352 Final Exam Study Guide

Ultimately, the topics for the final exam includes anything from class, videos, the readings, the PAs, and any other material covered before the exam. This list is just provided as a guide.

Topics

- General UNIX, Linux, bash knowledge
- How to use various bash commands (ls, cd, pwd, cut, grep, cal, echo, printf, sed, etc)
- Standard input, output, error
- Piping and redirection
- GCC and some of the command line arguments used in class (-Wall, -Werror, etc)
- C programming
    - This includes everything covered about C in this course. Language basics, types, functions, I/O, pointers, memory management, l-values and r-values, etc, etc
- Standard library functions
    - You don't have to know every standard library function in the whole library, but you should be familiar with ones used in class and homework assignments. For example, ones from `stdio.h` such as printf, fprintf, fopen, fclose, etc.
- Basics about C program layout and the stack
- How to use GDB
- General UNIX file system knowledge
- Navigating a file system via a terminal such as bash
- UNIX files, inodes, inode structure
- File permissions
- File Systems
- Dynamic memory allocation
- Stack and heap
- malloc, calloc, free, etc
- Managing memory
- Valgrind
- Data structures in C (linked lists, trees, graphs, dynamically-sized arrays, etc)
- Make
- Processes, process control, top command, kill command
- Regular expressions, grep, sed
- Shell scripting
- The steps of C compilation
- C preprocessor
- Binary IO and bit manipulation
- Object files, executables, ELF, linking
- Function pointers
- The kernel and system calls
- fork, exec, pipes, waitpid, etc

Problems

I am not going to be including solutions for these problems. In order to check if your solutions are correct you can try:

1. Analyzing your solutions yourself after writing
2. Comparing with other students
3. For coding problems: Run and test the code!

For some of the coding questions, you can try implementing two versions - one with the standard library functions allowed (which could make it shorter) and another version with no standard library. You should not rely on only this study guide for final exam preparation.

Problem 1

In this problem, you should write a C function named **`build_multiplication_table`**. This function should have a single parameter of type **`int`** that represents the largest number to include in the multiplication table. You should expect that the argument is a positive integer. This function will return an **`int**`** which is an array of arrays of pointers. The first row/column of the table represent the numbers that will be multiplied to get a result. You should allocate heap memory to store all of the results. For example, if the function was passed the value **4,** a pointer to the following table should be returned:

```
[[0, 1, 2,  3, 4],
 [1, 1, 2,  3, 4],
 [2, 2, 4,  6, 8],
 [3, 3, 6,  9, 12],
 [4, 4, 8, 12, 16]]
```

Problem 2

In this problem, you should write a bash file named **`search.sh`**. This script will take two command line arguments. The first argument is the name/path to a file and the second is a term to search for within the file. This script will print out the lines of the file with a couple key details:

1) if the line contains the search term, a line of 10 dashes (-) should be printed before and after the line. The lines of dashes shouldn't have line numbers
2) Line numbers should be printed along with the line itself

Example Input:

```
in.txt
Hello
World
Foo
Bar
Baz
Foobar
```

```
bash seach.sh in.txt Foo > out.txt
```

```
out.txt
1 Hello
2 World
----------
3 Foo
----------
4 Bar
5 Baz
----------
6 Foobar
----------
```

Problem 3

In this problem, you should write a C function named **convert**. This function should have one parameter of type **char***. You can expect that this parameter will be a string containing only upper and lower case letters. The function should convert any upper-case letters to The character **'0'** and lower-case letters to **'1'** in the string that was passed in. The function should not return anything (a **void** function).

Problem 4

In this problem, you should write a C function named **count_ones**. This function should have one parameter of type **unsigned long**. The function should count how many 1 digits exist in the binary representation of that **unsigned long** number in memory and return the count as an **int**. For example, say that on a test system, an **unsigned long** was 64 bits, and the **unsigned long** with the value 25 was passed in. The underlying binary of this would be:

```
0000000000000000000000000000000000000000000000000000000000011001
```

The function should return 3 in this case.

