Princeton University
COS 217: Introduction to Programming Systems
Spring 2008 Final 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 were covered after the midterm exam are in ***boldface****.*

1. C programming
   - The program preparation process
   - Memory layout: text, stack, heap, rodata, data, bss sections
   - Data types
   - Variable declarations and definitions
   - Variable scope, linkage, and duration/extent
   - Variables vs. values
   - Operators
   - Statements
   - Function declarations and definitions
   - Pointers
   - Call-by-value and call-by-reference
   - Arrays
   - Strings
   - Command-line arguments
   - Constants: #define, enumerations, the "const" keyword
   - Input/output functions
   - Text files
   - Structures
   - Dynamic memory management: malloc(), calloc(), realloc(), free()
   - Void pointers
   - Function pointers and function callbacks
   - Macros and their dangers (see King Section 14.3)
   - The assert() macro
   - Bitwise operators
   - **Unions**
   - **The fwrite() and fread() functions**

2. Programming style
   - Modularity, interfaces, implementations
   - Design by contract
   - Multi-file programs using header files
   - Protecting header files against accidental multiple inclusion
   - Opaque pointers
   - Stateless modules
   - **Abstract objects**

- Abstract data types
- Memory "ownership"
- **Checking invariants**
- Testing
- **Profiling and instrumentation**
- **Performance tuning**

3. Representations
    - The binary, octal, and hexadecimal number systems
    - Signed vs. unsigned integers
    - Binary arithmetic
    - Signed-magnitude, one's complement, and two's complement representation of negative integers

4. **IA-32 architecture and assembly language**
    - **General computer architecture**
        - **The Von Neumann architecture**
        - **Control unit vs. ALU**
        - **The memory hierarchy: registers vs. cache vs. memory vs. disk**
        - **Instruction pipelining**
        - **Little-endian vs. big-endian byte order**
        - **CISC vs. RISC**
        - **Language levels: high-level vs. assembly vs. machine**
    - **Assembly language**
        - **Directives (.section, .asciz, .long, etc.)**
        - **Mnemonics (movl, addl, call, etc.)**
        - **Instruction operands: immediate, register, memory**
        - **Memory addressing modes**
        - **The stack and local variables**
        - **The stack and function calls**
            - **The C function call convention**
    - **Machine language**
        - **Opcodes**
        - **The ModR/M byte**
        - **Immediate, register, memory, displacement operands**
    - **Assemblers**
        - **The forward reference problem**
        - **Pass 1: Create symbol table**
        - **Pass 2: Use symbol table to generate data section, rodata section, bss section, text section, relocation records**
    - **Linkers**
        - **Resolution: Fetch library code**
        - **Relocation: Use relocation records and symbol table to patch code**

5. **Operating systems**
    - **Services provided**
    - **Processes**
        - **The process lifecycle**
        - **Context switches**
        - **Virtual memory**
    - **Computer security**
        - **Buffer overrun attacks**
    - **UNIX processes**
        - **System calls: getpid(), execvp(), fork(), wait(), system()**
    - **UNIX I/O**
        - **The stream abstraction**
        - **System calls: open(), creat(), close(), read(), write(), dup(), dup2()**
        - **Buffering**
    - **UNIX inter-process communication**
        - **Pipes**
        - ~~**Sockets**~~
        - **System calls: pipe(), close(), dup(), dup2()**
    - **UNIX Signals**
        - **Sending signals**
            - **Via keystrokes**
            - **Via the kill command**
            - **Via the raise() and kill() functions**
        - **Installing signal handler functions**
            - **The signal() and sigaction() functions**
        - **Ignoring signals**
        - **Race conditions**
        - **Blocking signals**
            - **The sigprocmask() function**
    - ~~**UNIX alarms and timers**~~
        - ~~**The alarm() function**~~
        - ~~**The setitimer() function**~~

6. Applications
    - De-commenting
    - Lexical analysis via finite state automata
    - String manipulation
    - Symbol tables, linked lists, hash tables
    - Dynamically expanding arrays
    - XOR encryption
    - **Dynamic memory management**
        - **Optimizing free()**
        - **Optimizing malloc()**
    - **Shells**

7. ~~Tools: The UNIX/GNU programming environment~~
    - ~~UNIX, bash, xemacs, gcc, gdb,~~ ~~**gdb for assembly language, make, gprof**~~

Readings

As specified by the course "Schedule" Web page. Readings that were assigned after the midterm exam are in ***boldface****.*

Required:

- *C Programming* (King): 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19.1-3,**20**
  Alternate: *The C Programming Language* (Kernighan & Ritchie): 1,2,3,4,5,6,7,B1,B2,B3,B4,B5,B6,B11
- *The C Programming Language* (Kernighan & Ritchie): **8.7**
- *The Practice of Programming* (Kernighan & Pike): 1,2,4,5,6,**7,8**
- *Programming from the Ground Up* (Bartlett): **1,2,3,4,9,10,B,E,F**
  Alternate: *Computer Systems* (Bryant & O'Hallaron): **2,3**
- ***Communications of the ACM*** **"Detection and Prevention of Stack Buffer Overflow Attacks"**
- *The UNIX Programming Environment* (Kernighan & Pike): **7.4,7.5**

Recommended:

- *C Programming* (King): 19.4
- *Programming from the Ground Up* (Bartlett): **5,6,7,8,11,12,13,C**
  Alternate: *Computer Systems* (Bryant & O'Hallaron): 1,**5,7**
- *Programming with GNU Software* (Loukides & Oram): 1,2,3,4,6,**7**,**9**
- *The C Programming Language* (Kernighan & Ritchie): **8.1,8.2,8.3,B9**

Recommended, for reference only:

- ***Using as, the GNU Assembler***
- ***IA32 Intel Architecture Software Developer's Manual: Volume 1: Basic Architecture***
- ***IA32 Intel Architecture Software Developer's Manual: Volume 2: Instruction Set Reference***
- ***IA32 Intel Architecture Software Developer's Manual: Volume 3: System Programming Guide***
- ***Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification***

There is no need to bring the reference manuals to the exam.