601.220 Intermediate Programming

Spring 2023, Day 9 (February 10th)

Today's agenda

- Exercise 8 review
- Multidimensional arrays, gdb
- Exercise 9

Reminders

- HW1 due *today*
- HW2 due Friday, Feb 17th
  - "Written" assignment (on Gradescope), late submissions will not be accepted

Exercise 8 review

```c
int concat(const char word1[], const char word2[],
    char result[], int result_capacity){
  int word1_len = strlen(word1);
  int word2_len = strlen(word2);
  if (word1_len + word2_len + 1 > result_capacity) {
    return 1; // not enough room in result array
  }
  int pos = 0;
  for (int i = 0; i < word1_len; i++) {
    result[pos] = word1[i];
    pos++;
  }
  for (int i = 0; i < word2_len; i++) {
    result[pos] = word2[i];
    pos++;
  }
  result[pos] = 0;
  return 0;
}
```

don't call in loop

Exercise 8 review

string_functions.h:

```
#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
```

Exercise 8 review

string_functions.c:

```c
#include <string.h>
#include "string_functions.h"

int concat(const char word1[], const char word2[],
    char result[], int result_capacity){
  // ...code omitted...
}
```

Exercise 8 review

```
run_concat.c:

#include <stdio.h>
#include <string.h>
#include "string_functions.h"

int main() {
  // ...code omitted...
}
```

Exercise 8 review

```
# Makefile
CC = gcc
CFLAGS = -std=c99 -pedantic -Wall -Wextra

run_concat: run_concat.o string_functions.o
    $(CC) -o run_concat run_concat.o string_functions.o

run_concat.o: run_concat.c string_functions.h
    $(CC) $(CFLAGS) -c run_concat.c

string_functions.o: string_functions.c string_functions.h
    $(CC) $(CFLAGS) -c string_functions.c

clean:
    rm -f *.o run_concat
```

Day 9 recap questions

❶ How do you declare a multi-dimensional array and pass it to a function?

❷ How do you initialize a multi-dimensional array using array initialization?

❸ What is the compile flag needed to compile a program such that we can debug it using gdb?

❹ How do you set a break point using gdb and check the call stack?

❺ 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. How do you declare a multi-dimensional array and pass it to a function?

Declaring a two-dimensional array:

```c
char board[3][3];
```

Accessing an element:

```c
board[0][2] = 'X';
```

Note that by convention, the first index is "rows" and the second index is "columns".

2-D array as parameter

char board [ ][3]

```c
void print_board(char board[3][3]) {
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      printf("%c", board[i][j]);
    }
    printf("\n");
  }
}
```

Note that the first dimension can be omitted, but the other dimensions are required.

2. How do you initialize a multi-dimensional array using array initialization?

Example:

```
char board[3][3] = {
row 0 {'O', 'X', 'X'},
row 1 {'X', 'O', 'O'},
row 2 {'X', 'X', 'O'},
    };
```

3. What is the compile flag needed to compile a program such that we can debug it using gdb?

The -g option causes the compiler to generate debug information.

Strongly recommended for all `Makefiles` for C programs:

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

4. How do you set a break point using gdb and check the call stack?

Set breakpoint at beginning of function:

```
break main
break bsearch
```

Set breakpoint at specific source line:

```
break functions.c:74
```

Print call stack (all of these are equivalent):

```
where
backtrace
bt
```

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`.

```
display
```

Exercise 9

- Two-dimensional arrays
- Debugging using gdb
- Talk to us if you have questions!