Problem 5

In this problem, you should write a C function named **`max_cross`**. This function should have two parameters: The first of type **`int**`** which you can expect to be a pointer to an array of pointers, which each point to the beginning of an int array. (In this question, I'll refer to this as a 2D array, though technically it is not one). You can expect the second parameter to be an **int** representing the number of rows and columns (you can assume that the 2D array will have an equal number of rows / columns, and every row will have the same number of elements). This function should iterate through every element in the 2D array. For each element, it should sum up all the values in that particular row and column. The function should return the highest of these sums. For example, say that this 3x3 2D array was passed in:

```
10, 20, 30
10, 20, 50
50, 15, 40
```

The function should return 185 in this case. This is because the max row/column sum combination is:

```
10, 20, 30
10, 20, 50
50, 15, 40
```

Problem 6

In this problem, you should write a C function named **`binary_tree_to_binary_file`**. This function should have two parameters. The first should be a **`NumTreeNode*`** representing the root of a binary tree with **`unit32_t`** values in each node. The second parameter should ba a **`char*`** representing the name of a file to write to. A **`NumTreeNode`** is defined as:

```
typedef struct NumTreeNode {
  uint32_t value;
  struct NumTreeNode* left;
  struct NumTreeNode* right;
} NumTreeNode;
```

The function should iterate through the tree via in-order traversal, and write each of the **`value`** values to the file specified by the second parameter. It should write each number in its binary representation, one after the other.

Problem 7

In this problem, you should write a C function named **`reverse_words`**. This function should have one parameter of type **`char*`** which you can expect will be a string with 0 or more words separated by spaces. The function should reverse each individual word within the string. For example, this code:

```
char words[] = "cheetah elephant cat";
reverse_words(words);
printf("%s\n", words);
```

Should print:

```
hateehc tnahpele tac
```

Problem 8

In this problem, you should write a C function named **`multiply_matrices.`** This function will calculate and return the dot product of the first and second square matrices. You should allocate enough heap memory to hold all values of the resulting product of the matrices. This function will have 3 parameters. The first two parameters are **`int**`**s that are pointers to arrays of pointers (similar to Problem 5). These two parameters represent the two matrices to multiply. The third is an **`int`** representing the number of rows and columns in the matrices. If you are unfamiliar with matrix multiplication, here is a resource to help you: How to Multiply Matrices

If the function is passed these two **2x2** matrices as parameters:

```
[[1, 2],       [[5, 6],
 [3, 4]]        [7, 8]]
```

A pointer to the following matrix should be returned from the function:

```
[[19, 22],
 [43, 50]]
```

Problem 9

ELF is the standard executable file format for UNIX systems. ELF files contain a number of sections. What is the purpose of the following three sections? Provide a 1-2 sentence description for each.

- `.text`
- `.data`
- `.rodata`

Problem 10

Write a bash command to print out a text representation of all of the **`pop`** assembly instructions from an ELF file named **`runme`**.

Problem 11

In this problem, you should write a short bash script. The script should list all files that end in **`.o`** from the **/lib** directory on a UNIX system, only if the file name (not including the extension) begins and ends with a vowel.

Problem 12

In this problem, you should write a bash script to complete the following task.

The bash script should have one command line argument, that being a path to some directory on the computer. The script should find every file that ends in .c or .h within that directory, and count how many lines are in each. It should print out the name of each file with the number of lines in the files.

