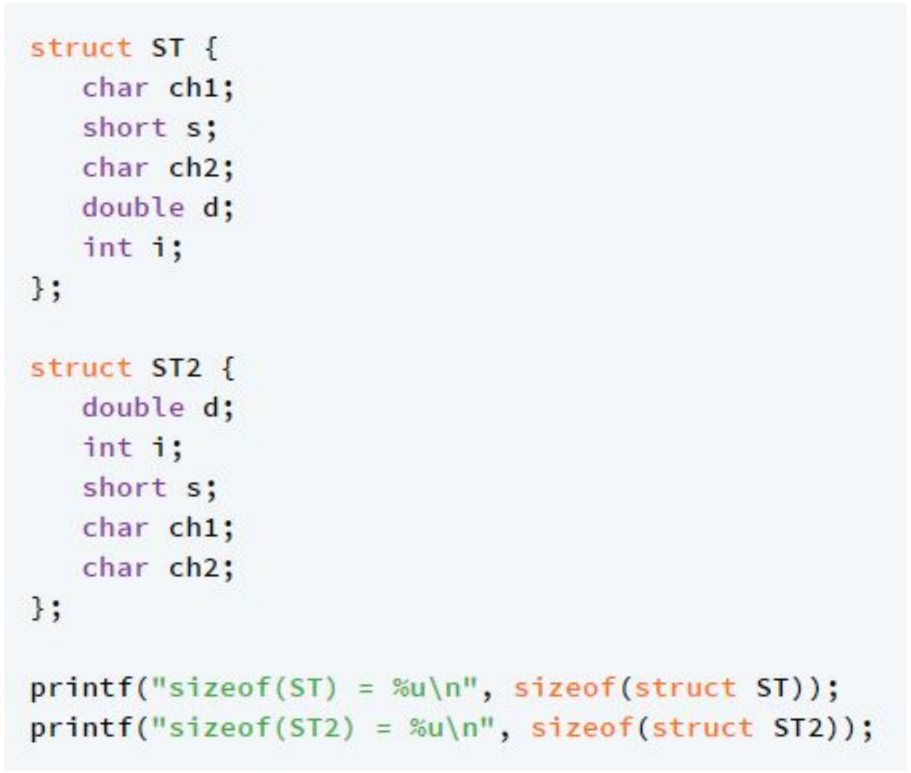
```
program.c:12:23: error: sizeof on array function parameter will return
size of 'int *' instead of 'int []' [-Werror,-Wsizeof-array-argument]
```

## sizeof() in functions

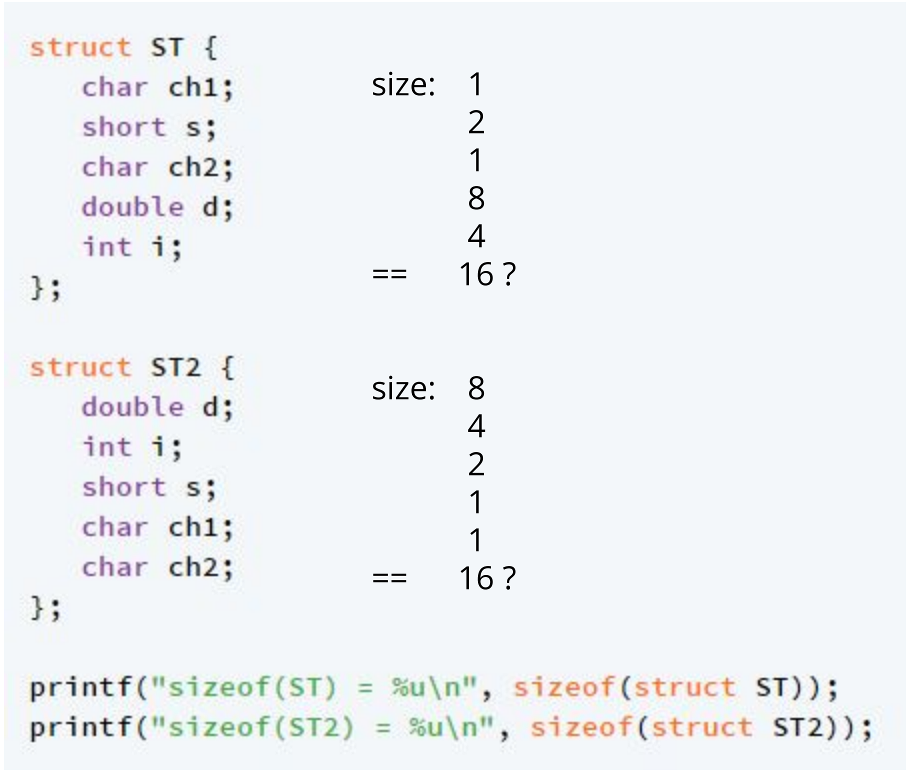
```
int A[] = {1, 2, 3, 4};
printf(“%d\n”, sizeof(A)); // 16
double_nums(A, 4);
…

void double_nums(int A[], int n) {
   …
   printf(“%d\n”, sizeof((int *) A)); // 8
}
```

