CSC 230 : C and Software Tools

Credit-By-Examination Study Guide

Instructions

- The exam covers features of the C programming language as well as the usage of related software development tools in a GNU/Linux environment. You may be asked to write or explain the behavior of programs or program fragments that include the following program elements, techniques or language features. You may also be asked to use the tools listed below to accomplish specific tests.
- The exam targets the C99 version of the language.
- You may bring a 3x5 note card containing handwritten notes on the material (both sides). You may not use other resources like books, cell phones or calculators.

Topics

- General concepts
  - Models of programming, Interpretation vs compilation/native execution, virtual machines, procedural, object-oriented, imperative, declarative.
  - Stages of compilation (preprocessing, lexical analysis, …), source files, object files and executables
- Language Elements
  - Fundamental data types (int, short, char, double), signed vs unsigned, overflow and underflow
  - Conditionals and booleans, _Bool, stdbool.h
  - Variables, scope, storage class and initialization. Extern and linkage.
  - Expression and expression evaluation, conversion and type casts, arithmetic operators, logical operators and short circuiting, the ternary operator, precedence and associativity, side effects and sequence points.
  - Bitwise operators, manipulating individual bits groups of bits, hexadecimal and octal literals.
  - Flow of control statements (if, for, while, do/while, switch, break, continue, goto)
  - Function definitions and prototypes, parameter passing and return types/values, storage of local variables and recursion.
  - Arrays, array initialization, representation of one- and multi-dimensional arrays, array parameters and return values, array bounds and buffer overflow.
  - Struct and union, padding, bit fields, typedef, struct parameter and return values.
  - Enums
  - Pointers, pass by address, function pointers, const, pointer vs arrays, nesting of pointers, arrays, structs and unions.
  - Preprocessing, includes, macros, conditional compilation, side-effects and macros.
  - Dynamic memory allocation, memory leaks, using sizeof, void pointers, NULL, malloc()/free()/calloc()/realloc().
- Standard Library and Applications

    - Use of header (.h) and implementation (.c) files, include guards.
    - Standard I/O (stdio.h, streams, fopen()/fclose(), printf()/sprintf()/snprintf()/fprintf(), scanf()/sscanf()/fscanf(), getchar()/putchar()/fgetc()/fputc()/getc()/putc(), feof(), ungetc(), fflush(), fread()/fwrite())
    - Math (math.h, sin()/cos()/tan()/asin()/acos()/atan()/atan2(), pow()/sqrt(), round()/floor()/ceil())
    - Strings (character and string representation, string.h, strlen(), strcpy()/strncpy(), strcat()/strncat(), strcmp()/strncmp())
    - Utility functions (exit(), memset(), memmove(), memcpy(), qsort(), rand(), srand())
    - assert
    - Error reporting, errno, errno.h, perror()
    - Command-line arguments
    - Resizable arrays and linked lists in C
- Tools and Environment
    - Compile and run C program in a Linux environment
    - gcc (compiling, linking, options -Wall -std=c99 -c -o -g -E -D)
    - gprof (compiling for profiling, collecting and evaluating profiling results)
    - gcov (compiling for coverage, collecting and evaluating coverage results)
    - Optimization, compiler options, techniques (code motion, reduction in strength, ...)
    - make (concepts, defining and using makefiles)
    - gdb (running programs, examining their state, setting breakpoints, stepping through execution)
    - valgrind, options --tool=memcheck, --leak-check=full, --track-fds=yes, interpreting valgrind output
    - git (concepts, cloning, adding files, committing and pushing changes)