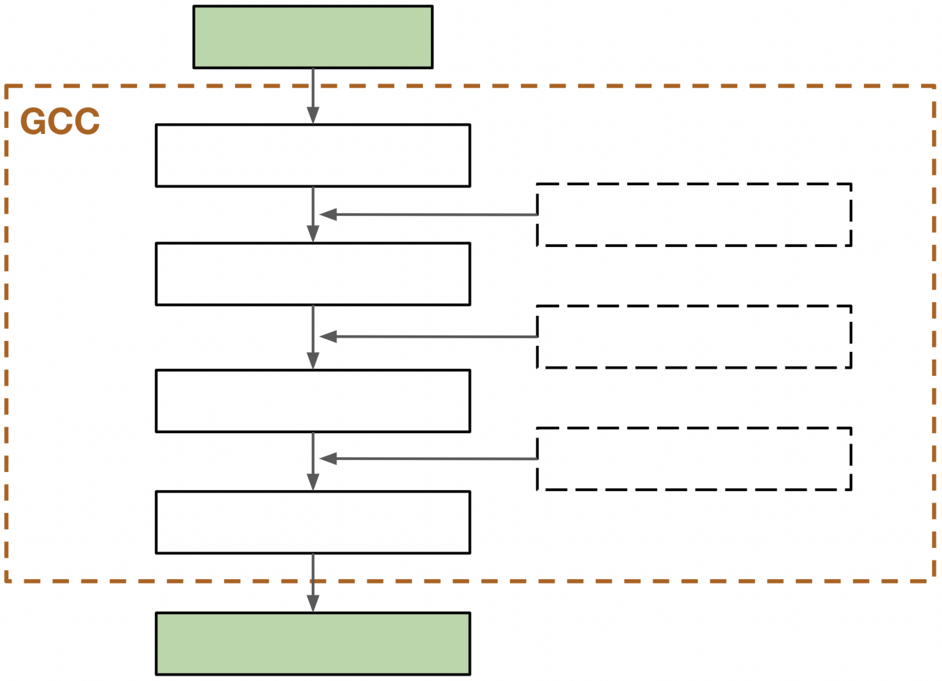
Problem 13

In this problem, you should write a bash script to complete the following task.

The bash script should have one line argument, which you can expect will be the name of a text file. The script should determine how many lines of the file begin with the letter **`a, b, c . . ., y, z`**. For each, it should print out the starting letter, accompanied by how many lines start with that.

Problem 14

Fill in the blanks of the diagram showing the steps of compilation with GCC:



Problem 15

Match the name of the command to the best-fitting description:

sed	displays the current user
ls	show running processes
elfdump	search for and optionally replace elements in text
objdump	show processes and what resources those processes are consuming
whoami	search through text
grep	display files
du	show information about ELF executables, including disassembly
ps	show information about the sections in an ELF file
top	display sizes of files

Problem 16

What is the difference between the **fork** and **exec** system calls? Why are these two system calls often used together? Give a clear example.

Problem 17

If we have a struct named Student as defined below, what is the minimum number of bytes that would be allocated when calling malloc(sizeof(Student)); on lectura.

```
typedef struct Student {
    char* name;
    uint8_t id;
    float gpa;
} Student;
```

Problem 18

Given the following main function, write out the function that thing points to.

```
int main(void) {
    int (*thing)(int) = &doubler;
    printf("%d\n", (*thing)(1));
    printf("%d\n", (*thing)(5));
    printf("%d\n", (*thing)(10));
    return 0;
}
```

This outputs:

```
2
10
20
```

Problem 19

Assume you're given a program named testprog which accepts input from standard in. You can also assume that there's a text file named input.txt which contains a sample set of inputs.

What is the command to run testprog, feeding it the inputs from input.txt on stdin, and checking for memory leaks all in one command?

Problem 20

Write a function that has a char * as an argument which represents the path to a file on the system and uses a system call to print out the modification time of that file. You will likely want to include time.h and sys/stat.h to do this.

Problem 21

Assume we want to anonymize some data in a csv file. Write a command using sed which replaces all occurrences of street addresses in a file with the string ####.

Street addresses will always take the form: “1234 E Streetname”
Where the the number can be any length
The direction will always be a single uppercase letter from {N, S, E, W}
And the street name will be a single word without spaces of unknown length.

Problem 22

True or False?

- The C language automatically initializes variables to safe values.
- Valgrind will show you more detailed error locations when you compile a library with -g
- Everytime you run your program it will have the same process ID
- When you call fork() the new process is a child of the caller.

Problem 23

Refer to the following program and give the minimum number of bytes which will be allocated when `malloc(sizeof(Jet));` is called.

```
typedef struct Jet
{
     int power;
     float max_ altitude;
     uint8_t color;
}Jet;
```

Problem 24

Name the system call/library suitable for the following description:

- To create a copy of the currently running process.
- To load an ELF file and execute it.

