Topics for Exam #1

CSE-381: Systems 2

As per the syllabus, Exam #1 for CSE-381: Systems 2 is scheduled

Thu, Oct 3 (Time: 7 PM to 9 PM)
In HUG-141 (Hughes Hall)

The topics enumerated below are a succinct list of major concepts that you are expected to know for Exam 1. It may not include all details covered in lectures and labs. Consequently, it is not an exhaustive list and you must use in-class exercises, lab exercises, and homework assignments in addition to these topics to fully prepare for the exam. Particularly this is the time to review the notes you have been diligently recording during labs and lectures.

The exam will be closed notes and closed books. A one-sheet handout on common methods and Linux commands will be supplied along with the exam -- see CommonMethodsAndCommands.pdf. No other reference materials or discussions will be permitted. Use of electronic equipment (other than those required for life support and have suitable doctor certification along with them) other than calculators (calculator must not have a qwerty keyboard or a graphical stylus input pad) is strictly prohibited.

Type of questions:
Same general format as most exams: Multiple-choice, fill in the blanks, short answer, "what does the following C/assembly code do", and "write the C/assembly code to do the following" types of questions. Specifically, you will be expected to read, comprehend, analyze, troubleshoot, and develop programs involving concepts covered in the course. Programs will involve suitable object oriented concepts and standard library concepts covered in the course.

Text coverage:

E-book titled "Operating System Concepts" -- Link in `Syllabus` page on Canvas (all students have free access to the electronic book):

- Chapter 1: Introduction
- Chapter 2: Operating System Structures
- Chapter 3: Processes

1. **Concepts pertaining to systems**
   i. The 3 key aspects that define a system (you must memorize the definition)
   ii. Open vs. closed systems
   iii. Common examples of systems
   iv. Concept of systems-of-systems
   v. Key components of a computing system (aka computer)
   vi. Terminology & expansion for acronyms: System, sub-system, CPU, ALU, Register, RAM, ROM, DBMS, File System, NIC, TCP, IP

2. **Functions of an operating system with examples**
   i. Types of OS: single user vs. multi-user, single tasking vs. multi-tasking
   ii. Batch processing vs. multiprocessing.
      1. Context switching -- preemptive vs. non-preemptive
   iii. Memory management, basics of virtual memory, pages and paging, mapping virtual memory to real memory.
   iv. Storage management. Basics of creating directories, files, and navigating directories using Linux shell commands.
   v. Device management (basic concepts only).
   vi. Security and Privacy:
      1. Security rings
      2. User-id, effective-user-id, group-id

3. **System calls**
   i. Basics of system calls
   ii. Three common approaches for implementing system calls
   iii. Tracing system calls using strace.
      a. First (`exec`) and last (`exit_group`) system calls observed via `strace`.

4. **Design of operating systems**
   i. Factors influencing design of OS
   ii. Design strategies for OS
      1. Monolithic,
      2. Microkernels,
      3. Layered,
      4. Hypervisors, Type-1 and Type-2 hypervisors
      5. Client-Server
   iii. Advantages & disadvantages of various OS design approaches
   iv. Identification of ideal OS design approach from scenario descriptions.

5. **Booting & Boot loaders**
   i. Concept of booting
   ii. Steps in booting
   iii. Boot loader concepts and design strategies (single vs. 2-stage booting)

6. Basics of processes

i. Concept of processes. Process vs. program
ii. Basic memory layout of processes (Text, Data, Heap, & Stack)
iii. Observing processes using `ps` and `top`
iv. Processes hierarchies and tracing hierarchies in Linux
v. Terminating processes using `kill`
vi. Process lifecycle: New→Ready→Running→Waiting→Terminating
    a. Scenarios when different transitions occur

7. Process creation

i. Creating processes using `fork` system call
    - Creating different hierarchies of processes
    - Understanding cloning of I/O streams
ii. Replacing existing program with `execvp` system call
    - Passing command-line arguments
    - Basic idea of `PATH` to locate programs
iii. Using `waitpid` to wait for process to complete
    - Using exit code (return value for main of child process)

8. Inter-Process Communication (IPC)

i. Motivation for IPC
ii. Shared memory vs. message passing
iii. Using pipes for IPC
iv. Named pipes/fifo
    - Creating FIFO (mkfifo) & setting permissions
    - Redirecting input/output at the shell
v. Using anonymous pipes at the shell
    - Creating ad hoc software pipelines via bash

9. Basics of File systems

i. Need for a file system
ii. Relative and absolute paths

10. Basics of Linux operations

i. Basics of process management and process hierarchies. Viewing process hierarchies in Linux. Starting and killing processes via Linux terminal.
ii. Creating and navigating directories via Linux terminal.
iii. Creating, copying, and deleting files using commands in a Linux terminal.
iv. Basics of compiling and running C/C++ program in Linux in a Linux terminal.
v. Process IDs, Parent process IDs, and Monitoring processes using `ps` command.
vi. Foreground vs. background processes in Linux.

