

1. What does SHA256 do? (The function)
2. The hash of my password starts 04A1ZBC9. (nothing)
What do you know about it?
3. Write a main function that prints the first command-line argument.

1 SHA256 takes:

- an input char array (any size)
- a length
- an output array (32 bytes available)

and changes the output array to have the SHA256 hash of the first length bytes of input

args.c

```
3. int main( int argc , char** args ) {
    char
    char* first_arg = args[1];
    printf( "%s" , first_arg );
}
```

```
$ gcc args.c -o args
```

```
$ ./args banana apple
```

```
banana $
```

```
0x...10
```

```
0x...20
```

```
args
```

```
0xA0
```

```
"./args"
```

```
"banana"
```

```
0x...A0
```

```
0x...10
```

```
0x...20
```

```
0x...30
```

args[0] => "./args"

args[1] => "banana"

⋮

```

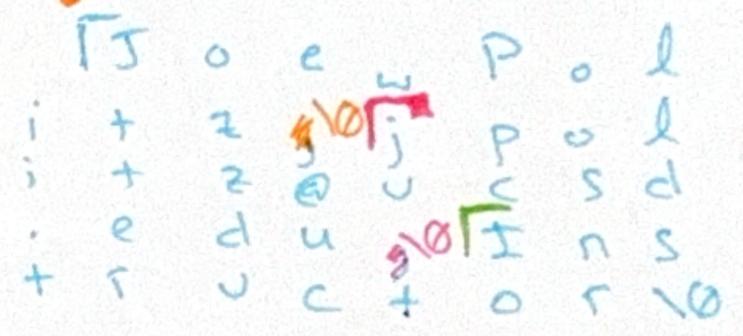
1 #include <stdio.h>
2 #include <string.h>
3
4 // char *strtok(char *str, const char *delim);
5 // Returns a pointer to the next token in str, delimited by delim.
6 // First call: pass the string. Subsequent calls: pass NULL.
7 // Replaces delim with '\0' returns pointer into original string.
8
9 int main() {
10 char str[] = "Joe Politz,jpolitz@ucsd.edu,Instructor";
11 printf("str%p: %p \"%s\"\n", &str, str, str);
12
13 char* a = strtok(str, ",");
14 char* b = strtok(NULL, ",");
15 char* c = strtok(NULL, ",");
16 char* d = strtok(NULL, ",");
17
18 printf("a%p: %p \"%s\"\n", &a, a, a);
19 printf("b%p: %p \"%s\"\n", &b, b, b);
20 printf("c%p: %p \"%s\"\n", &c, c, c);
21 printf("d%p: %p\n", &d, d);
22
23 printf("\nstr after strtok:\n");
24 for (int i = 0; i < 39; i++) {
25     if (i % 8 == 0) printf("%p:", &str[i]);
26     if (str[i]) printf(" '%c'", str[i]);
27     else printf(" NUL");
28     if (i % 8 == 7) printf("\n");
29 }
30 printf("\n");
31 }

```

~ strtok remembered 0x...5B d
 ~ strtok remembered 0x...6C c
 ~ strtok remembered 0x...77 b
 a

Variable/Role	Address	Data
	0x...00	
	0x...08	
	0x...10	
	0x...18	
	0x...20	
	0x...28	
	0x...30	
	0x...38	
	0x...40	
	0x...48	
str	0x...50	
	0x...58	
	0x...60	
	0x...68	
	0x...70	
	0x...78	
	0x...80	
	0x...88	
	0x...90	
	0x...98	
	0x...A0	
	0x...A8	
	0x...B0	
	0x...B8	
	0x...C0	
	0x...C8	
	0x...D0	
	0x...D8	
	0x...E0	
	0x...E8	
	0x...F0	
	0x...F8	

0x0 (NULL) — interior pointer
 0x16b7b6e6d — interior pointer
 0x16b7b6e5c — interior pointer
 0x16b7b6e51



```

$ gcc strtok.c -o strtok
$ ./strtok
str@0x16b7b6e51: 0x16b7b6e51 "Joe Politz,jpolitz@ucsd.edu,Instructor"
a@0x16b7b6e40: 0x16b7b6e51 "Joe Politz"
b@0x16b7b6e38: 0x16b7b6e5c "jpolitz@ucsd.edu"
c@0x16b7b6e30: 0x16b7b6e6d "Instructor"
d@0x16b7b6e28: 0x0

str after strtok:
0x16b7b6e51: 'J' 'o' 'e' ' ' 'P' 'o' 'l' 'i'
0x16b7b6e59: 't' 'z' NUL 'j' 'p' 'o' 'l' 'i'
0x16b7b6e61: 't' 'z' '@' 'u' 'c' 's' 'd' ' '
0x16b7b6e69: 'e' 'd' 'u' NUL 'I' 'n' 's' 't'
0x16b7b6e71: 'r' 'u' 'c' 't' 'o' 'r' NUL

```

How to implement a shell. You've been using "bash"
on mac - "zsh"
Joe likes - "fish"
windows - "PowerShell"

1. Prints a prompt
2. "Parses" a command + args the user typed
3. Runs that command (by asking the OS)

plus bonus:

- tab complete
- saves history
(up arrow, Ctrl-R)
- I/O redirection
\$./prog <input >output