Problem 25

Each line in the file words.txt only includes one word. Count the number of six- or seven-character words in this file that begin with "p" or "b" and end with "ing."

Problem 26

Three files, main.c, aux.c, and btree.c, as well as a universal header file b.h, make up a C program. The executable of the software is known as btree. Create Makefile rules for the program's modular compilation and linking for the following targets: btree, main.o, aux.o, and btree.o.

Problem 27

Create a C function called frontPush that accepts a linked list head pointer as an argument. Assume that the linked list is sorted by points in decreasing order, beginning with the second entry in the list (from high to low). In order to sort the entire list by points in decreasing order, the function transfers the initial element to the proper location in the list. The resultant list's head pointer is returned.
The function prototype is:

```
struct node* frontPush(struct node* head);
```

Problem 28

Finish the function stripStars which should return a pointer to a string that is str with the asterisks ('*') stripped off both ends. For example, the output of the program below should be:

```
Danger
```

You only need to write the missing code and it can be done in 4 lines of code. (A curly brace on a line by itself is not a line of code). NOTE: Your function is allowed to change the contents of the string pointed at by str.

```
#include <stdio.h>
#include <string.h>
char *stripStars(char *str) {
    char *ptr;
    /* your code here */


    return ptr;
}

int main() {
    char str[] = "***Danger***";
    printf("%s\n", stripStars(str));
    return 0;
}
```

Problem 29

The following function returns a character array (string) with all of the vowels removed. Unfortunately I am having trouble using it in my code, specifically it is causing a memory error. Describe and fix any issues with the function that you see (there is supposed to be one). You don't have to rewrite the entire function, just the part that needs to be changed. You are guaranteed that any string passed in only contains alphabetic characters and there won't be more than 30 characters. You are also guaranteed there are no issues with the isVowel function, that works perfectly.

```
char *removeVowels(char *str) {
  char buff[31];
  int buffIndex = 0;
  for ( ; *str; str++) {
    if (!isVowel(*str)) {
      buff[buffIndex++] = *str;
    }
  }
  buff[buffIndex] = '\0';
  return buff;
}
```

Problem 30

What's the output of this program?

```
#define DEBUG 1
#ifdef DEBUG
    printf("This is the test version");
#else
    printf("This is the production version!")
#endif
```

Problem 31

Why do C programmers start header files with lines like:

```
#ifndef UTILS_H
#define UTILS_H
```

(choose one)
a. so that other programmers will know the file name
b. to avoid including the contents of the header file more than once
c. to guarantee that at least one variable is defined
d. so the compiler will know what file to include
e. because the program will not compile without it

Problem 32

The following macro function computes the maximum of two numbers:

```
#define max(x,y) ((x>y)?x:y)
```

And here is a regular C function that computes the maximum of two numbers:

```
int max(int x, int y) {return (x>y)?x:y;}
```

Is the run time of the two functions the same or is one of them faster? What is the reason for that?
Tip: Think about creation and destruction of a stack frame for a function call.

Problem 33

The file shown below is saved as cmdlineargs:

```
#!/bin/bash
# File: cmdlineargs
# illustrates: command line argument handling
Args=$@
N=1
for X in ${Args} ; do
    echo "ArgNo ${N}: ${X}"
    ((N++))
done
```

a. What is the output if the following is entered?

```
cmdlineargs I am a test argument
```

b. What does the first line in the script do and why do we need it?

c. If I wanted to print the 2nd argument, can I do it by using the following commands?

```
i=2
echo $i
```

Problem 34

When it comes to processes and the CPU scheduler, what is a context switch? Why do context switches happen?

Problem 35

Write a grep command to search a text file for all string that begin with X and end with Y that can contain any number of digits in-between.

Problem 36

Write a regular expression that would match and not match the below strings:

Match: antelope artichoke art-gallerie azzzzzzzzzzze
Don't match: ate arte canyon snapshot

Problem 37

Write a regular expression that would match and not match the below strings:

Match: “100 chairs” “300 lemurs”
Don’t match: “50 cups” “1250 cars”

Created by: Benjamin Dicken
with additional problems by Michael Stark, Michael Arroyo, Kartikey Shukla, Mahdi Rahimi