Day 9 (Fri 02/11)

- Exercise 8 review
- Day 9 recap questions
- Exercise 9

Reminders:

- HW1 due *this evening* by 11pm
- HW2 due Friday, Feb 18th by 11pm
  - late submissions not allowed!

slido.com
jhuintprog01

Exercise 8 review

Checking whether result capacity is sufficient:

```
int word1_len = strlen(word1);
int word2_len = strlen(word2);

if (word1_len + word2_len + 1 > result_capacity) {
    return 0;
}
```

amount of space available in the result array

amount of space required for concatenation of word1 & word2

Exercise 8 review (continued)

Concatenating word1 and word2, placing concatenation in result

  - copy all characters from word1 into result
  - copy all characters from word2 into result
  - place NUL terminator after all of the characters from
    word1 and word2

word1
length=5

word2
length = 3

result

NUL terminator
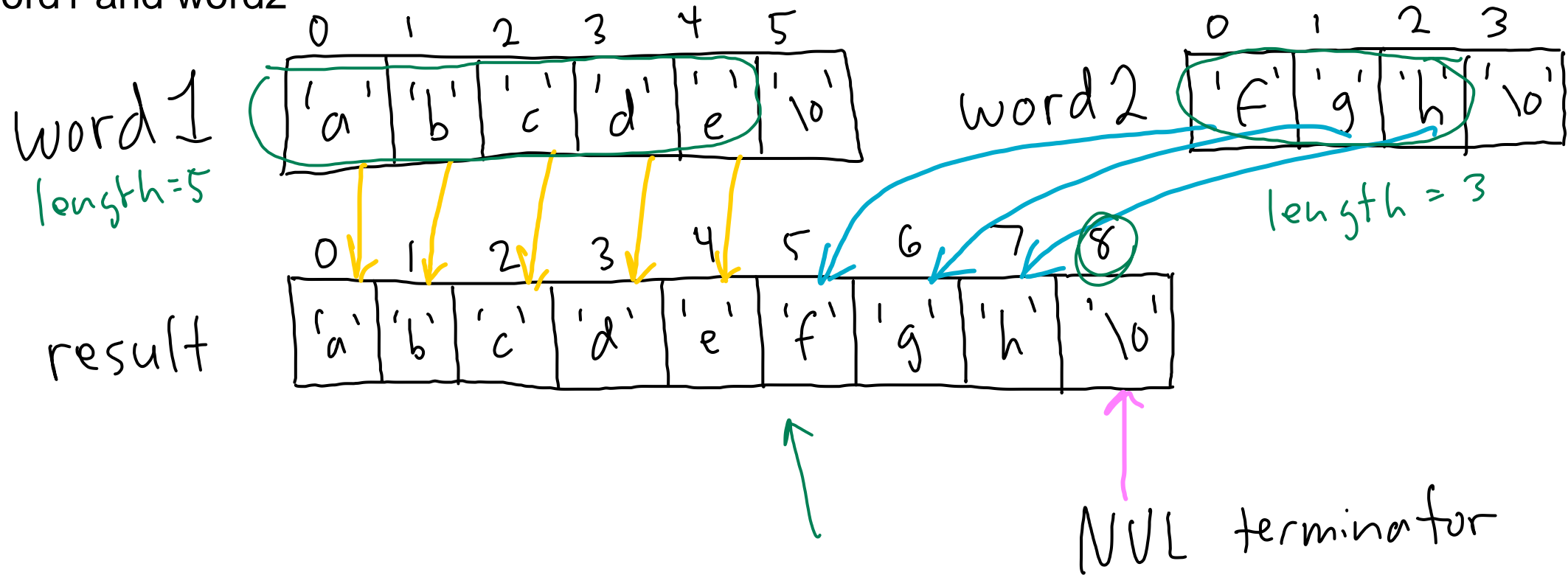
Exercise 8 review (continued)

Modularizing the program

main function goes in run_concat.c

concat function goes in string_functions.c

concat function declaration goes in string_functions.h

it should look like this:

```
#ifndef STRING_FUNCTIONS_H
#define STRING_FUNCTIONS_H

int concat(const char word1[], const char word2[], char result[], int result_capacity);

#endif // STRING_FUNCTIONS_H
```

header guard

C source file:
collection of func. defs

C header file:
func. declarations
#define
struct types.

function declaration
for the concat function

Exercise 8 review (continued)

Both run_concat.c and string_functions.c should

  #include "string_functions.h"

Why? This allows the compiler to check that function definitions are consistent with their declarations

Example scenario: you add a parameter to the concat function in string_functions.c, but forget to add the parameter to the declaration in string_functions.h. Now all of the calls to concat are wrong (they pass the wrong number of arguments), but the compiler won't warn you about it.

Editing the Makefile

string_functions.o : string_functions.c
string_functions.h

The Makefile is already set up correctly. All that should be needed
is to edit the file names.

gcc · · ·

If it's working correctly and you run "make" for the first time, it should
run the following commands:

```
gcc -std=c99 -pedantic -Wall -Wextra -c run_concat.c
gcc -std=c99 -pedantic -Wall -Wextra -c string_functions.c
gcc -o run_concat run_concat.o string_functions.o
```

One suggested improvement: change

CFLAGS=-std=c99 -pedantic -Wall -Wextra

to

CFLAGS=-std=c99 -pedantic -Wall -Wextra -g

include debug symbols
so debugging the program
in gdb works correctly

What happens when the user enters more than 10 characters for a string?

scanf will store characters past the end of the array. This is undefined behavior, and will generally corrupt other variables in the program.

If the overrun is large enough, the main function's return address will be corrupted, and the program will (likely) segfault when it tries to return.

Soon: we will use valgrind to detect this type of error.

Day 9 recap questions

1. How do you declare a multi-dimensional array and pass it to a function?
2. How do you initialize a multi-dimensional array using array initialization?
3. What is the compile flag needed to compile a program such that we can debug it using gdb?
4. How do you set a break point using gdb and check the call stack?
5. Check the gdb cheat sheet and find the command to print the content of a variable per step, instead of only printing it once using `print`.

1. E.g. a two dimensional array:

```
  int chessboard[8][8];
```

For 2-D arrays, first dimension is "rows" and second dimension is "columns".

Passing to a function:

can omit

```
  int is_checkmate(int board[8][8]) {
    // ...code...
  }
```

must be explicit

Note that the size of the first array dimension can be omitted, but the sizes of the other array dimensions are required.

2. E.g.:

```
// assume 0=empty, 1=pawn, 2=rook, 3=knight, 4=bishop, 5=king, 6=queen
// note that we should really have #define constants or an enum
// type so that these values can have meaningful names

int chessboard[8][8] = {
  { 2, 3, 4, 6, 5, 4, 3, 2 },
  { 1, 1, 1, 1, 1, 1, 1, 1 },
  { 0, 0, 0, 0, 0, 0, 0, 0 },
  // etc. for other rows
};
```

row 0   (chessboard [0][0], chessboard [0][1], etc...)

row 1   (chessboard [1][0], chessboard [1][1], etc...)

gdb myprog

(gdb) command

3. -g: is required when compiling, but not required when linking

4. break name_of_function
   break source_filename.c:N   (N is the line number)

backtrace

5. display name_of_variable