Lecture 5: `main()` character energy

CSE 29: Systems Programming and Software Tools

Olivia Weng

Announcements

- Sent out email about 1-on-1 check ins

- Problem set 1 is due this Wednesday at 10am PT

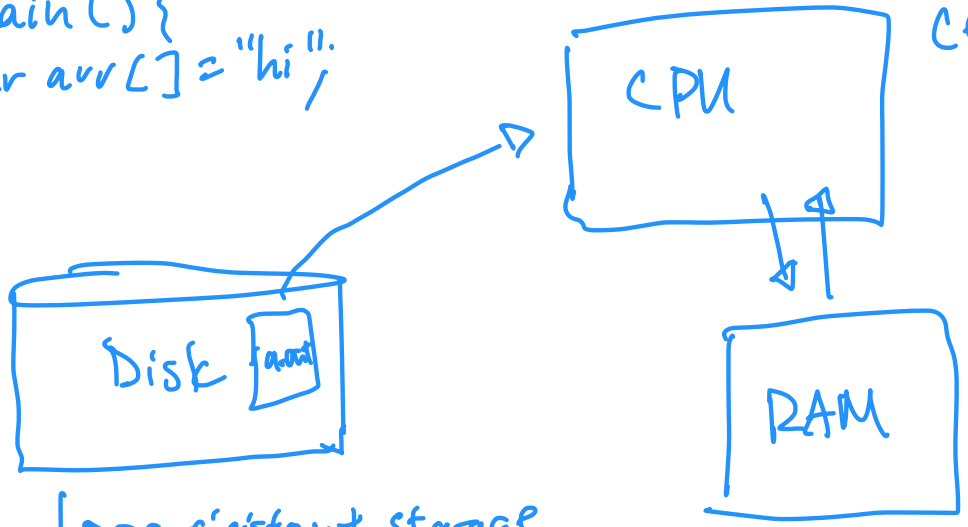
What happens when you run a program?

- Hardware is involved?
  - CPU? RAM? Disk??

write C → gcc → a.out

> ./a.out

```
int main() {
    char arr[] = "hi";

}
```



central processing unit
"brain"

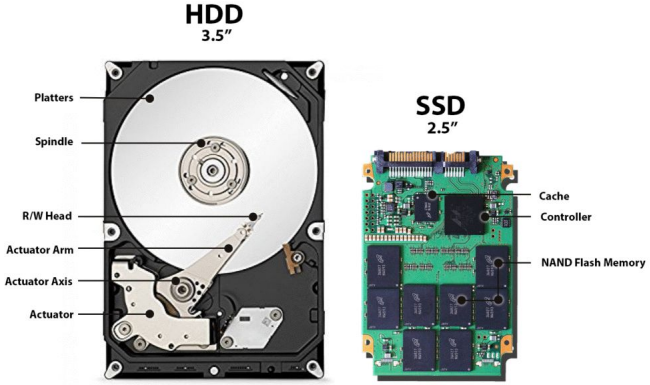
random access memory
memory
volatile

↳ persisistent storage
non-volatile

What happens when you run a program?

- Hardware is involved?
  - CPU? RAM? Disk??

Disk



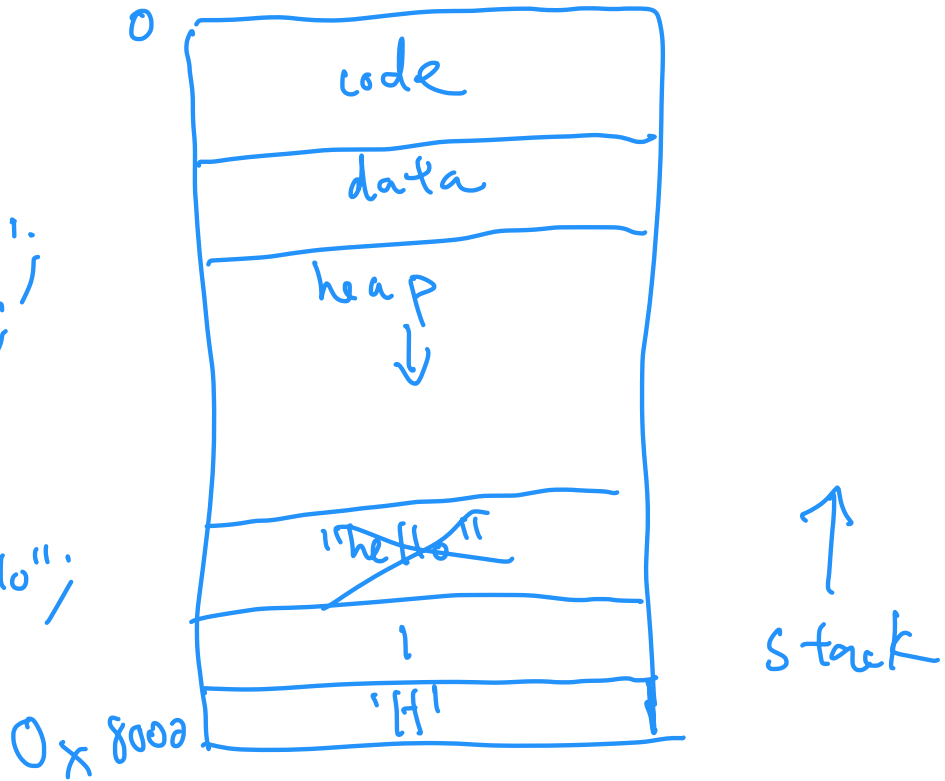
RAM

CPU

What's in a program's memory?

