Style, Corruption, and `make`

Lecture 21
Class 23 of 28 | April 11th 2023 | COMP 211-002 | Joshua Bakita

Welcome!

Today:

- ➔ Revisiting style from A2
- ➔ Review on consequences of memory corruption
- ➔ The `make` command

Logistics:

- ➔ Final exam exceptions: https://eef.oasis.unc.edu/
- ➔ For regrade rqs, prefer Gradescope or Pizza
- ➔ Research opportunity if you get an A/A-

Fun fact...

vim is highly configurable—just put any commands you want run when it starts in ~/.vimrc.

Examples:

Semi-automatic indenting:

```
set cindent
```

Tab width:

```
set tabstop=<num chars>
set shiftwidth=<num chars>
```

Assignment 2 Style Review

Average style grade: B

Assignment 2 Style Review

Common Mistakes

Avoid these mistakes in subsequent assignment submissions!

General feedback:

- Check return codes of important library functions
- Return early on failure
- Use "else" in chains of ifs for efficiency
- Avoid >80 character lines
- Avoid meaningless conditionals
  - *eg. check <0 on unsigned int*
- Avoid repeat function calls
- Avoid mixing tabs and spaces

Helpful Tricks

To execute a vim command:

1. Enter normal mode (Esc)
2. Press the colon key, ":"
3. Type your command
4. Press Enter

What I use in my ~/.vimrc:

```
set nowrap
set tabstop=4
set shiftwidth=4
set list listchars=tab:»·,trail:·
set cindent
```

Vim Configuration

Semi-automatic indenting:

- set cindent

Show margin indicator for 80-character lines:

- set colorcolumn=80

Auto-convert tabs to spaces:

- set expandtab

Render tabs using alternate symbols:

- set list listchars=tab:»·

Don't wrap long lines:

- set nowrap

Memory Layout & Corruption

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define MAX_SIZE 1024
char inputt[MAX_SIZE];
char output[MAX_SIZE];

int main() {
    // Read up to MAX_SIZE characters from standard input
    int res = fread(inputt, 1, MAX_SIZE, stdin);
    if (res == 0 && ferror(stdin)) {
        perror("Unable to read from stdin");
        return 1;
    }
    // Translate to upper case
    for (int i = 0; i < res; i++)
        output[i] = toupper(inputt[i]);
    // Make sure that output is NULL-terminated
    output[res] = '\0';
    // Output results
    printf("Original input: \"%s\"\n", inputt);
    printf("Translated to all-caps: \"%s\"\n", output);
    return 0;
}
```

Try it yourself!

```
$ wget https://www.cs.unc.edu/~jbakita/teach/comp211-s23/l21/upper.c
$ gcc upper.c -o upper
$ echo "Hello world" | ./upper
```

This program claims empty input if our input is >= 1024 characters. Where does it go wrong?

https://PollEv.com/joshuabakita182

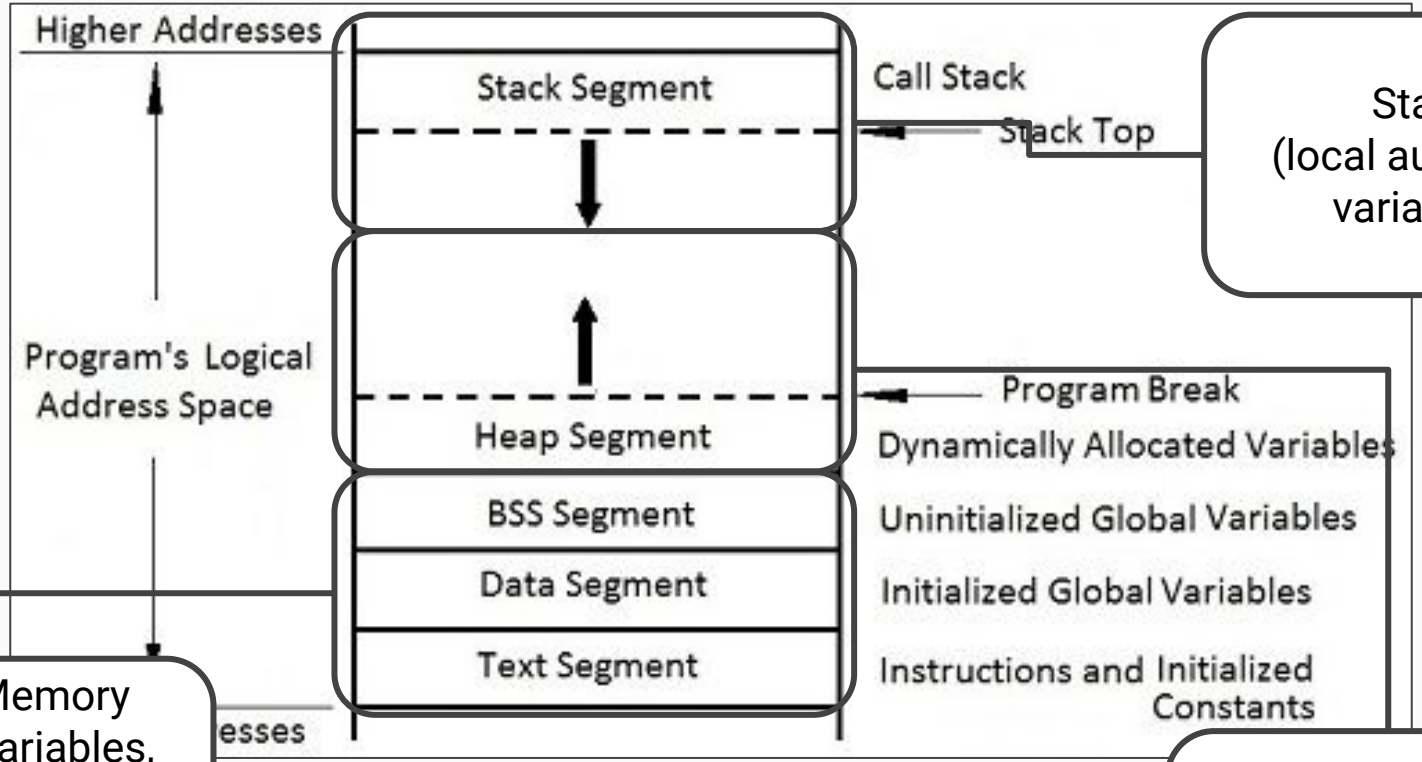

Image credit: Aman Vats

Stack
(local automatic variables)

Static Memory
(Global variables, `static` variables, string literals, compiled code)

Heap Memory
(malloc() et al.)

The `make` command

A great way to keep from accidentally deleting your code…

Key `make` terms

- *target*
- *recipe*
- *prerequisite*

See the `make` manual for an extensive discussion:
https://www.gnu.org/software/make/manual/make.pdf

Page 131 discusses automatic variables

Questions?

Contact:
Email: hacker@unc.edu
Twitter: @JJBakita
Web: https://cs.unc.edu/~jbakita

Old Well, University of North Carolina at Chapel Hill, Winter 2017