Can’t get the sizeof arrays passed to functions!

## sizeof structs (W8)



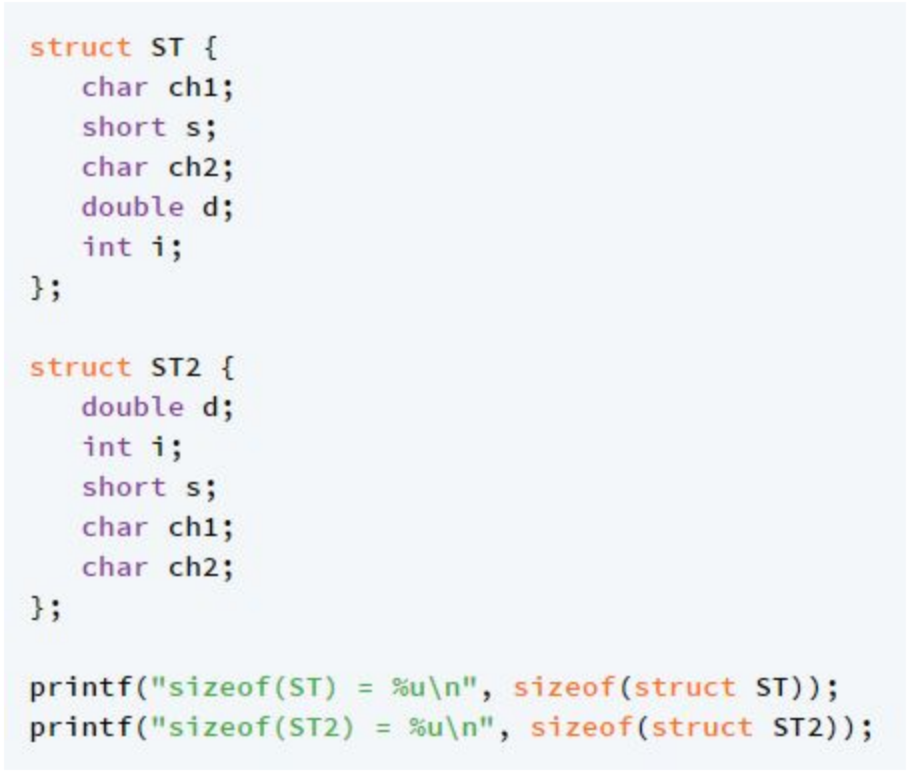
```c
struct ST {
   char ch1;
   short s;
   char ch2;
   double d;
   int i;
};

struct ST2 {
   double d;
   int i;
   short s;
   char ch1;
   char ch2;
};

printf("sizeof(ST) = %u\n", sizeof(struct ST));
printf("sizeof(ST2) = %u\n", sizeof(struct ST2));
```



```c
struct ST {
   char ch1;
   short s;
   char ch2;
   double d;
   int i;
};

struct ST2 {
   double d;
   int i;
   short s;
   char ch1;
   char ch2;
};

printf("sizeof(ST) = %u\n", sizeof(struct ST));
printf("sizeof(ST2) = %u\n", sizeof(struct ST2));
```