- Address space

```
int main() {
  char a = 'H';
  int b = 1;
  hello();
}

void hello() {
  char hello[] = "hello";
}
```



What's a C `main()` function?

```
int main(int argc, char *argv[]);
```

CLI arguments

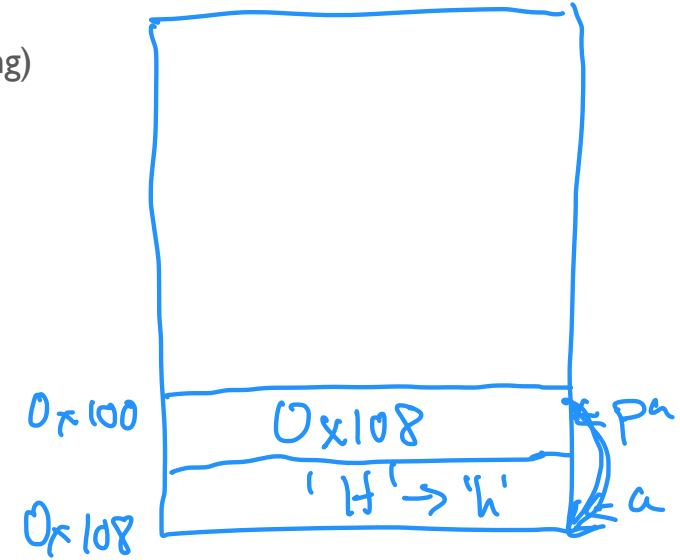
char *c = {'H'};

char to_lower(char *v)

What is char * data type?

- char * is a pointer (aka address) to memory storing another value of type char
  - a pointer is just a number, i.e., the address
  - pointer == address == reference (all mean the same thing)

```
char a[] = {'H'};
char *pa = a;

printf("Values: %c %c\n", a[0], pa[0]);
assert(pa[0] == 'H');

printf("Addresses: %p %p\n", a, pa);
assert(a == pa);
```



What will be printed?

- Hint: draw the stack!

```
char b[] = {'C', 'S', 'E'};
char *ptr_b = b;

printf("Values: %c %c\n", b[0], ptr_b[0]);

printf("Addresses: %p %p\n", b, ptr_b);
assert(b == ptr_b); // fail or not?
```

What will be printed?

- Hint: draw the stack!

```
char c = 'a';
char b[] = {'C', 'S', 'E'};
char *ptr_b = b;
b[2] = 'I';

printf("Values: %c\n", ptr_b[0]);
printf("Values: %c\n", ptr_b[2]);
```

'C'

'I'

What will be printed?

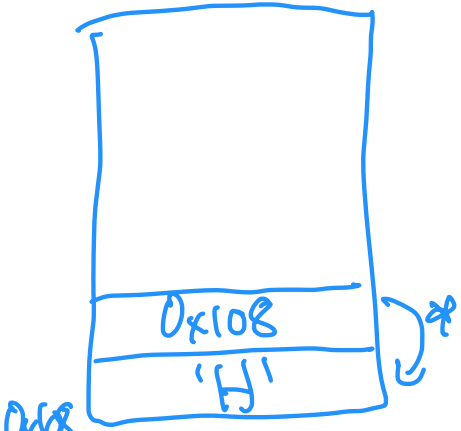
- Hint: draw the stack!

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

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

Dereferencing pointers

- To get the value stored at a pointer's address/reference, we dereference it
  - * is the dereference operator
  - What happens when we dereference?

```
char a[] = {'H'};
char *pa = a;

printf("Indexing values: %c %c\n", a[0], pa[0]);
printf("Dereferencing values: %c %c\n", *a, *pa);
```



Address or value?

```
char a[] = {'H'};
char *pa = a;
char b = *pa;
```

- Is this an address or value?
  - `a` → addr
  - `a[0]` → value
  - `*pa` → value
  - `pa` → addr
  - `b` → value

What will be printed?

- Hint: draw the stack!

```
char a[] = {'H'};
char *pa = a;
char pb = *pa;

printf("%c\n", pb);
```

What will be printed?

- Hint: draw the stack!

```
char a[] = {'H'};
char *pa = a;
char *pb = pa;

printf("values: %c %c %c\n", a[0], pa[0], pb[0]);
printf("values: %c %c\n", *pa, *pb);
```

What will be printed?

- Hint: draw the stack!

```
char a[] = {'H'};
char *pa = a;
??? // fill me in
```

char b = a[0];

char b = *pa;

b

```
printf("Same values: %c %c %c\n", a[0], pa[0], ??);
```