Lecture 6: More pointer practice

CSE 29: Systems Programming and Software Tools

Olivia Weng

Announcements

- Problem set 1 is due tomorrow at 10am PT

- Sign up for Exam 1 on prairietest.com
  - Sign in with your UCSD credentials
  - Time slots available for Thursday, Monday, and Tuesday

What will be printed?

- Hint: draw the stack!

```
char b[] = {'C', 'S', 'E', '\0'};
char *ptr_b = b;
b[2] = 'I';
ptr_b[1] = 'H';
char c = *b;

printf("Values: %c\n", ptr_b[1]); → H
printf("Values: %c\n", b[1]); → H
printf("ptr_b = %s\n", ptr_b); → CHI
printf("b = %s\n", b); → CHI
printf("c = %c\n", c); → C
```

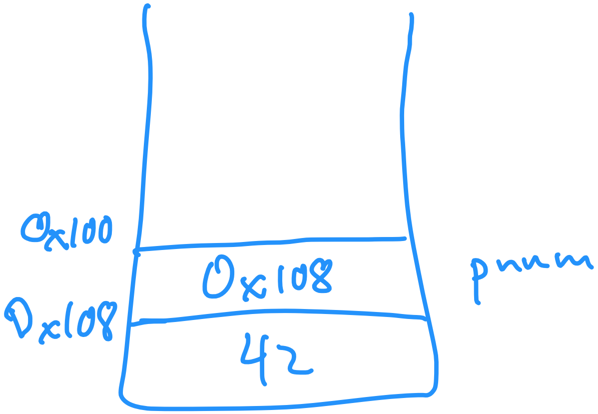
Pointers

word size

- Pointers are 8 bytes
  - Why?

Pointers

- A pointer can be an address to any type
  - Type determines size of the value stored at the address
    - Ex: `int *`, `char *'`



- To get the address of a variable, use &

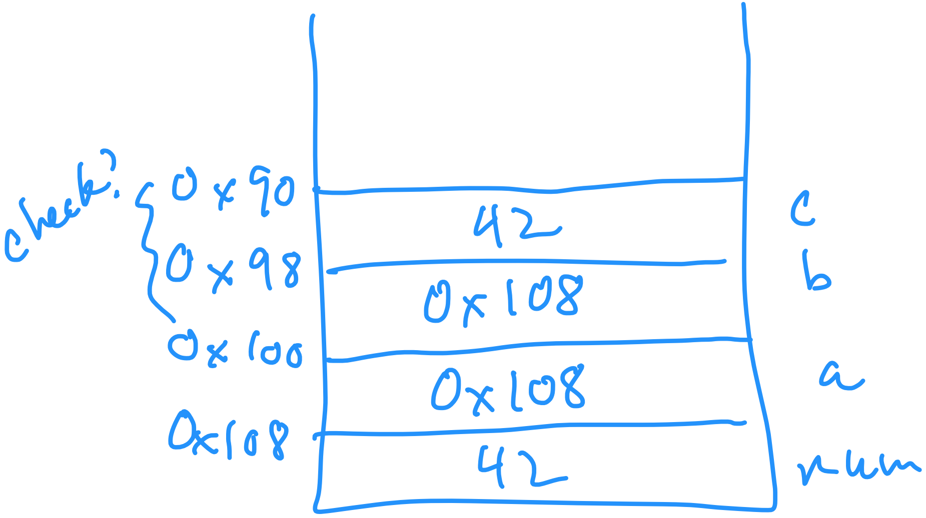
```
int num = 42;

int *pnum = &num;

printf("%p\n", &num);

printf("%p\n", pnum);
```

Address or value?

```
int num = 42;
int *a = &num;
int *b = a;
int c = *a;
```

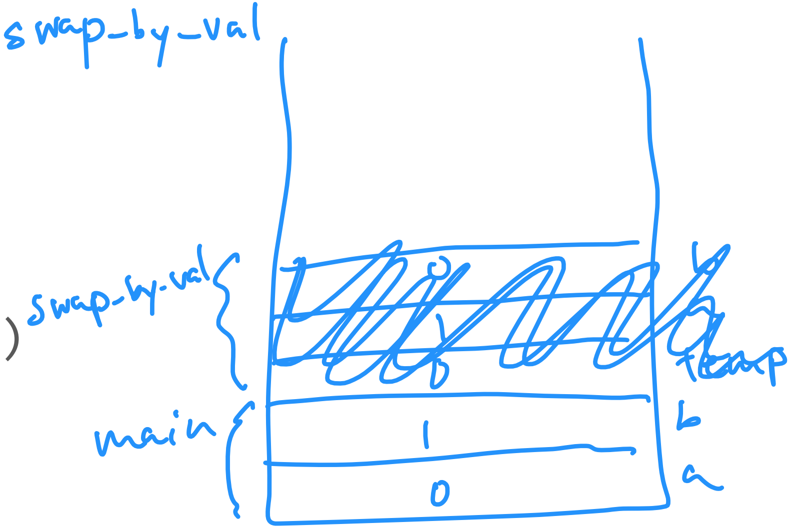
- Is this an address or value?
  - a → addr
  - *a → value
  - c → value
  - b → address
  - *b → value
  - &c → addr

