CSC209 Summer 2015 — Software Tools and Systems Programming

www.cdf.toronto.edu/~csc209h/summer/

Week 3 — May 28, 2015

Peter McCormick
pdm@cs.toronto.edu

Some materials courtesy of Karen Reid

Announcements

- Assignment 1 has been posted

- Tutorial notes on exploring with the shell (from week 1)

Agenda

- C concepts: `sizeof` and address-of `&` operator
- The memory model of the machine
    - Processes and logical address spaces
- C address pointers
- C structures

Integer Ranges in C

Type	sizeof	Min	Max
`unsigned char`	1	0	255
`signed char`	1	-128	127
`unsigned short`	2	0	65535
`signed short`	2	-32768	32767
`unsigned int`	4	0	4294967295
`signed int`	4	-2147483648	2147483647
`unsigned long`	8	0	18446744073709551615
`signed long`	8	-9223372036854775808	9223372036854775807

Memory Model

Memory Model

- System memory is can be viewed as a sequence of *bytes* (8 bit values)

- Each location in that sequence (and thus its associated value) is assigned a unique *address*

- Each address is just a number:

1st address	→	1st byte
2nd address	→	2nd byte
3rd address	→	3rd byte
…		…

`0x0001`	→	1st byte
`0x0002`	→	2nd byte
`0x0003`	→	3rd byte
…		…

Memory Model

A 32 bit address can give a unique address number
to ~4 billion (2^32) different bytes

4294967296 bytes

~4294967 thousand bytes

~4295 million bytes

~4 billion bytes

aka ~4.29 gigabytes

== 4 gibibytes (4x2^30)

Memory Model

- A 32-bit system can address, and thus is limited to, a maximum of 4GB of *addressable* system memory (RAM)

- A 64-bit system has a much higher limit (~16 billion GB worth of unique addresses, less usable in practise)

  - The CDF server *Wolf* is a 64-bit machine (with 64GB of physical RAM)

  - This is indicated by the string “x86_64 “in the output of **`uname -m`**

Memory Model

- Java and Python hide (shield?) all of this from you
- C does not

- Requires maturity and diligence to handle properly

Processes & Memory

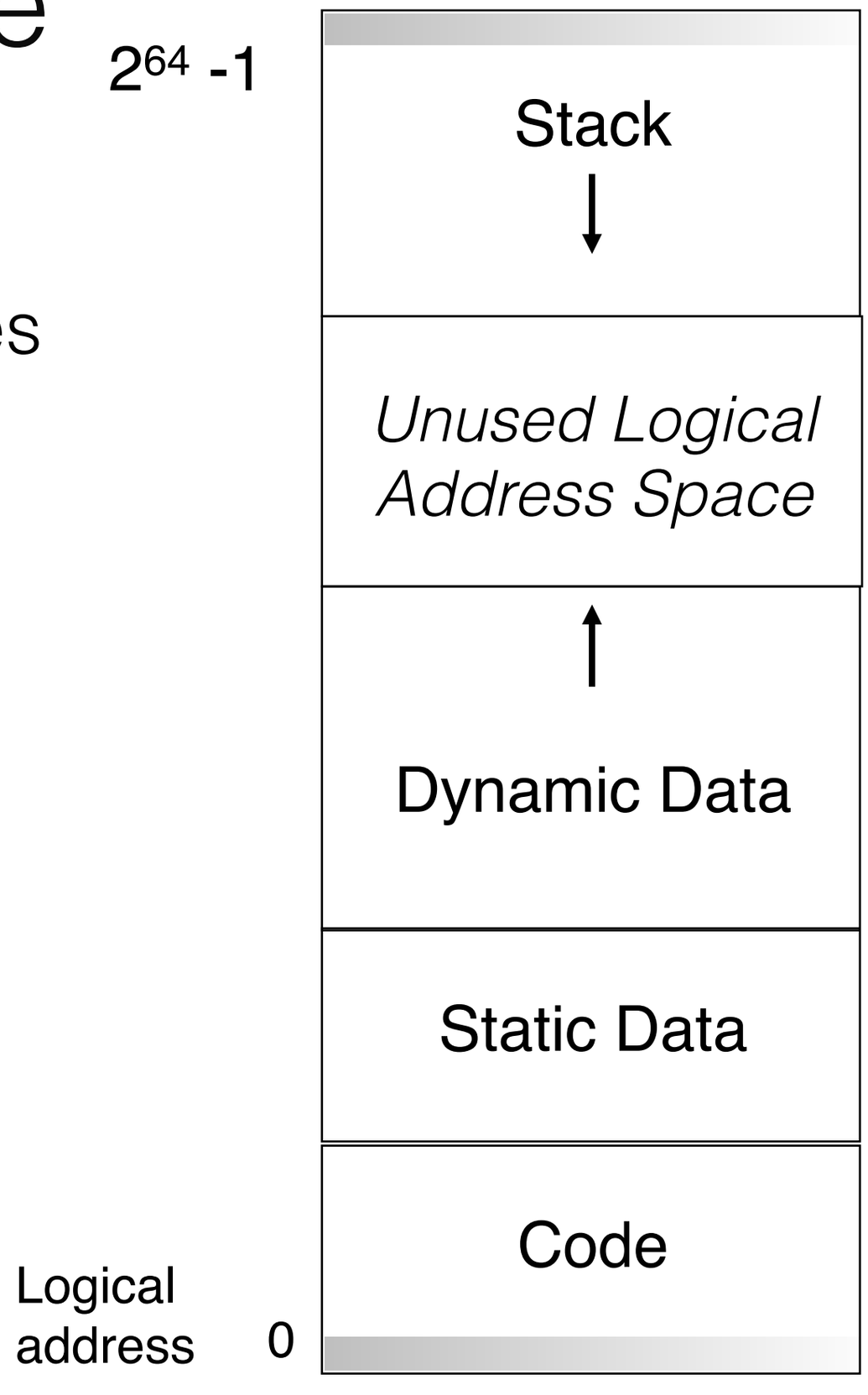
- Each *process* (a running program) on the system has its own isolated view of memory

