

Lecture 7: Pointers



CSE 29: Systems Programming and Software Tools
Aaron Schulman (Shalev)

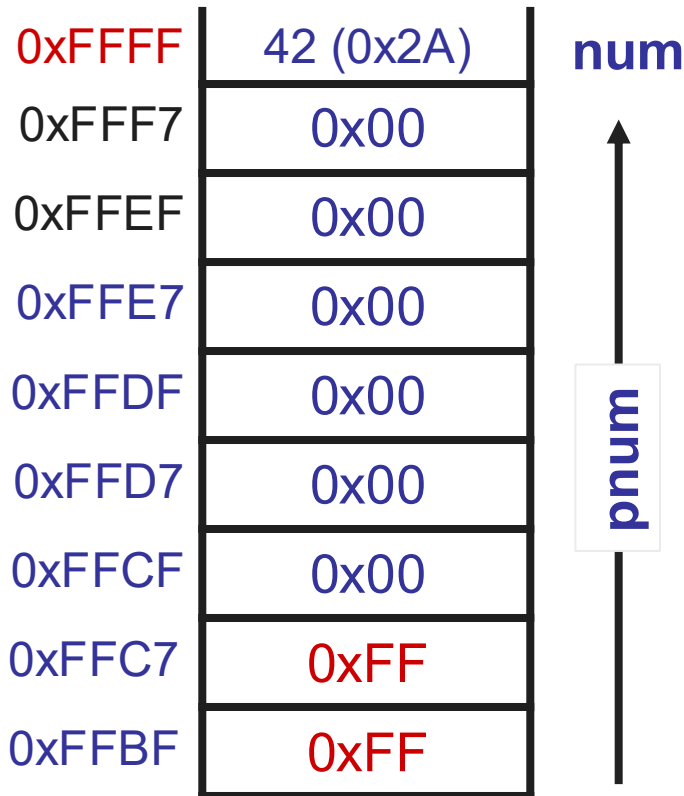


Introducing pointers in C

```
char num[] = { 42 }  
char* pnum = num;  
assert(pnum[0] == 42);
```

All variables reside in memory at a specific address.

Pointer: A variable in C that can store the address of another variable (or function)



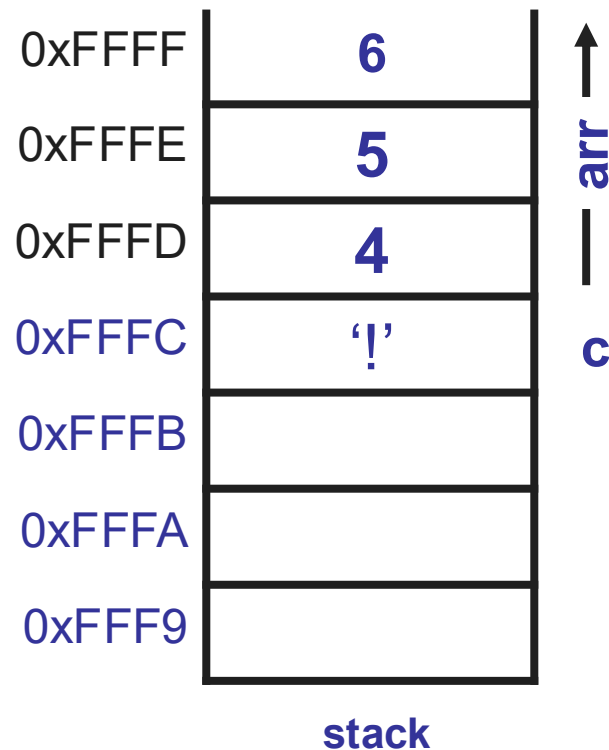
Layout of arrays and other variables in memory



```
char arr[] = {4,5,6};  
char c = '!';
```

Variable: A region of memory allocated to a **single value** of a specific data type

