Test 1

CPSC464, Spring2023

Assigned: Thursday, February 16, 2023
Due: Saturday, March 4, 2023, by 11:59pm

Part 1: Network Daemon

Write a C++ daemon that listens for requests. Requests may be one of two: (1) initiate parallel daemons, or (2) run a program at a specific port, say D. When a request for initiating N parallel daemons is received, the daemon should contact N computers running the daemon and send them a request to run a program (specified in the request to initiate parallel daemons) on these N computers, and go back to listening on port D. When a request to run a program is received, the daemon should fork a child process to run the requested program (with its arguments) and go back to listening on port D. The daemon can lookup a file called, say `nodes.txt`, that contains the IP addresses of the computers running the daemon. (You need to create this file after starting daemons on multiple computers.)

Part 2: Parallel Mergesort

Write a C++ program that acts as a master process or a slave process. The master process gets as its command line argument the number of processes, N, to be used. The slave process gets as its command line arguments the port number, say P, for the master process, the slave processes ID (1, 2, etc.) and the prefix of the name of the file it should mergesort.

You may assume that the input file has one number on each line.

Master Process

The master process should split the input file into as many files, with a specified prefix for the file names, as the number of processes. Suppose the input file is called `input.txt`, and number of slave processes is 5, then the command

```
split -n l/5 input.txt
```

will split the file into 5 files without splitting lines. Note that the number of lines in each file may not be the same. Also note that the option in the above command is "ell"/5.

Then, the master process should open a server socket at a port, say S.

The master process should then connect to the daemon on its computer and have it initiate N parallel daemons and run the program in slave mode and use port P to contact the master process.

The master process should then wait for notification from the slave processes. Once all the slave processes have notified the master process, the master process should print out a message stating that the sorting is done, and then send all the slave processes a message to exit.

Slave Process

The slave process should mergesort the file it is responsible for, then send a message to the master process that the slave process is done, and then wait for the master process to inform the slave process that it should exit. Once the slave process gets the exit message from the master process, the slave process should exit.