~~How to Write a Design Document~~ "What I See in the Design Documents of Students Who Also Do Well in this Course"

CS161 Course Staff
cs161@eecs.harvard.edu

February 9, 2014

1 A Table of Contents

1. Data Structures
2. Bootstrapping/Initialization
3. File Syscalls
4. Fork/Exec
5. Whatever
6. Stuff

Break down your design into parts. The parts of your design document should reflect the parts of the assignment. Do not write monolithic descriptions of your entire system, or you will end up writing a monolithic system.

2 Data Structures

"These are the fields we're adding to the Process struct:"

```
struct Proc {
  CV isDead; // Or a semaphore, or whatever you need for waitpid / etc.
  struct FileTable fT;
  ...
}
```

"This is the format of a file handle:"

```
struct FileHandle {
lock_t baz // protects access to biff
...
}
```

You can put this at the beginning, or the end, or break it up on a section-by-section basis. Whatever you think is most clear. But don't write about your data structures in broad terms; make them explicit!

In particular, identify the synchronization structures in your design, and what they are for/how and where they are used.

3 Real Pseudocode

This is not pseudocode for fork:

```
def Fork:
  //Create a new child process of the caller (parent) process.
  //Give the child a copy if its parent's address space and file handles.
  //If we are the parent, return the child's PID.
  //If we are the child, return 0.
```

This is just a description of fork!

I want to know how your system will implement fork.

1. This is how I create a new process. This is how I make it a child of the caller process.
2. This is how I copy its address space and file handles.
3. Fork is (sys)called once, but returns to userland twice. So how do I return to userland twice with only one trap frame? This is how.
4. Fork involves two threads. Do they share data? Does this require synchronization? (MAYBE. It depends on your implementation!)
5. If any of the above steps fails, then I (clean up and return an error/kernel panic/give up and go play DOTA2 or whatever kids think is cool these days).

I have probably forgotten some steps. You will probably not solve every step. But these are the kinds of questions you should try to ask and answer.

If you cannot figure out how to do something, then you'll know to ask for help! If you miss some subtlety, then your TF can point it out to you. But if you simply repeat back to us what your code is SUPPOSED to do in your own words, then all we can confirm is that you understand the requirements of the assignment. Which is good! But not as good as knowing how to do the assignment.

3.1 Function Definitions

- It is good to identify the functions you need to modify for each part of your design.
- It is good to identify some helper functions you think will be worth implementing.
- It is bad to paste huge blocks of real C code complete with error handling, etc. into your document. We are unlikely to find bugs in it, and it will crowd out the conceptual stuff, which is generally more challenging.

4 Effort Proportional to Difficulty

Fork is strictly harder than opening a file. So if you spend the same amount of time and space explaining them, then either you are underestimating fork or padding open file. You do not get bonus points for length; actually, it is much harder to give good feedback on a lengthy design document.

4.1 A Realistic Timeline

You are required to submit a timeline. Doubtless you will fail this timeline, due to stuff happening. The point of the timeline is to show us you have realistic expectations of how long each part of the assignment will take. If you do something like budget only one day for debugging, then we can warn you in advance that's not going to work. Similarly, if you want to divide up the work between partners, we can warn you if the division is unfair.

5 A Note on Partner Dynamics

This applies to life after this course.

Teams are always assymetric, always. You should try to work together when you can, and divide work evenly when you can't. But in practice, some part of the assignment will take more time than expected, or will be harder than expected, or someone might be unable to work in the second week due to other projects, or whatever. There will be assymetries, so you need to be flexible, and if your partner needs help, then help.

If you feel there's a problem with your teamwork, then please talk to the staff. Do not work independently from your partner until the day before the project is due, and then act surprised when "their part" of the assignment doesn't work with yours.

And do not go off and implement the whole assignment on your own, because you feel (your partner is slacking/your code is better/whatever). This will not work, and will make you both miserable. If you are at the point where you would consider it, please talk to the course staff.