Princeton University
COS 217: Introduction to Programming Systems
Fall 2005 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. New topics 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
Text files
Structures
Dynamic memory management (malloc(), calloc(), realloc(), free())
Dynamic memory management errors (dangling ptr., memory leak, multiple free)
Void pointers
Function pointers
The assert() macro
Unions and tagged unions
Bitwise operators
The fwrite() and fread() functions

2. Programming style

Modularity, interfaces, implementations
Programming 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"
- **Invariants**
- Testing strategies
- **Profiling and instrumentation**
- **Performance tuning**
- **Portable programming**

3. IA-32 architecture and assembly language

- **General computer architecture**
  - **Control unit vs. ALU**
  - **Registers vs. cache vs. memory vs. disk**
  - **Instruction pipelining**
  - **Little-endian vs. big-endian byte order**
  - **CISC vs. RISC**
- **Assembly language**
  - **Directives (.section, .asciz, .long, etc.)**
  - **Mnemonics (movl, addl, call, etc.)**
  - **Instruction operands: immediate, register, memory**
  - **The stack and local variables**
  - **The stack and function calls**
    - **The C function call convention**
- Number representation
  - The binary, octal, and hexadecimal number systems
  - Signed numbers: signed mag., one’s comp., two’s comp.
- **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**

4. Operating systems

- **Services provided**
- **Processes**
- **The process life-cycle**
- **Context switches**
- **Virtual memory**

- **System calls**
    - **open(), creat(), close(), read(), write(), the standard I/O library**
- **Computer security**
    - **Buffer overrun attacks**
- **Signals and alarms**
    - **Race conditions**
    - **Blocking signals**
    - **The kill command**
    - **The signal() function**
    - **The sigaction() function**
- **Alarms and timers**
    - **The alarm() function**
    - **The setitimer() function**

5. 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
    Execution profiling

6. 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. New readings are in ***boldface****.*

Required:

The C Programming Language (Kernighan & Ritchie): 1, 2, 3, 4, 5, 6, 7, **8.1, 8.2, 8.3, 8.7**, B1, B2, B3, B4, B5, B6, **B9**, B11

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**

Recommended:

Programming with GNU Software (Loukides & Oram): 1, 2, 3, 4, 6, **7**, **9**

Programming from the Ground Up **(Bartlett) 5, 6, 7, 8, 11, 12, 13, C**

Communications of the ACM **"Detection and Prevention of Stack Buffer Overflow Attacks" article**

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

Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification