This homework is due in class on Monday, September 26. You can discuss the questions with other people in the class, but you should write up your own answers. Some of the questions do not have definite, unambiguous answers. In all cases, justify and explain your answers!

1. (Based on Exercises 2.13 and 2.14 from the textbook.) Suppose that you have to implement an operating system on hardware that supports interrupts and processor exceptions but does not have a trap instruction (for implementing syscalls). Can you devise a satisfactory substitute for traps using interrupts and/or exceptions? If so, explain how. If not, explain why.

Now, suppose that the hardware supports exceptions and traps but not interrupts. Can you Can you devise a satisfactory substitute for interrupts using traps and/or exceptions? If so, explain how. If not, explain why.

2. (Based on Exercise 3.3 from the textbook.) Explain what each of the following programs does when it is run on UNIX. (What's the difference between the behavior of the two programs?)

```
int main() {
    while (fork() >= 0) {
    }
}
```

```
int main() {
    while (fork() > 0) {
    }
}
```

3. (Exercise 3.9 from the textbook.) Consider the following program:

```
int main(int argc, char **argv) {
    int child = fork();
    int x = 5;
    if (child == 0) {
        x += 5;
    } else {
        child = fork();
        x+= 10;
        if (child) {
            x += 5;
        }
    }
}
```

How many different copies of the variable x are there? What are their values when their processes finish?

4. In Lab 2, you worked with memory allocation of variable-sized blocks of memory. How would things be different if all memory allocations have to be of the same size (say 4 kilobytes, for example)? How might you implement memory allocation in that case? What data structures would you use? Write a substantial essay to respond to these questions. (There's more than one way to do it! It would be nice to discuss more than one possibility.)