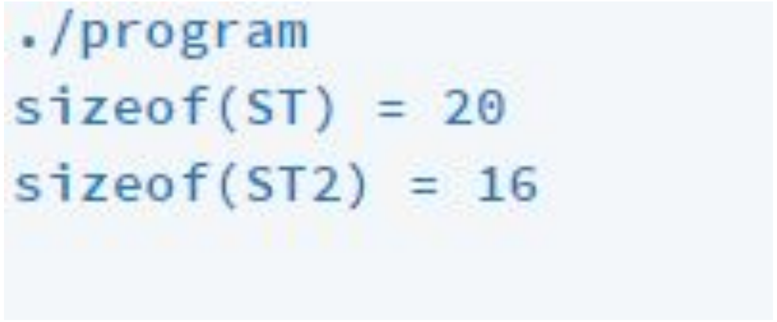
```
size:   1
        2
        1
        8
        4
==     16 ?
```

```
size:   8
        4
        2
        1
        1
==     16 ?
```



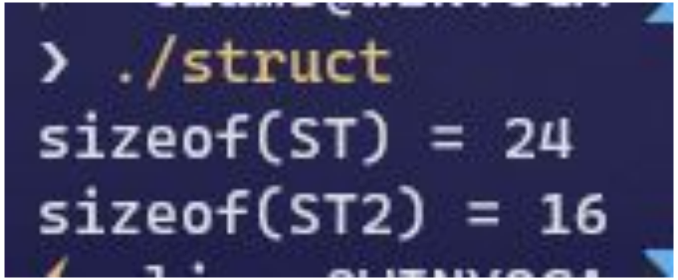
```c
struct ST {
   char ch1;
   short s;
   char ch2;
   double d;
   int i;
};

struct ST2 {
   double d;
   int i;
   short s;
   char ch1;
   char ch2;
};

printf("sizeof(ST) = %u\n", sizeof(struct ST));
printf("sizeof(ST2) = %u\n", sizeof(struct ST2));
```

32-bit



```
./program
sizeof(ST) = 20
sizeof(ST2) = 16
```

64-bit



```
❯ ./struct
sizeof(ST) = 24
sizeof(ST2) = 16
```

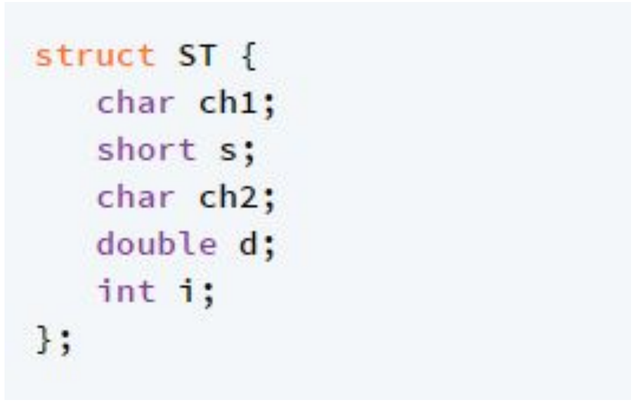
```c
struct ST {
   char ch1;
   short s;
   char ch2;
   double d;
   int i;
};
```

```
./program
sizeof(ST) = 20
sizeof(ST2) = 16
```

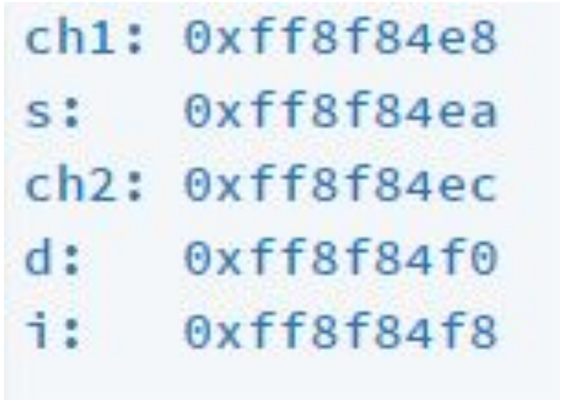
```c
struct ST st = {1,1,1,1,1};

printf("ch1: %p\n", &st.ch1);
printf("s:   %p\n", &st.s);
printf("ch2: %p\n", &st.ch2);
printf("d:   %p\n", &st.d);
printf("i:   %p\n", &st.i);
```

```
ch1: 0xff8f84e8
s:   0xff8f84ea
ch2: 0xff8f84ec
d:   0xff8f84f0
i:   0xff8f84f8
```

```
size:  2   ...?
       2
       4   ...?
       8
       ??
==    16 + ??   (so ?? == 4)
```

