CS 214: System Programming

(Sec 02 10:35 AM - 11:30 AM ARC-105)

Zhe(Jay)Wang
10/05/2017
Email: jay.wang@rutgers.edu
Office Hours: Friday 2:30pm-4:30pm at **Hill 264A**

What will we do in this recitation?

- Reflection About Project Sorting
- Review malloc's placement strategies
- Introduction of using Valgrind
- Homework

- Reflection About Project Sorting

Segmentation fault (GDB run, analyse the context)
== and =
Rules of sorting (string, integer, float) (20 180)

$\downarrow$ How to avoid?

Unit Test
(test driven)   clear defined input & output   debugging tools   more patience

what if a function fail??? if the input parameter is a invalid one??? null pointer?

Coding Standard in C   Review with each other   Don't Rush

Malloc Placement Strategy

refer to :
http://cs241.cs.illinois.edu/wikibook/memory-part-1-heap-memory-introduction.html#what-are-placement-strategies

https://github.com/angrave/SystemProgramming/wiki

https://stackoverflow.com/questions/1518711/how-does-free-know-how-much-to-free

https://stackoverflow.com/questions/2759845/why-is-address-zero-used-for-the-null-pointer

What happens when I call malloc?

Can malloc fail?

Where is the heap and how big is it?

Do programs need to call brk or sbrk?

What is calloc?

Why is the memory that is first returned by sbrk initialized to zero?

Why doesn’t malloc always initialize memory to zero?

What is realloc and when would you use it?

Where can I read more?

How important is that memory allocation is fast?

What is the silliest malloc and free implementation

and what is wrong with it?

What are placement strategies?

What is external fragmentation?

What effect do placement strategies have

on external fragmentation and performance?

What are the challenges of writing a heap allocator?

How do you implement a memory allocator?

What is the silliest malloc and free implementation and what is wrong with it?#

```
void* malloc(size_t size)
// Ask the system for more bytes by extending the heap space.
// sbrk Returns -1 on failure
   void *p = sbrk(size);
   if(p == (void *) -1) return NULL; // No space left
   return p;
}
void free() {/* Do nothing */}
```

refer: http://cs241.cs.illinois.edu/wikibook/memory-part-1-heap-memory-introduction.html

Valgrind Demo

Tool to detect memory management bug

https://www.youtube.com/watch?v=bb1bTJtgXrI&t=2s

HW last time

1. Write some code that declares two arrays of size 10 that are string literals.

Make a pointer to one of the arrays, cast it to be an int pointer, and print out its value.

Make a new integer, set it equal to the value of your int pointer, then make a pointer to that integer, cast it to be a char pointer, and print out 8 chars.

What happened? Why?

HW

- 0. What are the differences between strlen and sizeof a string in C? Why?

HW

1. Write the function

```
replace(char string[], char from[], char to[])
```

which finds the string from in the string string and replaces it with the string to. You may assume that from and to are the same length. For example, the code

```
char string[] = "recieve";
replace(string, "ie", "ei");
```

should change string to "receive".

HW

2.
Write a short program to read two lines of text, and concatenate them using strcat. Since strcat concatenates in-place, you'll have to make sure you have enough memory to hold the concatenated copy.

For now, use a char array which is twice as big as either of the arrays you use for reading the two lines. Use strcpy to copy the first string to the destination array, and strcat to append the second one.