C Introduction, Memory

CS 2130: Computer Systems and Organization 1

Xinyao Yi Ph.D.
Assistant Professor

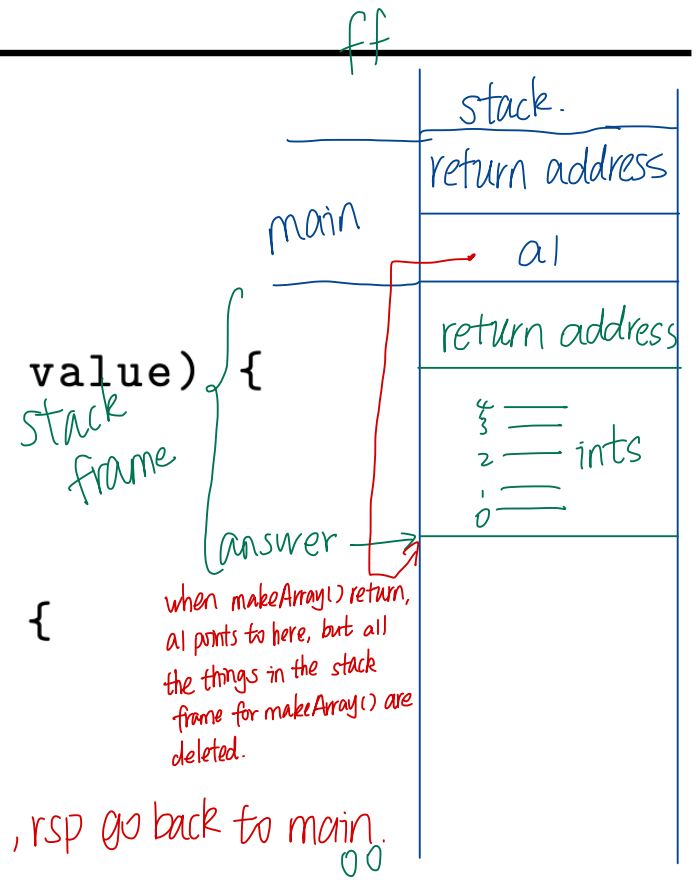
UNIVERSITY of VIRGINIA | ENGINEERING

Announcements

- Homework 8 due Monday on Gradescope
- Quiz 8 out today, due Sunday night

An Interesting Stack Example

```
int *makeArray() {
    int answer[5];
    return answer;
}

void setTo(int *array, int length, int value) {
    for(int i=0; i<length; i+=1)
        array[i] = value;
}

int main(int argc, const char *argv[]) {
    int *a1 = makeArray();
    setTo(a1, 5, -2);
    return 0;
}
```



once makeArray returns, remove everything, rsp go back to main.

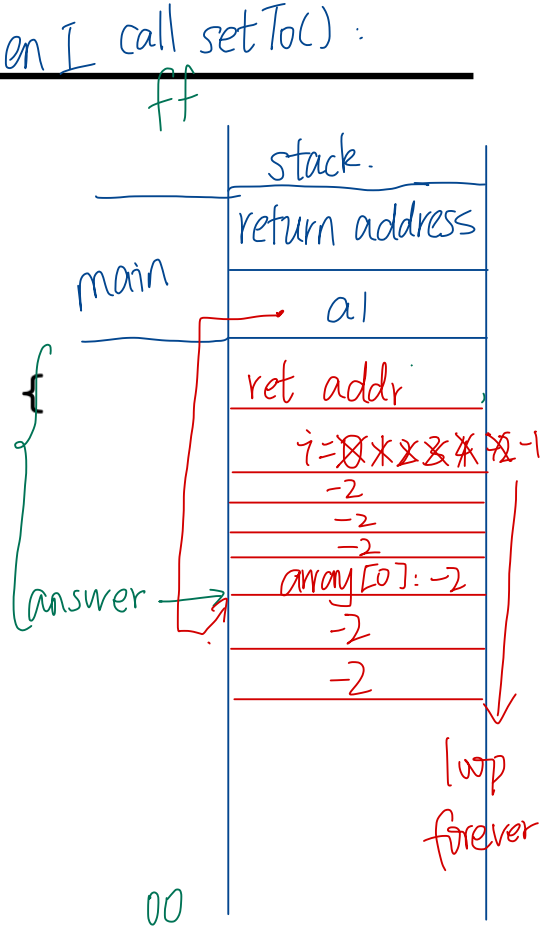
An Interesting Stack Example

```
int *makeArray() {
    int answer[5];
    return answer;
}

void setTo(int *array, int length, int value) {
    for(int i=0; i<length; i+=1)
        array[i] = value;
}

int main(int argc, const char *argv[]) {
    int *a1 = makeArray();
    setTo(a1, 5, -2);
    return 0;
}
```

escaping pointer

in registers

stack frame



if I compile it using clang xxx.c, then it will loop forever.
But if I compile it with "-O1". same warning, but it optimize the code
maybe put "i" in register, maybe do unrolling?

