Week 03-1

CSC209 Fall 2023

Dr. Demetres (dee) Kostas

September 26, 2023

Announcements

- A1 due tomorrow
  - So glad to see people helping on Piazza
  - I'll do a big session of question answering
    - later this afternoon
- Lab 3 is this week
  - the other two to be graded
- A2 tentatively released
  - Friday

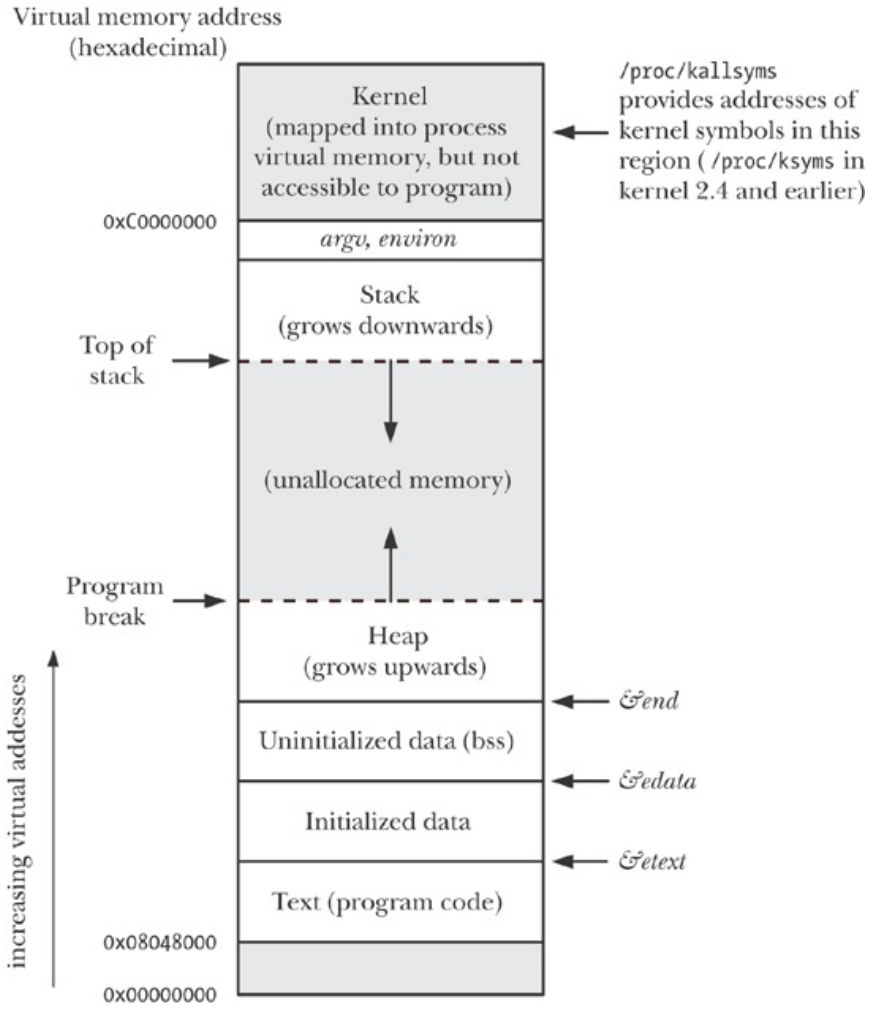
A more serious memory model

- in the past you've seen memory models
- variables local to stack frames
  - addresses for variables
- This is probably the first time
  - you've seen where *code* is in memory
  - and this thing called the **heap**
    - as well as some others

Segments

- These are the broad categories:
- **Text:** program instructions as machine-interpretable bytes
- **Data:** static variables like string literals, global vars
- **Heap:** dynamically (self) managed program memory
- **Stack:** LIFO for context frames
  - e.g. a function's context
    - space for *local* variables

Visual Layout



Kerrisk 2010 – Figure 6-1

Using the heap

- this is where *serious* memory is used
- you need to ask for a
  - **specific number of bytes**

- you do this using `malloc(<num_bytes>)`
  - the `sizeof` function helps
- returns a `(void *)`
  - a pointer that doesn't dereference
    - until you re-understand (cast it)
      - as a different type

`free`-ing memory

- the `free` system call
  - is the opposite of malloc
- give it the start of `malloc`'d memory
  - and that memory is no longer yours
    - no longer reserved for program
- **But**, if you *loose* this pointer
  - or loose the value of the address
    - (you can't call free)
  - *you have a memory leak*
    - if you can't or don't call `free`

`malloc_basics.pdf`