


Processes, Stack Memory

CS2263 – Systems Software Development



1



References

Lu, Yung-Hsiang. 2015. CRC Press. New York. Pp 9-27 (Chapter 2)



2

Don't Forget To Be Awesome

- Let Me Google That For You: <https://imgtfy.com/>
- Let Me Find That On The LMS For You
 - (LMFTOTMFY)

You're seeing this site before the course starts because there are things you need to do to be ready to roll. Read through the topics that have been posted and ensure that you understand how the course will operate and have the software you need installed and you've looked at the applicable guides. *Don't be that person!*



3

Lecture Learning Outcomes

At the conclusion of this presentation students should be able to:

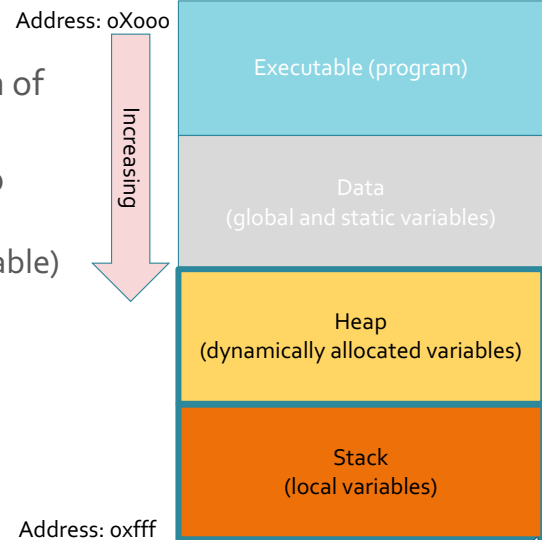
- List and describe the parts that make up a process in memory
- Explain how calling a function affects the stack
- Explain how stack frames and variable scope are related



4

Process Anatomy

- Occupies an assigned region of RAM
- Subdivided into sections:
 - Text (executable)
 - Data
 - Heap
 - Stack

