CSE 220: Systems Programming

Midterm Review

Ethan Blanton

Department of Computer Science and Engineering
University at Buffalo

Time and Place

Your midterm will be

- on UBlearns
- at your regular lecture time

You must start in the first five minutes of lecture.

Log in early!

Resources

You may use from this semester:

- Lecture slides provided by me
- Lab handouts and lab `README.md` files
- Programming assignment handouts
- *Computer Systems: A Programmer's Perspective*[1]
- *The C Programming Language*[2]
- Notes written by you:
  - From one of the above allowed sources
  - From lecture content

Nothing else.

Not even notes written by you from another source!

Time Lapse

You must take a time lapse video of your exam.

Instructions are on Piazza.

Set it up and test it ahead of time.

Be aware of:

- Requirements of what must be visible
- Lack of feedback on UBlearns upload

Format

There will be several types of question on the exam:

- True/False
- Multiple choice
- Calculated values
- Short answer

On my tests, short answer is short answer:
typically, two words to two sentences, answer it and stop.

Introduction to C

- C is a high level language used in systems programming.
- Architectural details are important in C.
- The C/POSIX model is:
  - A dedicated machine for each program
  - Sequential execution of program instructions
  - Data is stored in accessible, addressed memory
- We explored some trivial C programs.

Variables, Strings, and Values

- 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.

Memory Allocation

- The heap is where you manually allocate memory.
- The C standard library contains a flexible allocator.
- Heap allocations are sized by the programmer.

Programming Practices

- Cultivate good work habits
- Design your programs purposefully
- Use your tools!
- Practice good style and form
- Debug with a plan

The only way to become a good programmer is to write programs.

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
- Ones' 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

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.

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

A Tour of Computer Systems

- Architectural details matter
  - Bus widths
  - Numeric properties
  - Performance details
- C and POSIX are just one possible system
- All systems have those details
- Software correctness can be critically important

Process Anatomy

- POSIX programs are laid out in sections
- ????? →
- The stack grows downward
- Automatic variables are allocated on the stack
- Stack frames track function calls
- Items removed from the stack are not cleared
- Stack-allocated arguments are how C is call-by-value

References I

Required Readings

[1] Randal E. Bryant and David R. O'Hallaron. *Computer Science: A Programmer's Perspective*. Third Edition. Pearson, 2016.

[2] Brian W. Kernighan and Dennis Ritchie. *The C Programming Language*. Second Edition. Prentice Hall, 1988.

License

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