The Heap

The heap is a section of memory that's different from the stack.
Anything you put on the heap will persist past function calls and returns — it stays there until you explicitly free it.
That means you can allocate space in one function, pass a pointer to another function, and still use that same memory later.
It's a part of memory that your program is allowed to use for dynamic data that needs to live longer than a single function call.

The heap: unorganized memory for our data

- Most code we write will use the heap
- *Not a heap data structure…*

Early computer scientists decided "heap" was a great name for this memory space —
but later someone also thought "heap" would be a great name for a data structure.
So remember: the heap in memory is not a heap data structure.
They just happen to share the same name.
You'll probably meet the other heap again in your Data Structures course.

The Heap: Requesting Memory

The way we ask for memory on heap is with a function called malloc.

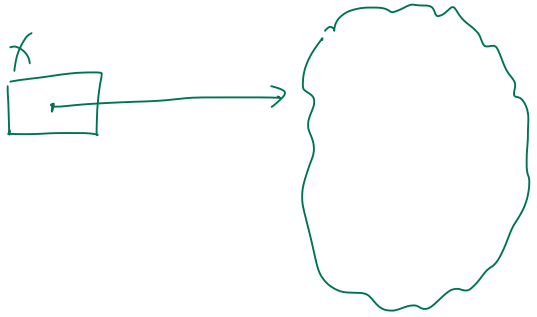
```
void *malloc(size_t size);
```

- Ask for `size` bytes of memory
- Returns a `(void *)` pointer to the first byte
- It does not know what we will use the space for!
- Does not erase (or zero) the memory it returns

Java

What is the closest thing to malloc in Java?

```
MyC x = new MyC();

int[] y = new int[100];
```



malloc **man page**

`calloc and realloc`

malloc **Example**

```
typedef struct student_s {
    const char *name;
    int credits;
} student;

student *enroll(const char *name, int transfer_credits) {
    student *ans = (student *)malloc(sizeof(student));
    ans->name = name;
    ans->credits = transfer_credits;
    return ans;
}
```

(*ans).name

if I don't free → run out the memory, things get really slow.

The Heap: Freeing Memory

Freeing memory: `free`

```
void free(void *ptr);
```

We will use the pointer to free the space. If I change the pointer, I will lose the information of where to free.

- Accepts a pointer returned by `malloc`
- Marks that memory as no longer in use, available to use later
- You should `free()` memory to avoid *memory leaks*

When my program ends, the operating system will come in and free the memory for me.

An Interesting Stack Example

```
int *makeArray() {
    int answer[5];
    return answer;
}

void setTo(int *array, int length, int value) {
    for(int i=0; i<length; i+=1)
        array[i] = value;
}

int main(int argc, const char *argv[]) {
    int *a1 = makeArray();
    setTo(a1, 5, -2);
    return 0;
}
```

int * answer = malloc(5 * sizeof(int));

int * answer = (int *) malloc(5 * sizeof(int));
↳ Better! But if you don't have it, it's fine.
because malloc returns a void*. it will automatically cast to int*.

free(a1);

Garbage

Garbage - memory on the heap our code will never use again

- Weird: defined in terms of the future!
- Compiler can't figure out when to free for you

It cannot tell what is actually garbage and what is not.

Garbage

Garbage - memory on the heap our code will never use again

- Weird: defined in terms of the future!
- Compiler can't figure out when to free for you

What about Java?

Does Java have something like free? Set the pointer to Null ? JAVA has the garbage collector.

Garbage Collector

"The garbage collector goes through and frees memory automatically that I'm not using anymore."

But it doesn't free all of garbage. It was not possible.

Garbage Collector - frees garbage "automatically"

- **Unreachable memory** - memory on heap that is unreachable through pointers on the stack (or reachable by them)
  - Subset of all the garbage
  - Identifiable!
- Takes resources to work
- *Very* popular - most languages have garbage collectors
  - Java, Python, C#, …

"Imagine your program stack has pointers to heap objects, and those objects might have references to other objects…"
The garbage collector pauses your program, walks through memory, and finds all heap objects that cannot be reached from the stack."

"This takes resources to work — it pauses your program and makes Java a little slower, but it's very popular."

"Most languages have it — only a few don't, like C, C++, and Fortran. Some make it optional like Rust, D, and Go."

Common Memory Bugs (reading)