COMP2001/2401 Review

(Final exam: June 25, 2012, SA 416 19:00 PM to 22:00 PM)

Lecture 01 and 02 (Intro. to system programming):

1. system programming
2. Windows and Linux
3. main tools under Linux
   - shell
   - text editor
   - complier (gcc)
   - debugger (gdb)

4.First programming

Lecture 03 (Linux): (Test#1)

5. Linux directories and files, file ownership, permissions
   - three user categories(world, group,owner)

6. commands: (Assignment#1)

man, ls, chmod, mkdir, pwd, rm, rmdir, cp, mv, cat, more, |, *, grep, spell

Lecture 04 (C basics: data types) (Assignment#1) :

7. Objected-oriented programming and procedural programming
8. primitive data types
9. control structure: break, continue
10.scope
12.enumerated data types
13.const
14. expression: e.g
    - parenthetical expressions
    - assignment and compound assignment
    - unary expression
    - more...
15. operators
16. precedence
17.debugging stuffs including programming errors

Lecture 05 (Bits bytes) :

18. bit models
    - ..... (all models are important.)
    - 07, 7, 0x7 (base8, base 10, base 16)
19. bit operations

- AND, OR, NOT, and others
- bitmask
- shifting

20. memory map
21. bit and byte
22. bases: base 10 (decimal), base 2 (binary), base 8 (octal), base 16 (hexadecimal)

Lecture 06 (Advanced data types): (Assignment#2) (Test#1, 2)

23. array (index from 0 to size-1)
24. accessing data out of bounds
25. 1D, 2D, 3D
26. according memory map (Test#1)
27. strings (Test#1)
    - NULL character (0, '\0')
    - scanf/printf
    - strlen, strcmp, strcpy, strcat, sprintf
28. command line arguments
29. structures (Test#1, 2)
    - defining structures
    - scope
    - the usage in function and assignment
    - nested structures

Lecture 07 (Pointers) (Assignment#3) (Test# 2)

30. Symbols: * , &, ., ->
31. pointer arithmetic
32. memory management
33. parameter passing (pass by values and pass by reference)
34. pointers and arrays (difference and the usage in common)
35. pointers and structures
    e.g. access a structure through a pointer.
36. pointer
    - NULL pointer exception

Lecture 08 (Dynamic memory) (Assignment#3)

37. memory allocation
    - malloc, calloc
38. access that memory
    - pointers
    - using array notation
39. four areas of memory
    - function call stack

- heap

40. double pointers
40. deallocate (free)
42. memory leaks

Lecture 09 (Linked Lists) (Assignment#3)

43. arrays and linked lists (difference, advantages, disadvantages)
44. basic linked lists and advanced linked lists
    - components
    - implementation in codes
    - insert, delete (first, last, middle)
        - pointers, memory, free
    - access
    - traverse (iteratively, recursively)
    - all in codes

Lecture 10 (I/O) (Assignment#3)

45. streams, buffers (line, block, unbuffered), pipes (>, <, |)
46. file pointers (fseek, ftell)
47. I/Os
48. fopen(), fclose(), fprintf(), fscanf(), fwrite, fread, fflush,
49. stdin, stdout
50. files, permissions

Lecture 11 (Programming building) (Assignment#3)

51. source code/object code/executable
52. preprocessing, compiling, linking
    - what is it?
    - how to?
53. Makefiles
54. header files

Lecture 12 (code organization) (Assignment#3)

55. function design (modularity)
56. variable scopes
57. data types, modifiers, qualifiers, storage class
58. multiple files
59. comments, indentations, variable names
60. use of preprocessing
    - typedefs

Lecture 13 (System Calls)

61. libraries calls and system calls (difference, advantage, disadvantage)
62. families of operations:
    - memory system calls
    - time system calls
    - file system calls
    - process system calls
    - signal system calls
    - socket system calls
65. process system calls
    - process (ps, characteristics, parent, child, shell commands)
    - fork(), execl(), execv(), wait(), waitpid(), system(),
66. signal system calls
    - inter-process communications
    - signal, signal handler
    - install, restore, ignore, send
    - kill
67. socket system calls (might be very useful for COMP3004)
    - client-server model (IPC)
    - steps in setting up socket communications
    - IP address and port number (ifconfig, netstat)
    - socket functions: socket(), bind(), listen(), connect(),accept(), recv(),send(), close()
    - one client process or multiple client processes

Lecture 14 (Libraries)

68. link in functions from an external library (.a files)
69. C standard library, X library, Curses library (basic, buffering, echoing, blocking), Xwindow,
70. make a library

Instructions for being prepared for your final exam:

1. Read and understand all of Lecture notes and examples in class;
   - especially, you could test yourself based on learning objectives and recaps in each lecture
2. Read your text book to help you understand concepts and examples;
3.Go through your assignments and your tests again to clarify your mistakes;
4.Practice example codes and run on your machine if you are confused with some concepts and codes;
5. Be prepared!
6. Good luck.

Final Exams:

- 3 HOURS
- 43 QUESTIONS (MULTIPLE CHOICES-- SINGLE ANSWER) [40 MARKS]
    - scantron (you have to know how to mark the answer)
    - mark your answer accurately;
    - do not make technical mistakes (such as misaligning your answers), which happens a lot.
- I strongly suggest you write down your answer on the exam paper also because your exam is going to return to us.
- There is a bonus question as the 44TH questions. [5 MARKS]
    - coding (not a multiple choice)
    - show your work as possible as you can.
    - steps take points.
- 10 pages not including the cover page.
    - Make sure you have all of pages before you answer your questions.
- No electronic device allowed.
- Bring your Student Card, pencils, erasers, pen...
- Follow exam rules.
- There are some simple questions and hard questions. Do not spend too long on a question.
    - Go through all of questions quickly when you get the exam;
    - Find out the simple ones and answer them correctly;
    - Then work on the harder ones;
    - Do not give up any question.
    - Do not think of a question too simple or too hard and try to rethink it. There might be some tricks you may forget.
- Every question is assumed in C under the Linux system.
- ONLY ONE ANSWER TO EACH QUESTION.

Question Samples

1. Given the following code,

```
unsigned char c = 20;
char d = 200;
printf("%d   %d\n", c, d);
```

what's the output:

(a) -20 56
(b) 20 56
(c) 20 -56
(d)-20 -56
(e) none of the above

2. Choose an accurate statement with regarding to the following code:

```
int *AssignArray(int element1, int element2)
{
    int myArray[2];
    myArray [0] = element1;
    myArray [1] = element1;
    return myArray;
}
```

a) This code is perfect with no faults.
b) This code is not correct because element1 and element1 should be pointers using pass by reference.
c) This code does not compile because we should dereference the stack allocation pointer.
d) This code is bad because it returns an address to unassigned memory on the stack. Yet it compiles.
e) none of the above

Choose A for TRUE and B for FALSE

4. C is a typical object-oriented programming language.
5. In C, it is ok not to deallocate your memory at the end of your program because your program will release it in the end.

6. In C, precompiler instructions are preceded by the # symbol. To use functions from a library named “library.h”, one needs to use the following instruction

(a) #import <library.h>
(b) #using <library.h>
(c) #include <library.h>
(d)#define <library.h>
(e) none of the above