- This sandbox is called a *logical* or *virtual address space*

- Logical addresses are mapped *onto* physical memory address by the operating system

Logical Address Space

- Memory is just a sequence of bytes
- A memory location is identified by an address
- *Code*: machine instructions
- *Static Data*: global variables and constants
- *Dynamic Data*: space your program asks for at runtime
- *Stack*: local variables, function parameters and the call stack



Pointers

… many examples …

Pointers & Arrays

`ptr[`*`i`*`]`

is equivalent to

`*(ptr +` *`i`*`)`

Pointers & Arrays

byte-address-of `ptr[i]`

==

(*byte-address value of* `ptr`) +

`(i * sizeof (*ptr))`

Assignment 1

Assignment 1 Suggestions

1. Start now

2. Carefully read the assignment (ask questions now & come to office hours if you don't understand something)

3. Play with **`wc`** and **`tar`** on CDF

4. Checkout your SVN repository and add/commit empty versions of all 6 required files (across 2 directories)

5. Extract example from **`getopt(3)`** manage as your starting point for **`wc209.c`**

6. Turn your play experiences into test cases

manpage references

`dirname(3)`

command/function

section number

1. User commands
2. System calls
3. C library functions

… and more

`wolf:~$ man *3* dirname`

http://man7.org/linux/man-pages/index.html

http://man7.org/linux/man-pages/man7/man-pages.7.html

`vi` in two minutes (1)

- `vi` is a text editing power tool
- Learning the basics will take only a moment and is an investment in your life/career
- Some variation of this editor will be available on practically all Unix systems (also, download GVim for Windows, or MacVim for OS X)
- Other editors (like Sublime) have Vi compatibility modes, so these skills are transferrable

`vi` in two minutes (2)

- Enter "**`vi` *`filename`***" from the shell prompt to start editing ***`filename`***
- Vi begins in *Normal mode*
- From Normal mode, type **`i`** to switch to *Insert mode*
  - Now type text normally and use the arrow keys to move around
  - Hit **`ESC`** to exit Insert mode and go back to Normal mode
- From Normal mode, type **`:w`** to save the file
  - Enter **`:q`** to quit
  - Enter **`:wq`** to save and quit
  - Enter **`:q!`** to quit without saving

`vi` in two minutes (3)

- Vi is a *model editor*
- Does a painter leave their paintbrush at rest *on* the canvas?
    - Why would your editor always be in the equivalent of Insert mode?
- Emacs is another popular and incredibly powerful editor you could check out (but I don't know anything about it)
- Learn more Normal mode commands
    - Very rich vocabulary of navigation and manipulation tools
- Ask me during lecture if you see me doing something and want to know what it was

Next Week

- Office hours on Tuesdays 2-4pm in BA3201

- Lecture: More on pointers, strings and the standard library

Labs

Last Name	Room	TA
A-H	BA2270	Daniel Kats
I-M	BA2240	Alexey Khrabrov
N-Z	BA2220	Michael Chiu Pan Zhang