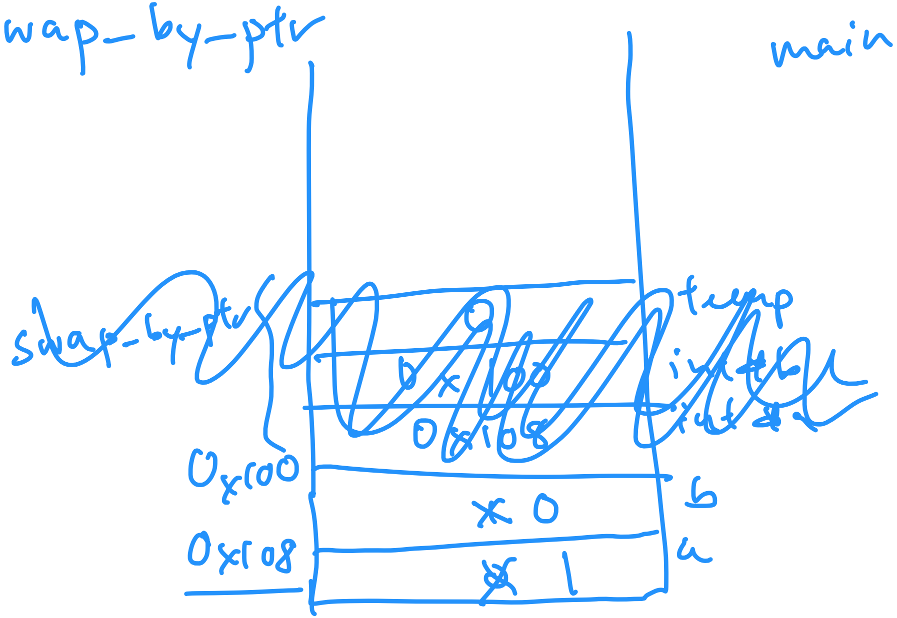
Bonus question: Does &c == &num?



Demo

- `void swap_by_val(int a, int b)`
- `void swap_by_ptr(int *a, int *b)`





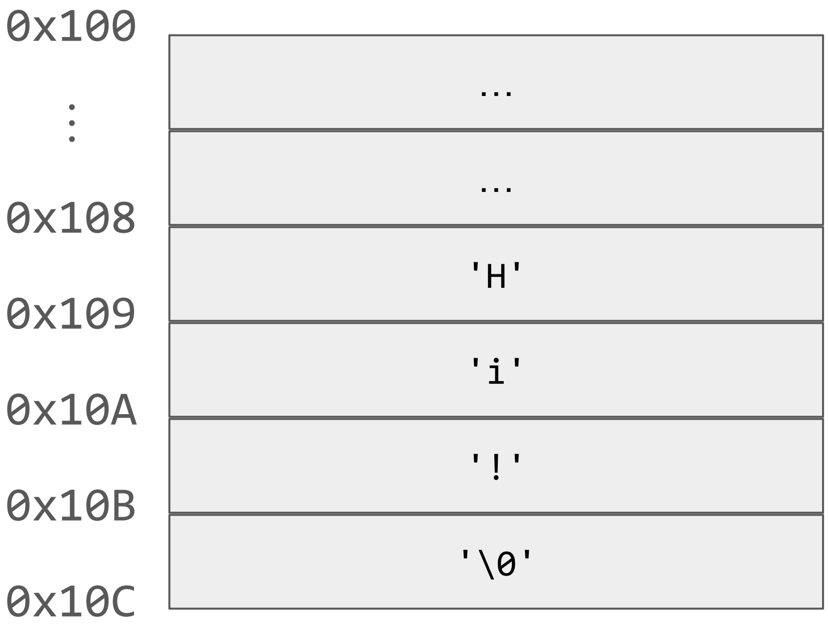
A pointer can also point to an array?