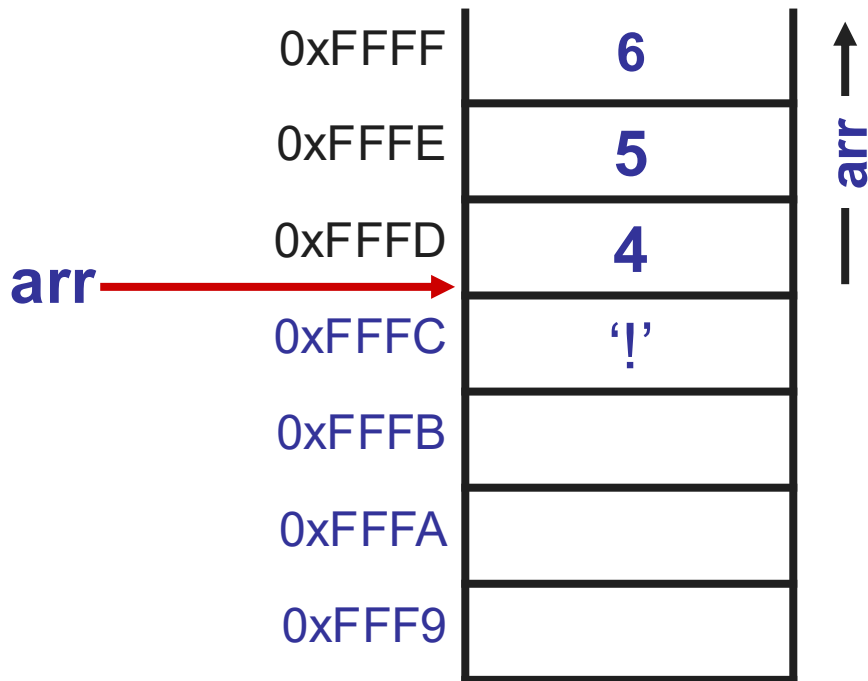
Array: A region of memory allocated to a **set of values** of a specific data type



Array variables are also pointers

```
char arr[3] = {4,5,6};  
printf("arr=%p\n", arr);
```

arr=0xFFFD



An array variable is always a pointer

- An array is a *contiguous region* of memory
 - The array variable **points to the start of this region**
- We add to the pointer to do indexing in the array
 - Array variable points to the **start**, to index into the array we **add** to this start address
- When you do an array index (e.g., `arr[3]`) the compiler will do pointer math
 - `arr[3]` will turn into `arr + (3 * sizeof(long int))`
- Pointers let us directly read and modify arrays “in place”



Example: string.h

- ❑ `int strlen(char *s)`
 - ❑ In-class notation would be `int strlen(char s[])`
- ❑ `void strcpy(char *dest, char *src)`
- ❑ `void strcat(char *dest, char* src)`
 - ❑ Copies from `src` to `dest`
 - ❑ Appends `src` to end of `dest`
- ❑ `char str[]` and `char *str` mean the same thing in function arguments
- ❑ `char*`: an address in memory where we can access `char` data
- ❑ `T*`: an address in memory where we can access `T` data
- ❑ `T` could be `int`, `double`, `int16_t`, `char*` etc



Special C syntax for pointers

& - Address of variable (get addr of var)

```
char* pnum = &num;
```

***** - *dereference* address (get var at addr)

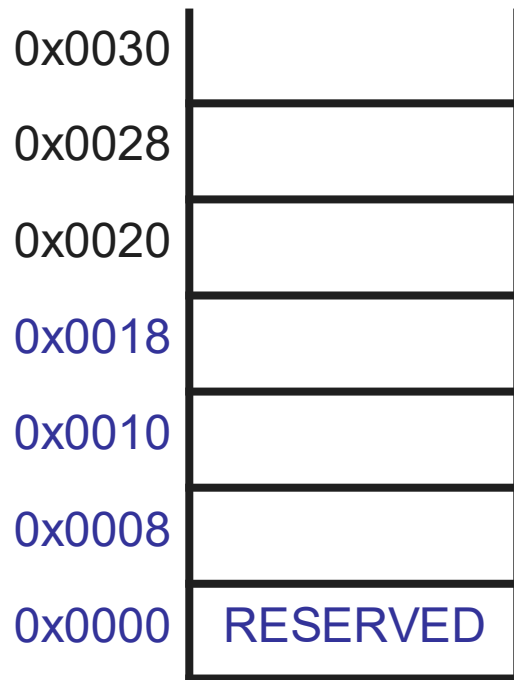
```
char num2 = *pnum;
```

How do you point to “nothing”

```
int *p = NULL;
```

```
// C always has this  
#define NULL 0
```

```
*p = 5;  
SEGMENTATION FAULT!!!!
```



stack
(strachpad memory) 8



Review Questions

1. What type of variable do functions that operate on strings take?
2. How many bytes of memory do a `uint64_t` and a `uint32_t` use?
3. How many bytes of memory does a `char*` pointer use?