Princeton University
COS 217: Introduction to Programming Systems
Fall 2019 Midterm Exam Preparation

Topics

You are responsible for all material covered in lectures, precepts, assignments, and required readings. This is a non-exhaustive list of topics that were covered. Topics that are crossed out will not appear on the midterm exam but may appear on the final exam.

1. Number systems
    - Binary, octal, and hexadecimal
    - Finite unsigned integers, operations, and overflow
    - Finite two's complement signed integers, operations, and overflow
    - Floating-point numbers
2. C programming
    - From source to executable: preprocess, compile, assemble, link
    - Program structure: multi-file programs with header files
    - Process memory layout: text, stack, heap, rodata, ~~data, bss~~ sections
    - Primitive data types
    - Variable declarations and definitions
    - Variable scope, ~~linkage, and duration/extent~~
    - Constants: `#define`, constant variables, enumerations
    - Operators
    - Statements
    - Function declarations and definitions
    - Pointers and arrays
        - Call-by-reference, arrays as parameters, strings
        - Command-line arguments
    - Input/output facilities: `getchar()`, `fgetc()`, `putchar()`, `fputc()`, `gets()`, `fgets()`, `puts()`, `fputs()`, `scanf()`, `fscanf()`, `printf()`, `fprintf()`
    - Structures
    - Dynamic memory management
        - `malloc()`, `calloc()`, `realloc()`, `free()`
        - Common errors: dereference of dangling pointer, memory leak, double free
    - Abstract data types; opaque pointer types
    - Generic data structures and functions
        - Void pointers
        - Function pointers and function callbacks
    - *Parameterized macros and their dangers (see King Section 14.3)*
3. Programming in the large
    - Modules and interfaces
        - Abstract data types and ADT design in C
        - Heuristics for effective modules: encapsulates data, manages resources, is consistent, has a minimal interface, detects and handles/reports errors, establishes contracts, has strong cohesion, has weak coupling
    - Program and programming style
        - Bottom-up design, top-down design, least-risk design
    - Building

        - Motivation for `make`, `make` fundamentals, non-file targets, macros
    - Testing
        - External testing
        - Internal testing and assertions: validating parameters and return values, checking invariants, checking array subscripts, checking function values
        - Unit testing with scaffolds and stubs
        - Test coverage: statement, path, boundary
    - Debugging
        - General heuristics for debugging: understand error messages, think before writing, look for familiar bugs, divide and conquer, add more internal tests, display output, use a debugger, focus on recent changes
        - Heuristics for debugging dynamic memory management: look for common DMM bugs, diagnose seg faults using gdb, manually inspect `malloc()`, calls, comment-out `free()` calls, use Meminfo, use Valgrind
    - ~~Performance improvement~~
4. Tools and the GNU/Linux programming environment
    - Linux, `bash`, `emacs`, `gcc`, `gdb`, `make`, ~~`gprof`~~
5. Common algorithms and data structures
    - Finite-state automata
    - Linked lists
    - Hash tables: hashing algorithms, key ownership and defensive copies
6. Applications
    - De-commenting
    - String manipulation
    - Symbol tables
    - Dynamically expanding arrays

Readings

As specified by the course Schedule *web page...*

Required:

- *C Programming* (King): 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20.1, ~~22~~, ~~24.1~~
- *Computer Systems* (Bryant & O'Hallaron): 1
- ~~*ARM 64-bit Assembly Language* (Pyeatt with Ughetta) 1~~

Recommended:

- *Computer Systems* (Bryant & O'Hallaron): 2, 5
- *The Practice of Programming* (Kernighan & Pike): 1, 2, 4, 5, 6, 7, 8
- *Unix Tutorial for Beginners* (website)
- *GNU Emacs Tutorial* (website)
- *Linux Pocket Guide* (Barrett) pp. 166-179
- *Deterministic Finite Automaton* Wikipedia article (website)
- *GNU GDB Tutorial* (website)
- *GNU Make Tutorial* (website)
- ~~*GNU Gprof Tutorial* (website)~~