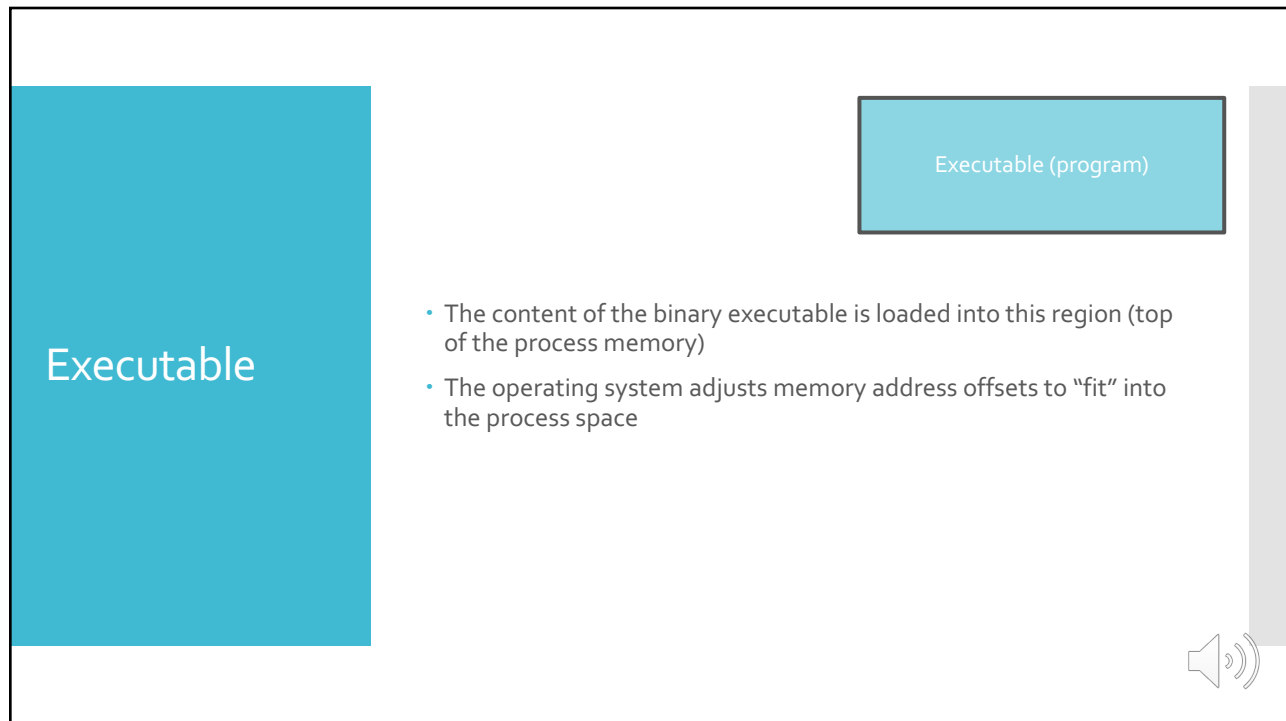


5

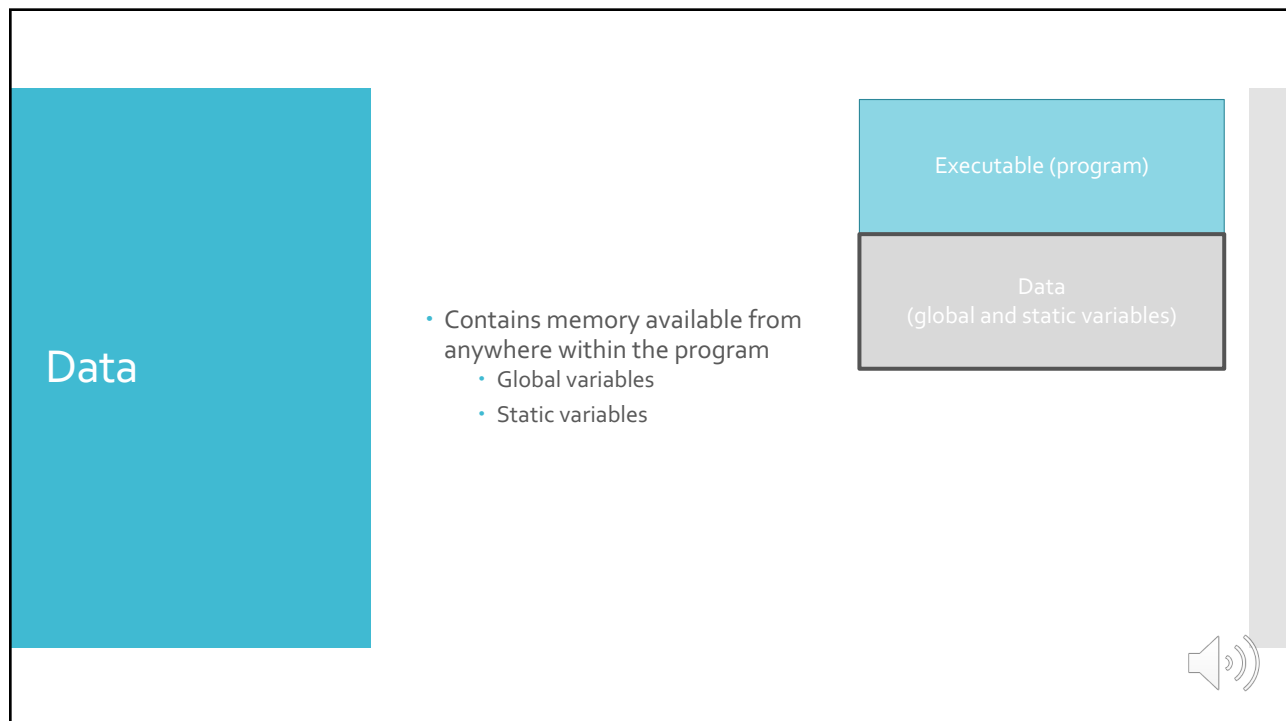
Addressing within the Process

- Addressing starts at the top/beginning of the process memory
- For our purposes, memory within a process is addressed from zero
 - Addresses numbers are natural, non-zero, non-null values

6



7

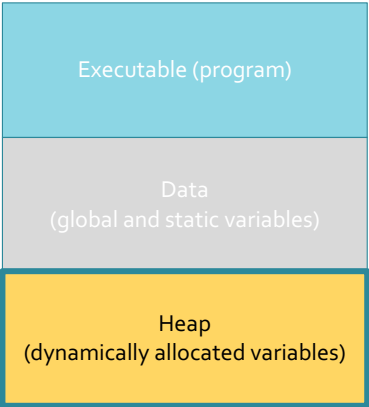


8

Heap

- Accommodates the need for programs to request memory during execution (run-time memory)
 - Java keyword `new`

```
Point pt = new Point();
int arr = new int[10];
```

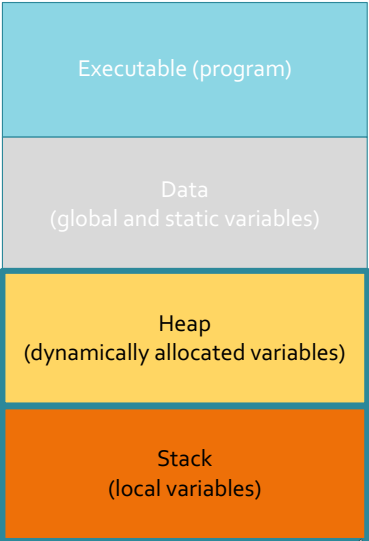


Speaker icon

9

Stack Memory

- This is where your program gets its local variable storage
- Each function has its own memory on the stack for local variables ("stack frame")
- Each function call generates a new stack frame



Speaker icon

10

Stack Frames

```
// first.c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char * * argv)
{
    int a = 5;
    int b = 17;
    printf("main: a = %d, b = %d, argc = %d\n", a, b, argc);
    return EXIT_SUCCESS;
}
```

- The OS allocates the process memory, establishes the process memory regions, loads the executable, transfers control
- `main()` begins
 - Create a stack frame!
- `printf()` begins
 - Create a stack frame!
- `printf()` ends
 - Forget the stack frame
- `main()` ends
 - Forget the stack frame
- The OS deallocates the process memory