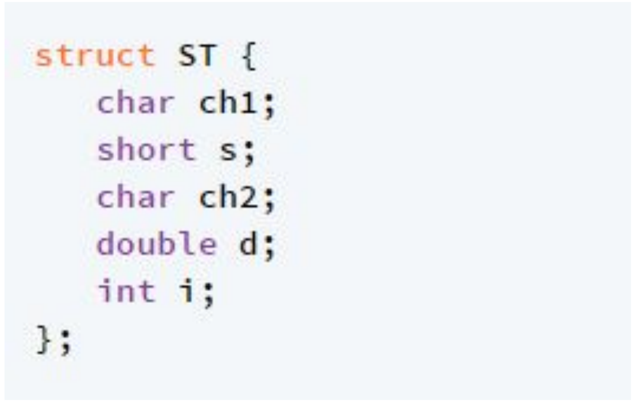


```c
struct ST {
   char ch1;
   short s;
   char ch2;
   double d;
   int i;
};
```

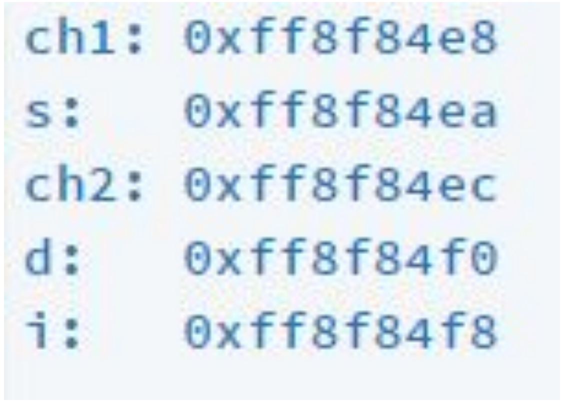


```
ch1: 0xff8f84e8
s:   0xff8f84ea
ch2: 0xff8f84ec
d:   0xff8f84f0
i:   0xff8f84f8
```

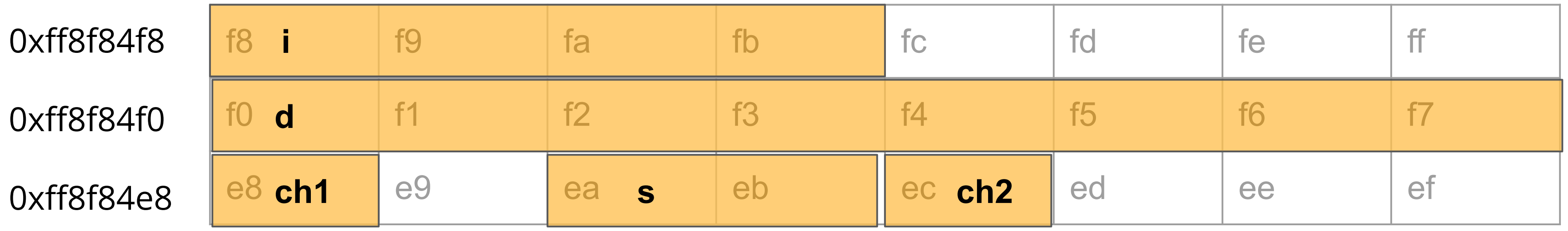
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0xff8f84f8 | f8 | f9 | fa | fb | fc | fd | fe | ff |
| 0xff8f84f0 | f0 | f1 | f2 | f3 | f4 | f5 | f6 | f7 |
| 0xff8f84e8 | e8 | e9 | ea | eb | ec | ed | ee | ef |



