```c
#include <stdio.h>
#include <string.h>
 
 
 
 
int parse_args(char* input, char** args) {
  char* current = strtok(input, " ");
  int current_index = 0;
  while(current != NULL) {
    args[current_index] = current;
    current = strtok(NULL, " ");
    current_index += 1;
  }
  return current_index;
}

int main() {
  while(1) {
    printf(" ");
    char input[2048];
    fgets(input, 2048, stdin);
    printf("Now computer run this: %s\n", input);

    char* args[1000];
    int argc = parse_args(input, args);
    for(int i = 0; i < argc; i += 1) {
      printf("args[%d]: %s\n", i, args[i]);
    }
 
 
 
 
 
 
 
 
 
 
 
 
 
 
 
  }
}
```

input

⇒ ls ..

args[0] = ls

args[1] = ..

want to run args[0]

```c
#include <stdio.h>

int NUM = 1000;
char HI[] = "Hi!";

int static_counter_fun() {
  static int ctr = 0; // Only happens on the first call!
  ctr += 1;
  printf("ctr:\t%d\n", ctr);
  printf("&ctr:\t%p\n", &ctr);
  return ctr;
}

int main(int argc, char** argv) {
  char hello[] = "hello everyone";
  char* ptr = hello;// could also write &hello
  char* hiptr = HI; // could also write &HI
  int* numptr = &NUM; // could NOT also write NUM
  printf("&main:\t%p\n", &main);
  printf("&main:\t%p\n", &main);
  printf("&NUM:\t%p\n", &NUM);
  printf("numptr:\t%p\n", numptr);
  printf("&HI:\t%p\n", &HI);
  printf("hiptr:\t%p\n", hiptr);
  printf("stdin:\t%p\n", &stdin);
  static_counter_fun();
  static_counter_fun();
  printf("&argv:\t%p\n", &argv);
  printf("ptr:\t%p\n", ptr);
  printf("&ptr:\t%p\n", &ptr);
  printf("&hiptr:\t%p\n", &hiptr);
  printf("hello:\t%p\n", hello);
  printf("argv:\t%p\n", argv);
  printf("argv[0]:%p\n", argv[0]);
}
```

```
$ ./layout
&static_counter_fun:    0x6046a1dcf169
&main:  0x6046a1dcf1c2
&NUM:   0x6046a1dd2010
numptr: 0x6046a1dd2010
&HI:    0x6046a1dd2014
hiptr:  0x6046a1dd2014
stdin:  0x6046a1dd2020
ctr:    1
&ctr:   0x6046a1dd202c
ctr:    2
&ctr:   0x6046a1dd202c
&argv:  0x7ffe89c5a5b0
ptr:    0x7ffe89c5a5d9
&ptr:   0x7ffe89c5a5c0
&hiptr: 0x7ffe89c5a5c8
hello:  0x7ffe89c5a5d9
argv:   0x7ffe89c5a718
argv[0]:        0x7ffe89c5c73c
```

How many bytes between the low + high printed addresses?

1. 1000s
2. 1000000s
3. billions
4. more

```c
#include <stdio.h>

int NUM = 1000;
char HI[] = "Hi!";

int static_counter_fun() {
  static int ctr = 0; // Only happens on the first call!
  ctr += 1;
  printf("ctr:\t%d\n", ctr);
  printf("&ctr:\t%p\n", &ctr);
  return ctr;
}

int main(int argc, char** argv) {
  char hello[] = "hello everyone";
  char* ptr = hello;// could also write &hello
  char* hiptr = HI; // could also write &HI
  int* numptr = &NUM; // could NOT also write NUM
  printf("&main:\t%p\n", &main);
  printf("&NUM:\t%p\n", &NUM);
  printf("numptr:\t%p\n", numptr);
  printf("&HI:\t%p\n", &HI);
  printf("hiptr:\t%p\n", hiptr);
  printf("stdin:\t%p\n", &stdin);
  static_counter_fun();
  static_counter_fun();
  printf("&argv:\t%p\n", &argv);
  printf("ptr:\t%p\n", ptr);
  printf("&ptr:\t%p\n", &ptr);
  printf("&hiptr:\t%p\n", &hiptr);
  printf("hello:\t%p\n", hello);
  printf("argv:\t%p\n", argv);
  printf("argv[0]:%p\n", argv[0]);
}
```

