“Success consists of going from failure to failure without loss of enthusiasm.”

Winston Churchill

Memory Layout

Bryce Boe

2013/10/07

CS24, Fall 2013

Outline

- Lab 2 Notes
- Wednesday Recap
- Memory Layout

Lab 2 Notes

- Don’t use functions from string.h
  - No strlen, strcpy, etc
  - DO: write these functions yourself
- Don’t allocate a fixed size for the new string
  - E.g., malloc(256)
  - DO: Allocate the exact number of bytes you require
- Partner up

WEDNESDAY RECAP

What are the sizes of the following primitive types (x86– Intel 32 bit)?

- int
- float
- double
- char
- void*
- int*
- char*

How large is the structure?

```
struct MyStructA {
        int a, b;
        float f;
};
```

How large is the structure?

```
struct MyStructB {
        char msg[20];
        char *another_message;
};
```

How large is the structure?

```
struct MyStructC {
        char *message;
        struct MyStructA a;
};
```

How large is the structure?

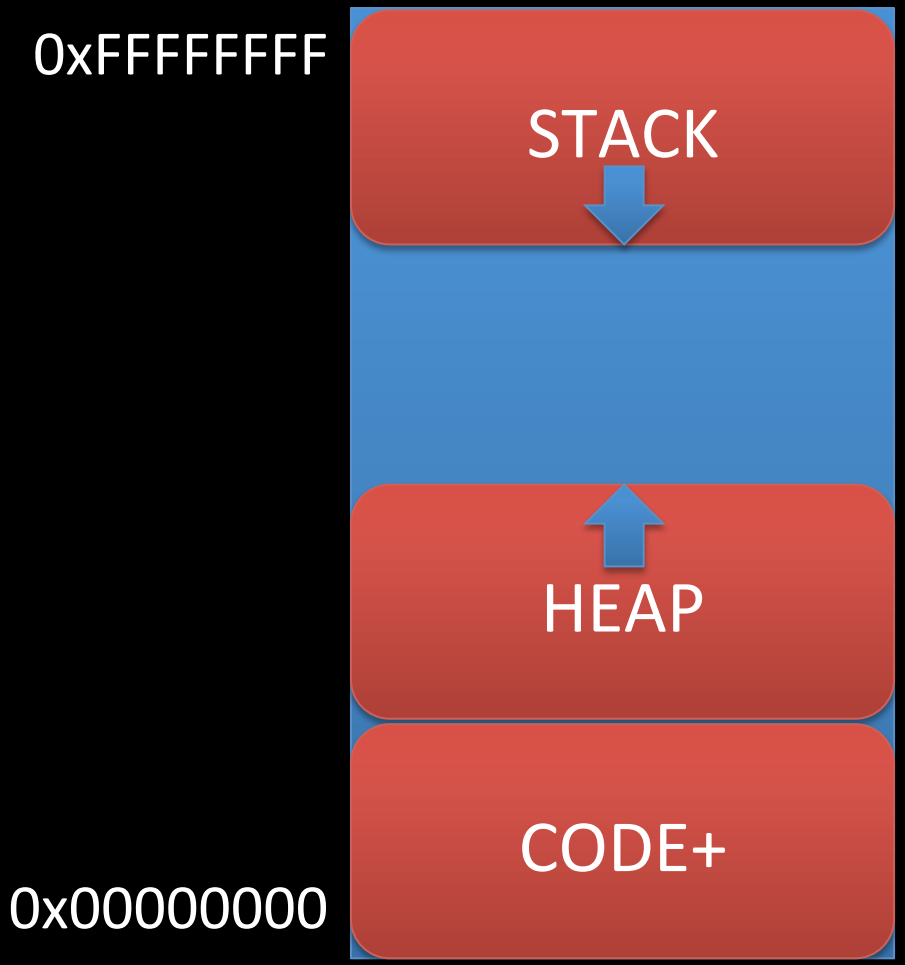
```
struct MyStructD {
        char *message;
        struct MyStructD *next;
};
```

MEMORY LAYOUT

View from three Levels

- The address space of a process
- Function activation records on the stack
- Data within an activation record

Simplified process's address space



Data Segments

- Code(+)
  - Program code
  - Initialization data, global and static variables
- Heap
  - Memory chunks allocated via malloc
- Stack
  - Function activation records

Creating and destroying activation records

- Program starts with main
- main calls strsafeconcat
- strsafeconcat calls strlength
- strlength returns
- strsafeconcat calls strlength
- strlength returns
- strsafeconcat returns

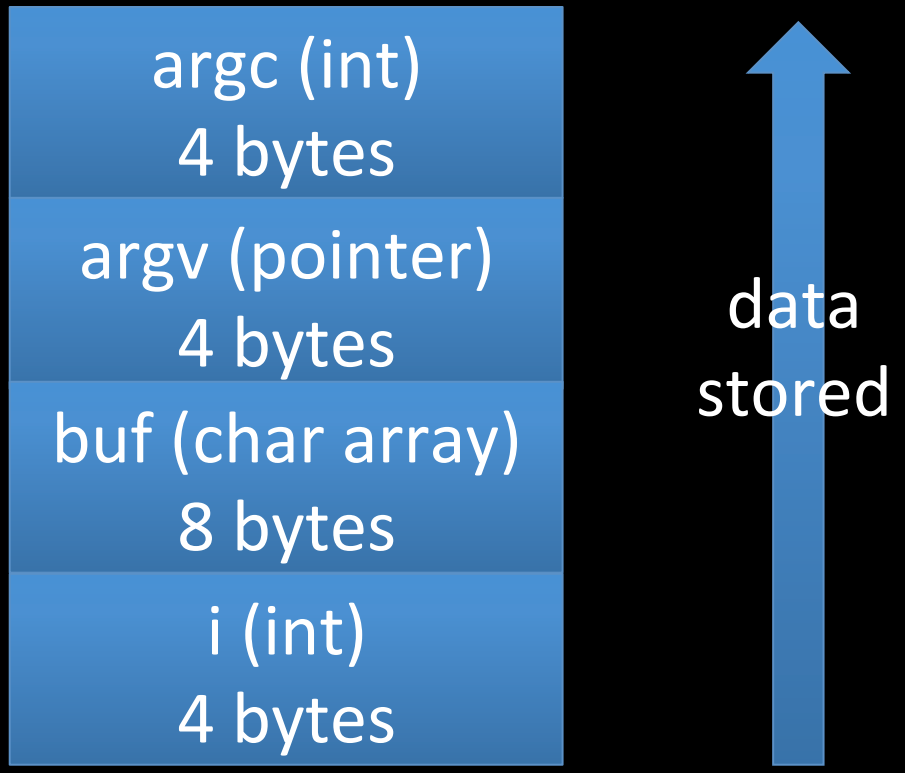
Simplified function activation records

- Stores values passed into the function as parameters
- Stores the function's local variables

How many bytes is the simplified activation record?

```
int main(int argc, char *argv[]) {
        char buf[8];
        for (int i = 0; i < argc; ++i)
                buf[i] = argv[i][0];
}
```

main's simplified activation record



Total: 20 bytes

Think about it

```
int main() {
        char msg[] = “hello”;
        char buf[] = “1234567”;
        char msg2[] = “world”;
}
```

- What value does buf[8] hold?
- What about msg[7]?

Similar but different

```
int main() {
        int x = 0xDEADBEEF;
        char buf[] = “1234567”;
}
```

- What value does buf[8] hold?

Inspecting addresses in a program

- <In class review of variables_in_memory.c>

For Wednesday

- Finish reading chapter 1 in the text book