CSE 220: Systems Programming
Midterm Review

Ethan Blanton

Department of Computer Science and Engineering
University at Buffalo
Introduction to C

- C is high level, but not very
- C exposes many architectural details
- Effective C requires understanding the machine
- POSIX and C provide simplified models of computers
Variables, Strings, and Loops

- C is a typed language
- Every variable has a type
- Variable values must match the type
- Variables have scope, and cannot be used outside that scope
- Arrays are contiguous memory locations
- Array syntax uses []
- C strings are arrays of characters
- Every C string is terminated with a zero byte
- For loop syntax
- For loops are very flexible
Conditionals and Control Flow

- All nonzero values are true conditions in C.
- All Boolean expressions use 1 for true.
- The `bool` keyword holds only 0 or 1.
- C uses short-circuit evaluation of Boolean logic.
- `if` and `switch` implement conditionals.
- Use blocks for `if` and `else`!
- Control flow is implemented with comparisons and jumps.
Memory and Pointers

- Memory locations are identified by addresses.
- Addresses are integers.
- Our system’s memory is like one large array.
- POSIX processes appear to have their own dedicated memory.
- Pointers hold addresses and have types.
- Unix processes are divided into sections.
- Pointers and arrays are closely related, but not the same.
A Tour of Computer Systems

- The architecture of the computer affects programs
- Program correctness depends on representations
- Performance depends on computer organization
- C provides very little memory protection
- People can die from bad programming
- Registers and busses have fixed widths
- Caching greatly improves performance
Structs and Dynamic Memory Allocation

- Static and automatic allocation aren’t always sufficient
- The dynamic allocator allows programmer control over memory size and lifetime
- NULL is the only guaranteed invalid pointer
- Allocated memory without an in-scope pointer leads to leaks
- C structs hold multiple fields
- Assignment of structs provides copy semantics
- Struct copying versus pointer passing has performance implications
Integers and Integer Representation

- The CPU and memory deal only in words
- Buses and registers have native word widths
- Integers have different:
  - Bit widths
  - Endianness
  - Sign representation
- One’s and two’s complement representation
Floating Point Numbers

- Numbers can have fractional portions
- Both fixed and floating point representations can be calculated in both binary and decimal
- IEEE 754 standardizes a floating point representation
- Floating point numbers have fixed precision, but variable magnitude
Alignment, Padding, and Packing

- Integers, pointers, and floating point numbers are **scalar types**.
- Arrays and structures are **aggregate types**.
- Structures can contain members of **mixed type**.
- Scalar types must be **aligned**.
- Aggregate types must **align for scalars**.
- Allocation normally aligns to the **largest type**.
- Pointer arithmetic uses **stride** in computations.
- `void *` has a **stride of 1**.
- The `void *` type can be used for **raw memory manipulation**
- Casting `void *` to another type is convenient
- Math on `void *` is **by byte**
Bitwise Operations

- C can manipulate individual bits in memory.
- Bit operations can be subtle and tricky!
- Signedness matters.
- Bit manipulations can force endianness or other representations.
A running process is related to its source code.
Processes have sections with different purposes.
The heap and stack are sections.
Heap memory lifetimes are explicitly managed.
Stack memory lifetimes are implicitly managed.
The stack grows downward in memory on x86_64.
Function calls have stack frames that store important information and automatic variables.
Stack-allocated arguments are why C is call-by-value.
License

Copyright 2019 Ethan Blanton, All Rights Reserved. Copyright 2019 Karthik Dantu, All Rights Reserved.

Reproduction of this material without written consent of the author is prohibited.

To retrieve a copy of this material, or related materials, see https://www.cse.buffalo.edu/~eblanton/.