Princeton University
COS 217: Introduction to Programming Systems
Spring 2017 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
    - The binary and hexadecimal number systems
    - Finite representation of unsigned integers
        - Operations on unsigned integers
    - Finite representation of signed integers
        - Two's complement
        - Operations on signed integers

2. C Programming
    - The program preparation process: preprocess, compile, assemble, link
    - Program structure: multi-file programs using header files
    - Process memory layout: text, stack, heap, rodata, ~~data, bss~~ sections
    - 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()` and `free()`
        - Common errors: dereference of dangling pointer, memory leak, double free
    - Abstract objects
    - Abstract data types; opaque pointers
    - 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
    - Testing
        - External testing taxonomy: statement, path, boundary, stress
        - Internal testing techniques: validate parameters, check invariants, check function return values, change code temporarily, leave testing code intact
        - General testing strategies: automate the tests, test incrementally, let debugging drive testing (fault injection)
    - Building
        - Separate independent paths before link
        - Motivation for make, make fundamentals, macros, abbreviations, pattern rules

- Program and programming style
    - Bottom-up design, top-down design, least-risk design
- 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
- Data structures and algorithms
    - Linked lists
    - Hash tables: hashing algorithms, defensive copies, key ownership
- Modules and interfaces
    - Abstract data types, specifications, reasoning about client code, representation vs. abstraction, underspecified behavior, ADT modules in C
    - Module qualities: encapsulates data, is consistent, has a minimal interface, detects and handles/reports errors, establishes contracts, has strong cohesion, has weak coupling
- Performance Improvement
    - Case study: buzz
    - When to improve performance
    - Improving execution (time) efficiency: do timing studies, identify hot spots, use a better algorithm or data structure, enable compiler speed optimization, tune the code
    - ~~Improving memory (space) efficiency: use a smaller data type, compute instead of storing, enable compiler size optimization~~

4. Applications
    - De-commenting
    - Lexical analysis using finite state automata
    - String manipulation
    - Symbol tables, linked lists, hash tables
    - Dynamically expanding arrays

5. Tools: The Linux/GNU programming environment
    - Linux, `bash`, `emacs`, `gcc`, `gdb`, `make`, ~~`OProfile`~~

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
- *Computer Systems* (Bryant & O'Hallaron): 1

Recommended:

- *Computer Systems* (Bryant & O'Hallaron): 2, ~~5.1-5~~
- *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)
- ~~*OProfile Manual* (website)~~