my_strsep() Lab

Steps:

1. Write your own version of strsep()
2. Have fun!

Stack			Heap		
Address	Name	Value	Address	Name	Value
0x9C	buf		0x34		
0x98			0x30		
0x94			0x2C		
0x90			0x28		
0x8C			0x24		
0x88			0x20		
0x84			0x1C		
0x80			0x18		
0x7C			0x14		
0x78			0x10		

placing an `&` (ampersand) before a variable name gives that variable's memory address
address of operator

placing an `*` (asterisk) before a variable name gives the value that variable is pointing to
dereference operator

0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F 0x20

buf | t | e | s | t | , | s | t | r | i | n | g | . | p | l | e | a | s |

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6

in C, only a single value may be returned from a function → how can we return multiple values from a function in C?

if you change the value at a variable's address, you change the variable's value

Write a function that can modify variables declared outside its own scope!

terminal

```
$ ./wc msno.txt
```

argc: 2

argv: ["./wc", "msno.txt"]

process

```
main(int argc, char **argv) {

1.  check args
2.  open() input file
3.  read() a chunk of 1023 bytes
                repeatedly...

buffer
  abc def\nghij klmno

4.  strsep() using " \n"
5.  lfind() for that token in the
    data structure
    1. add a new entry if not found
    2. increment found entry
6.  print_and_free()
7.  clean up                    user
```

creates...

kernel

file pointer: 9987

number of bytes from the beginning of the file, the file pointer's offset

A0

```
print_and_free(WORD_T *words, size_t total_words, char *infile) {

                                                    user
```

Functions to study:

difficulty

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
```
easy

```
#include <fcntl.h>

int open(const char *pathname, int flags);
```
easy

```
#include <string.h>

char *strsep(char **restrict stringp, const char *restrict delim);
```
tricky

```
#include <search.h>

void *lfind(const void *key, const void *base, size_t *nmemb,
            size_t size, int(*compar)(const void *, const void *));
```
This entire thing is a single function parameter!

wtf

```
#include <stdlib.h>

void *malloc(size_t size);
```
easy

```
#include <stdlib.h>

void *realloc(void *ptr, size_t size);
```
easy

```
#include <string.h>

void *memmove(void *dest, const void *src, size_t n);
```
tricky