printf("&static_counter_fun: %p\n", &static_counter_fun)

code + globals

```
$ ./layout
&static_counter_fun:    0x6046a1dcf169
&main:  0x6046a1dcf1c2
&NUM:   0x6046a1dd2010
numptr: 0x6046a1dd2010
&HI:    0x6046a1dd2014
hiptr:  0x6046a1dd2014
stdin:  0x6046a1dd2020
ctr:    1
&ctr:   0x6046a1dd202c
ctr:    2
&ctr:   0x6046a1dd202c
&argv:  0x7ffe89c5a5b0
ptr:    0x7ffe89c5a5d9
&ptr:   0x7ffe89c5a5c0
&hiptr: 0x7ffe89c5a5c8
hello:  0x7ffe89c5a5d9
argv:   0x7ffe89c5a718
argv[0]:        0x7ffe89c5c73c
```

static variable is like a global but can only "see" it from the function that declared it
→ this is what strtok does: static char* current;

main stack frame

Operating system

0x6046... → cf 169, cf 1c2

CODE/TEXT
→ static_counter_fun
→ main

GLOBAL/STATIC
← 1000 →
'H' 'i' '!' '\0'
0 1 2

NUM	d2010
HI	d2014
ctr	d202c
stdin	d2020

HEAP
↓
10 trillion bytes?

STACK

main stack frame

argv	a5b0	a718
ptr	a5c0	
hiptr	a5c8	
hello	a5d9	h e l l o w e v e r y
		o n e \0

	a718	c73c
	c73c	"./layout"

0x7ffe ... →

0xFFFFFFFF...

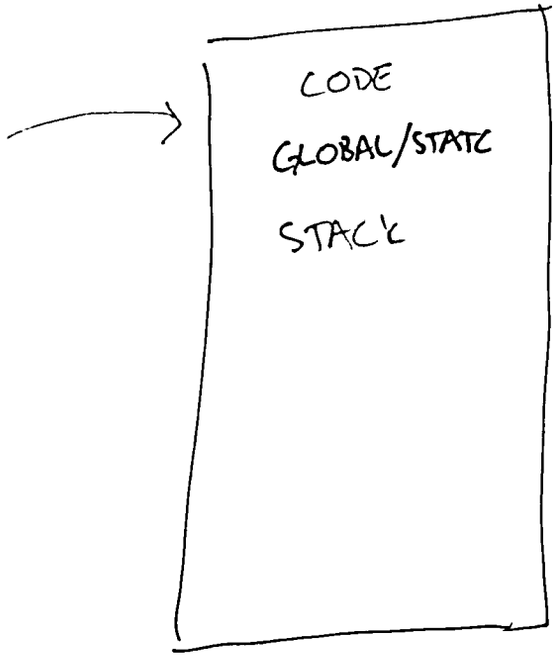
Command-line args

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



int parse_args(char* input, char** args) {
  char* current = strtok(input, " ");
  int current_index = 0;
  while(current != NULL) {
    args[current_index] = current;
    current = strtok(NULL, " ");
    current_index += 1;
  }
  return current_index;
}

int main() {
  while(1) {
    printf(" ");
    char input[2048];
    fgets(input, 2048, stdin);
    printf("Now computer run this: %s\n", input);

    char* args[1000];
    int argc = parse_args(input, args);
    for(int i = 0; i < argc; i += 1) {
      printf("args[%d]: %s\n", i, args[i]);
    }









  }
}
```

fork()

makes a copy of the current process

- same code keeps running in both



..... happy c code

fork()

printf("hi") printf("hi")

- same globals, same stack, etc.
- only difference is in "child" (new) process returns 0
  in "parent" (original) returns pid

```
int pid = fork();
if (pid == 0) {
    // child stuff (new work)
}
else {
    // parent stuff (continue loop/monitor/etc)
}
```