HW 5: Map Reduce

MapReduce is a programming model for scalable and highly parallelized data processing. It abstracts away the complexities of developing a fault tolerant distributed system by exposing a simple API where the user specifies two functions:

- *map*: produces a set of key/value pairs from the input data
- *reduce*: combines values corresponding to the same key

With these functions, tasks can be automatically parallelized and executed on a cluster. This paradigm is particularly powerful since it allows programmers with little background in distributed systems to write parallelizable code for a wide variety of real world tasks.

In this assignment, which is loosely based on a lab from MIT, you will be implementing your own fault tolerant MapReduce system in C. Specifically, you will be implementing a coordinator process that distributes tasks to worker processes that carry out the actual computation. You will also handle worker failure by implementing task redistribution. The system design you will be building is similar to that outlined in the MapReduce paper.

> **NOTE**
> In this assignment, we use the term "coordinator" rather than "master".

> **NOTE**
> The purpose of this assignment is to teach you about distributed algorithms, so we will not be too strict on memory management (i.e. we will not test that you `free` all of the data you allocate). Of course, poor memory management that causes segmentation faults or other major issues will cause you to fail tests, but do not spend a lot of time worrying about memory leaks that do not impact the functionality of your code.

Components

This assignment is split up into three parts with distinct deadlines to help you space out the workload. Deadlines can be found on Ed.

RPC lab

To help you get a basic idea of how the coordinator will communicate with workers, we have put together a lab that walks you through how `rpcgen` works. `rpcgen` is a protocol compiler that helps with writing Remote Procedure Call (RPC) applications in C. This lab is worth 5% of your grade on the entire assignment. The code you write for the lab will be in a separate directory (`lab-rpc`) from the rest of the HW Map Reduce code (`hw-map-reduce`). To trigger the autograder for the lab portion, you must push the code in the `lab-rpc` directory. **There are no extensions for the lab portion of this assignment.**

Checkpoint

You will be expected to complete the first part of the assignment (up to and including the Tasks section) by an earlier deadline. This checkpoint is worth 5% of your grade on the entire assignment. **Note that for the checkpoint, you have to manually trigger the autograder build.** If you miss the checkpoint, you may get up to a 95% on the assignment if you pass all the tests by the final deadline, assuming that you have completed the lab in time. If you additionally do not turn in the lab in time, you may get up to a 90% on the assignment. **There are no extensions for the checkpoint portion of this assignment.**

Final

The final component consists of the rest of the tasks (through Fault tolerance).

Getting started

To get started, log in to your development environment and get the starter code.

```
cd ~/code/personal
git pull staff main
cd hw-map-reduce
```

If you made changes to your repo on a different (virtual) machine, make sure to run `git pull personal main` as well.