- An array is a region of memory allocated to a set of values of a specific data type
    - The name of an array corresponds to the address of the first element of an array

```
char arr[4] = {'H', 'i', '!', '\0'};
char *parr = arr;

printf("Values: %c %c\n", arr[1], parr[1]);

printf("Addresses: %p %p\n", arr, parr);
assert(arr == parr);
```

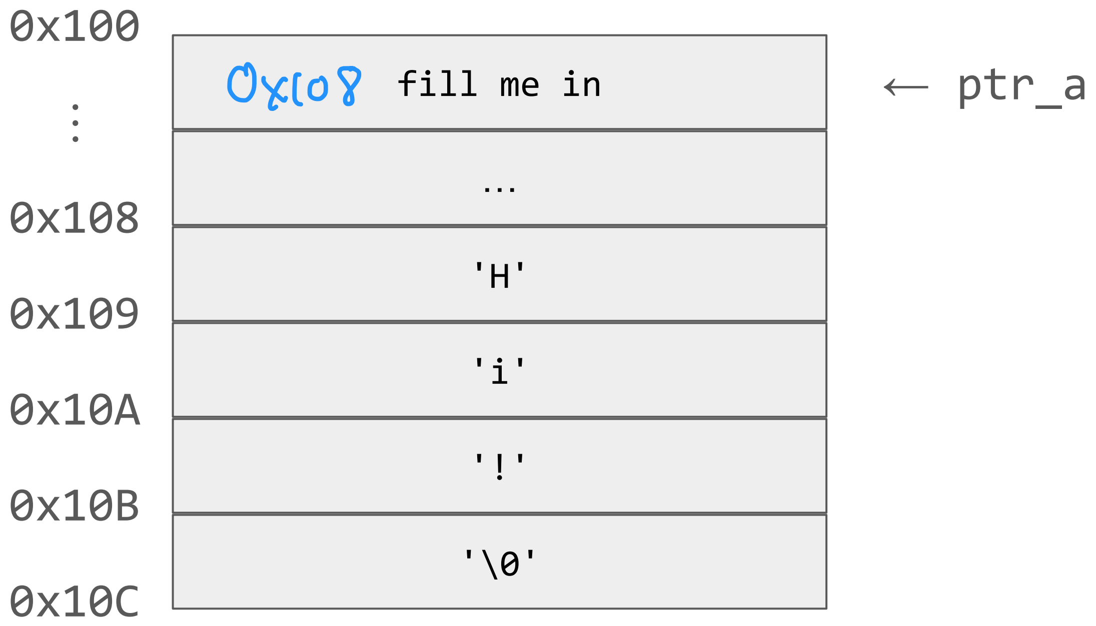
Where is arr pointing to? → 0x108

```
char arr[4] = {'H', 'i', '!', '\0'};
char *ptr_a = arr;
```



What value is ptr_a?

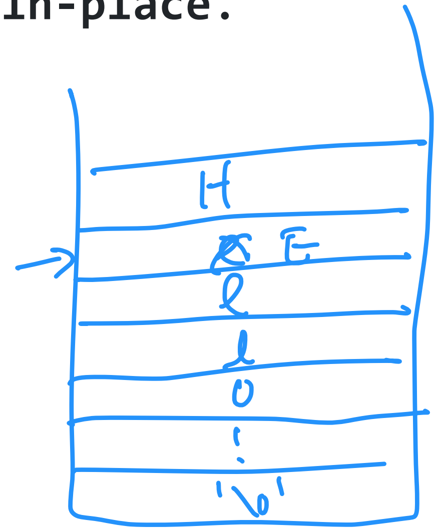
```
char arr[4] = {'H', 'i', '!', '\0'};
char *ptr_a = arr;
```



Pointers allow us to modify an array in-place

```
// Returns the number of characters capitalized and capitalizes
// the lowercase a-z ASCII characters of str in-place.
int32_t capitalize_ascii(char str[]);

int main() {
    char str[] = "Hello!";
    int num_cap = capitalize_ascii(str);
    printf("Capitalized str: %s\n", str);
}
```

str[1] = E



Pointers allow us to write results to a new place

```
// Encode a uint32_t number as UTF-8 (assuming 2-byte encoding)
void encode2(uint32_t num, char result[]);


int main() {
    uint32_t num = 128;
    char result[2];
    encode2(128, result);
    printf("num = %x\n", num);
    for (int i = 0; i < 2; i++) {
        printf("%02X ", (unsigned char) result[i]);
    }
}
```