C/C++ programming concepts:

11. Basic program constructs

i. Stages in compiling a C++ program.
ii. Variables & expressions
iii. Constant variables vs. literal constants
iv. Signed vs. unsigned data types
v. if and if-else statements
vi. switch statement
vii. Looping constructs (for, while, do-while, range-for)
viii. Basic mathematical problem solving concepts
- Deciding number is even/odd, positive/negative, factor/divisor/dividend/quotient
- Using division and modulo operations for basic number manipulation, e.g.: reverse a number with loops & math (without using string)
- Detecting if a number is prime.
- Identifying largest/smallest number in a set of inputs
- Finding average (i.e., mean) of a given set of numbers

ix. Functions/methods
- Pass by value versus pass by reference
    a. Preferred approach for primitive data types vs. objects
- Memory/copy impact of pass-by-value
- Using `const` keyword for parameters.

x. Default values for parameters

12. Basics of objects

i. Differences between primitive and object data types in C++
ii. Calling methods on objects (e.g.: `string::length`)
iii. Using `std::string`
- Constructors for string.
- String comparisons
- Methods for operating and accessing strings
- Conversion to-and-from numeric data types to std::string.
- Formatting strings into HTML, given HTML tags to use (you don't need to know HTML)

13. Arrays

i. Basics of old-style arrays.
ii. 1-D arrays
iii. 2-D arrays
iv. Command-line arguments
- Designing programs that use command-line arguments
- Figuring out what and how many command-line arguments a program ought to take.
- When to prefer command-line arguments instead of reading data from files/console.

14. Basics of Pointers

i. Concept of memory and address
ii. Basics of pointers to hold addresses
iii. Basic pointer operators
- Address of operator (&)
- Indirections/dereferencing a pointer (*)
- Using object dereference operator (->)

iv. Pointer arithmetic
v. Pointers ↔ array operation similarities and code conversion
vi. Understanding command-line arguments
- Array of pointers.

15. Vectors

i. Use of vectors instead of arrays for processing data
ii. Differences between vectors and arrays
iii. Defining and using vectors of different data types
iv. Using vectors in method definitions and method calls
v. Create type aliases via the using clause in C++
- Creating aliases given English description
- Tracing aliases back to their original types.

vi. Operations on a vector: adding elements, accessing elements, removing elements, etc.
vii. Reading/printing/writing vectors to I/O streams
viii. Vectors of user-defined classes

16. Hash maps (`unordered_map`)

i. Concept of `unordered_map`
ii. Using `unordered_map` as associative arrays
iii. Defining and using `unordered_maps` of different data types
iv. Looking-up values in `unordered_maps`
v. Iterating over all the entries in a map and processing them
vi. Reading/printing/writing vectors to I/O streams
vii. Maps of user-defined classes

17. Basic text file I/O operations

i. Reading and writing data to console using `std::cin` and `std::cout`.
ii. Using stream-insertion (<<) and stream-extraction (>>) operators to read and write data.
- Understanding these operators and how they handle whitespaces.

iii. Using `std::getline` method to read a full line of text
iv. Using `std::ifstream` and `std::ofstream` to read/write text files.
v. Using `std::istringstream` and `std::ostringstream` to perform I/O with strings.

18. Other exercises

i. Converting English statements to corresponding C++ statements
ii. Describing C++ statements in English
iii. Code walkthroughs to determine operation and output from a C++ program
iv. Developing a C++ program given a functional description
v. Identifying performance or memory issues in C++ programs
vi. Rewriting C++ program to address memory or performance issue

19. Linux commands and shell

i. Basic operations at the shell prompt
    a. Navigating directory structures
    b. Listing files
    c. Copying files -- **including using `scp`**
    d. Troubleshooting common problems given error message(s)
ii. Compiling and running programs
iii. Using pipes to create ad hoc software pipelines
    a. Redirection to create (>) or append (>>) to existing files
    b. Redirection to supply input from a file (<)
    c. Using pipe (|) to create software pipelines

Preparation Suggestions:

1. As a general note you should expect to repeat questions from lab exercises and homework.
2. You should know all the material in lecture slides.
3. Do read the E-book materials used in homework while paying attention to implementation/application details.
4. Redo lab exercises. Develop short programs to test/verify your understanding of concepts. Review developing classes and overloading operators. Review how to call overloaded operators. Review vectors, how to use vectors. Review `unordered_map` and how to use it.
5. Review homework solutions on Canvas.
6. Review the functionality of pertinent methods and commands in the supplied method/command sheet.
7. Review the handouts material and videos on Canvas.