Lecture 6: `main()` character energy

CMPE 120: Computer Organization & Architecture

Olivia Weng

Logistics

- HW1 released last Thursday

- HW0 grades released
  - If your signature was missing, please send me a message on Piazza

- Olivia has extended office hours this week
  - Tuesday: 3:30 - 5pm
  - Friday: 9 - 11:30am

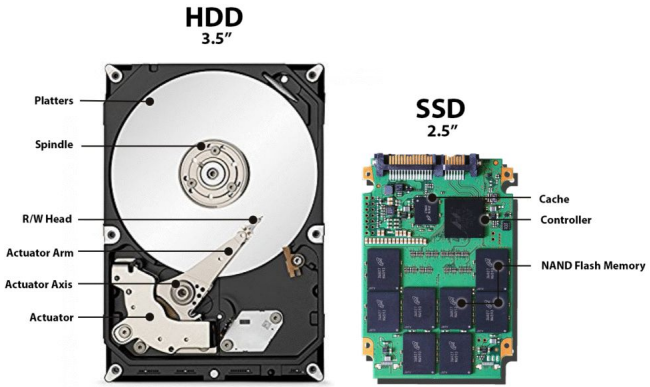
What happens when you run a program?

- Hardware is involved?

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

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

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

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', 'M', 'P', '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!

A. P, E
B. C, M
C. C, A
D. P, a

```
char c = 'a';
char b[] = {'C', 'M', 'P', 'E'};
char *ptr_b = b;
b[1] = 'A'; // syntactic sugar

printf("Values: %c\n", ptr_b[0]);
printf("Values: %c\n", ptr_b[1]);
```

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);
```

A. all H
B. all h

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);
```

Pointers

- 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 (a) or value (v)?
  - `a`
  - `*a`
  - `c`
  - `b`
  - `*b`
  - `&c`

Bonus question: Does &c == &num?

A.	v, a, v, v, a, v
B.	v, a, a, a, v, a
C.	a, v, v, a, v, a
D.	a, v, a, a, v, a