```c
#include <stdio.h>
#include <string.h>

// char *strtok(char *str, const char *delim);
// Returns a pointer to the next token in str, delimited by delim.
// First call: pass the string. Subsequent calls: pass NULL.
// Replaces delim with '\0'returns pointer into original string.

int main() {
  char str[] = "Joe Politz,jpolitz@ucsd.edu,Instructor";
  printf("str@%p: %p \"%s\"\n", &str, str, str);

  char* a = strtok(str, ",");
  char* b = strtok(NULL, ",");
  char* c = strtok(NULL, ",");
  char* d = strtok(NULL, ",");

  printf("a@%p: %p \"%s\"\n", &a, a, a);
  printf("b@%p: %p \"%s\"\n", &b, b, b);
  printf("c@%p: %p \"%s\"\n", &c, c, c);
  printf("d@%p: %p\n", &d, d);

  printf("\nstr after strtok:\n");
  for (int i = 0; i < 39; i++) {
    if (i % 8 == 0) printf("%p:", &str[i]);
    if (str[i]) printf(" '%c'", str[i]);
    else printf(" NUL");
    if (i % 8 == 7) printf("\n");
  }
  printf("\n");
}
```

remembers stopped at 0x...5C; d
remembers stopped at 0x...6D; c
remembered all done! b
a
str

```
$ 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
```

Variable/Role	Address	Data
	0x...00	
	0x...08	
	0x...10	
	0x...18	
	0x...20	
	0x...28	0x0 (NULL)
	0x...30	0x16b7b6e6d — interior pointer
	0x...38	0x16b7b6e5c — interior pointer
	0x...40	0x16b7b6e51
	0x...48	
	0x...50	J o e ␣ P o l
	0x...58	i t z \0 j p o l
	0x...60	i t z @ u c s d
	0x...68	. e d u \0 I n s
	0x...70	t r u c t o r \0
	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	

Shell — how do they work?

"bash"	Bourne-again shell
"zsh"	Mac default
"fish"	Joe likes it
"PowerShell"	Windows default

---

1. Print prompt — jpolitz[@ieng6~201] $
2. Reads + "parses" a command from user
3. Asks the OS to run it

Bonus features:

- History of commands (up-arrow, Ctrl-R)
- control over working dir
- hand control over to vim/etc.
- tab for autocomplete
- I/O redirection

  ./prog < input > output

1. What does the SHA256 function do?

2. The start of the hash of my password is 0x01AB3F
What do you know about my password?

3. Write a main() function that prints the 1st command line argument

SHA256 takes:

- input char array (at least length bytes long, doesn't have to be C string)
- length
- output char array (at least 32 bytes long)

Calculates the hash of the first length bytes of input, stores the result in output (changes output)

argtest.c

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

```
$ gcc argtest.c -o argtest
$ ./argtest cse29 hello
cse29
$
```

```
0x..10 | . / a r g t e s t \0
0x..20 | c s e 2 9 \0
0x..30 | h e l l o \0

0xB0 | 0x...10
     | 0x....20
     | 0x....30

args 0xF0 | 0x...B0
```

args[0] = "./argtest"
args[1] = "cse29"