Memory Errors

CSC 213 – September 12, 2022
Agenda for Today

1. Q&A
2. Code Quality Updates
3. New Assignment
4. Memory Errors
5. Address Translation Practice *(if there's time)*
6. Wrap Up
Q&A
Assignment Questions

Can the print function display a whitespace before the newline at the end?
   No! The autograder won't complain, but the human grader will.

Do we have to use memmove?
   Not for this assignment (although I recommend it), but in the future, yes.
Lab Questions

Do we need to support exit and cd with ampersands or semicolons?
   Ideally these would work with semicolons. They're nonsensical with
   ampersands so that doesn't have to work. Neither is required for full credit.
Code Quality Updates
New Graded Elements

Check Error Codes
You must check for errors from any standard library function that can return an error code, with the following exceptions:

- printf
- fprintf
- malloc
- realloc

Do Not Duplicate Standard Functions
Don't write functions to do something a standard function can already do. You don't have to know about every library function, but I do expect you to use these:

- strsep
- strtok_r
- strpbrk
- getline
- memcpy
- memmove
- realloc

plus the familiar ones from CSC 161
New Graded Elements

Avoid Unnecessary Memory Allocations

If you can accomplish a task using locals instead of `malloc`-ed memory, please do so. This is sometimes a judgement call, and I'm happy to discuss it with you.

These new criteria are in effect for the shell lab (not the sorted list assignment). I made comments about these issues on the ngram assignment but did not deduct points. After sorted-list these will also apply to assignment grading.
New Assignment
Memory Errors
Discussion of Memory Errors

For each type of memory error, answer the following questions:

**What causes this error?**
Describe a situation or give a code example.

**What is the immediate effect of the error?**
At the moment the incorrect code executes, what happens? We're not worried about the long-term consequences yet—just the immediate effect.

**What are the long-term consequences of the error?**
Does the program crash? Is data corrupted? Does the error always manifest the same way?
Types of Memory Errors

Invalid Pointer Write (forgetting to allocate memory)
We have a pointer, which we use before assigning it to valid allocated memory.
For example: `char* s; strcpy(s, "Hello");`
The program will access an unknown memory location (depending on the uninitialized value of s in the above example).
Long-term consequences: segfault, access garbage data, or overwrite other data.

Uninitialized Read
We access the value of a memory location that we have never written to.
The program will use some unknown value in our calculations/operations.
The compiler might warn us (yay!).
Many bad things could happen. We could crash with a segfault, calculate something with garbage inputs, etc.
Types of Memory Errors

**Use After Free** (dangling pointer)
Allocate memory, free it, and then continue using it (reading or writing to it). Unless free overwrites the freed memory (it usually doesn't), the program will see the data it left in the memory before freeing it. The code might work fine, or any of the behaviors with an uninitialized read could occur if the memory has been overwritten. If the memory is re-allocated to another malloc call, that data could be overwritten at any time.
Types of Memory Errors

Buffer Overflow

Copy a string to a malloc-ed location, but don't request enough space for the null terminator. Any time we use more bytes at a location than we requested. The program will write beyond the end of the requested memory. We might get away with this because malloc (and locals) can be larger than requested. Or, the code could overwrite another variable/memory location. We could also get a segfault if the write goes beyond accessible memory.
Types of Memory Errors

Double/Invalid Free
A double free is when the same location is passed to free twice. An invalid free is when we try to free any pointer that didn't come from malloc. The memory allocator may detect the error and tell us about it. If the memory was re-allocated since the first free, the second free will free it. The memory allocator may "double count" the memory, meaning it could be allocated twice. With an invalid free, any local could be overwritten.

Memory Leak
We allocated memory, but we did not free it. At first, the program will be fine. If the leaking continues, eventually the program may run out of memory.
AddressSanitizer
(your new best friend)
Wrap Up
Reminders

Reading
Review the Virtual Memory lab before Wednesday's class

Assignment
The sorted-list assignment is due tonight.
The archive printer assignment is due next Monday.

Lab
The shell lab is due on Wednesday (9/14) for full credit.
Turn the lab in by the following Wednesday (9/21) for up to 80% credit.