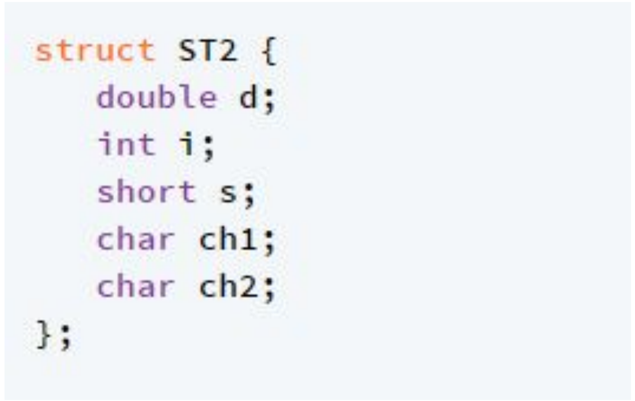
```
struct ST {
    char ch1;
    short s;
    char ch2;
    double d;
    int i;
};
```



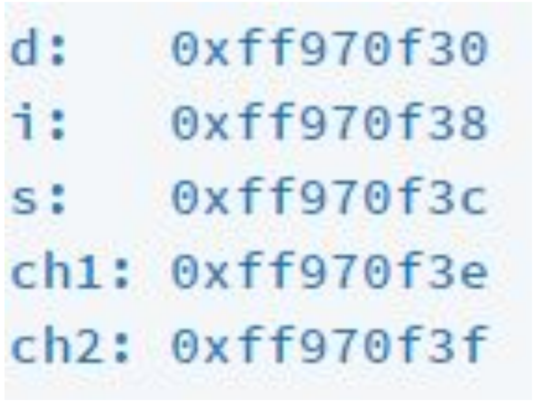
```
ch1: 0xff8f84e8
s:   0xff8f84ea
ch2: 0xff8f84ec
d:   0xff8f84f0
i:   0xff8f84f8
```



Why…? Preserve ABI
Application Binary Interface



```c
struct ST2 {
   double d;
   int i;
   short s;
   char ch1;
   char ch2;
};
```

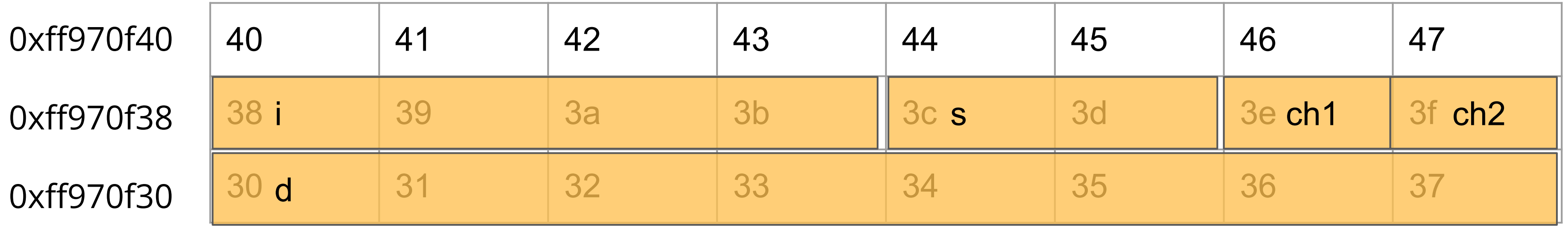


```
d:   0xff970f30
i:   0xff970f38
s:   0xff970f3c
ch1: 0xff970f3e
ch2: 0xff970f3f
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0xff970f40 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 0xff970f38 | 38 | 39 | 3a | 3b | 3c | 3d | 3e | 3f |
| 0xff970f30 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |

```c
struct ST2 {
    double d;
    int i;
    short s;
    char ch1;
    char ch2;
};
```

```
d:   0xff970f30
i:   0xff970f38
s:   0xff970f3c
ch1: 0xff970f3e
ch2: 0xff970f3f
```



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0xff970f40 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 0xff970f38 | 38 i | 39 | 3a | 3b | 3c s | 3d | 3e ch1 | 3f ch2 |
| 0xff970f30 | 30 d | 31 | 32 | 33 | 34 | 35 | 36 | 37 |