CMPSC 311 Exam 1

February 14, 2014

Closed book, closed neighbor, no electronic tools or additional papers.

True/False

1. One point each. Circle T or F.

   T   F   a. "Each program should do only one thing and do it well" is a tenet of the Unix philosophy.

   T   F   b. It is a syntax error to add an integer and a pointer together.

   T   F   c. When you first start Vim, it is in Insert mode.

   T   F   d. The `PATH` environment variable tells the shell where to look for executable files.

   T   F   e. Unions are an example of weak typing in C.

   T   F   f. The API provided by the operating system kernel is called the shell.

   T   F   g. The expression `ptr->x` is equivalent to `*(ptr.x)`.

   T   F   h. A common reason for poor caching performance is that a program has too much locality.

   T   F   i. On a 64-bit architecture, `sizeof(char *)` is equal to 8.

   T   F   j. Unix is primarily intended as a single-user operating system.

   T   F   k. The dereference operator (`*`) can only be used on pointers and equivalent types.

   T   F   l. Functions which take an array as a parameter can determine the number of elements by using `sizeof`.

   T   F   m. The statement `int* x, y;` declares two pointers.

   T   F   n. In a Makefile, "`$@`" means the current target.

   T   F   o. The filename "." always refers to the current working directory.

   T   F   p. All of the following are common Unix commands: `ls`, `mv`, `cp`, `cd`, `mkdir`, `man`.

   T   F   q. When a userspace process makes a system call, the system call will be executed in user mode.

   T   F   r. Memory allocated on the stack is automatically freed when the function returns.

   T   F   s. The size of a struct is the size of all of its members added together.

   T   F   t. The size of a union is the size of its largest member.

Short Answer

Five points each. Short responses (3 sentences or less) and/or simple diagrams.

2. Write the decimal number 214 in both hexadecimal and binary. Show your work.

3. Imagine a compiler called `xcc` which reads a source code file from standard input (stdin) and writes the object file to standard output (stdout). Write a shell command which uses redirection or piping to compile `heart.c` into an appropriately named object file using `xcc`.

4. Write a two-line executable shell script that prints its first argument, the characters `<3`, and then its second argument. Be sure to use appropriate quoting. For example:

```
$ ./script.sh Cory Topanga
Cory <3 Topanga
```

5. Finish the implementation of this swap function in C:

```
void swap_gifts(int *gift1, int *gift2) {




}
```

Make sure you don't end up with the same gift you originally brought to the function!

6. C does not use exceptions. What is commonly used instead (in the standard library, for example) to indicate whether a function has succeeded or failed? What represents success, and what represents failure?

7. Consider the following code:

```
double date = 2.14;

if (date / 2 == 1) {
  printf("Equal\n");
} else {
  printf("Not equal\n");
}
```

What does this code print? What happened behind the scenes? What could the programmer do to make his/her intentions more clear?

8. Name two top-level directories in the traditional Unix filesystem and what kind of files are stored in each.

9. Based on the code snippet, finish the function prototype for the `chocolates` function, clearly showing the type of each argument.

```
void chocolates(                                        );

...

int x = *y;
chocolates(&x, &y, y);
```

10. Explain the Vim command `d5h`: what does it do, and why? (Remember that `h` moves the cursor one character to the left.)

11. You are selling flowers! Write code which creates a struct type called `flower_order` to keep track of the number of flowers in an order (an integer), the total price (a float), and the name of the person to deliver to (64 characters should be enough for anybody).

Long Answer

Ten points each. Responses should be thorough (address all parts of the prompt) but stay focused.

12. A C compiler will generate machine instructions for the "arithmetic" statement

    ```
    x = x + 1;
    ```

    in one of three ways, based on the type of the variable x. What are these three options, and how do they differ?

13. Name three of the major stages in the build process of a C program, from source code to an executing process. (Example that you may use: linking.) For each of these three stages, state the type of files used as input and the file or abstraction which is output. In addition, if the stage can include a certain type of library as input, you should mention that.

    [*Bonus:* do five stages instead of three.]

14. Consider the following Makefile rules:

```
valentine: candy.lc

candy.yc: candy.y
        mix -o $@ $^

candy.lc: candy.l candy.yc
        cook $^ > $@

%: %.c
        gcc -o $@ $^
```

Draw a dependency diagram for the build process that happens when you run `make valentine` (assuming that there exists a file called `valentine.c`). If you then make changes to the file `candy.l`, which other file or files must be re-built?