Project 01

Teddy Brombach, Emily Obaditch, Lauren Ferrara

20 April 2016

1 Summary

The goal of this project was to use low-level system calls related to networking along with object-oriented programming in Python. Two programs were created: `thor.py` and `spidey.py`. `Thor.py` is a basically HTTP client that makes requests to a remote HTTP server. The program fetches the contents of the specified URL and prints them to standard out. `Thor.py` can run multiple processes and requests to the server using the -r and -p flag. `Spidey.py` is a basic HTTP server that supports directory listings, static files, and CGI Scripts. The program opens a socket in the specified port and handles HTTP requests using an HTTP Handler class. This class has multiple functions to deal with the different type of requests (for files, directories, or CGI scripts). The flags for spidey are -h to see what the usage is, -v sets the logging to debug, -f enables forking, -d sets the docroot, and -p changes the port.

The work was divided evenly between the group as we spent the majority of the time working on the project together. We used Bitbucket to push the code to a shared repository so when we worked on it individually everyone had the most recent version.

2 Latency

Latency can be described as the amount of time it takes for the programs to run. We measured latency using a shell script which ran `thor.py` on `spidey.py` 100 times. `Thor.py` was run with processes set to 10 and requests set to 10. We used the average time outputted by the logger to determine the latency for running `spidey.py` on a file, directory, and script. The average latencies are as follows in Table 1 and are displayed in Figure 1.

Type	Latency (single)	Latency (forking)
File	2.44	0.91
Directory	0.30	0.50
Script	13.73	4.90

Table 1: Latency (seconds)

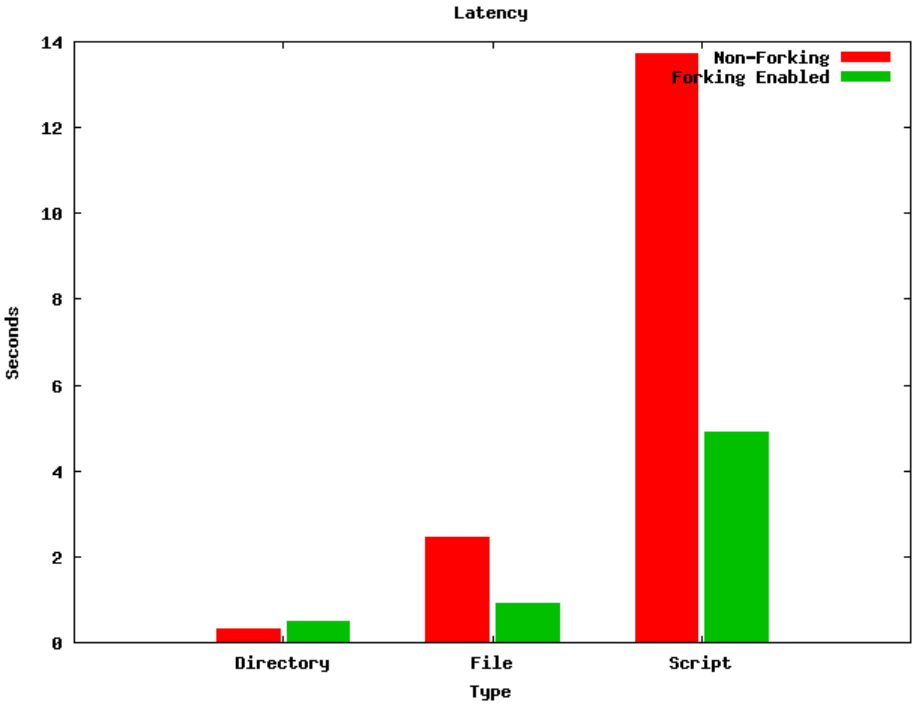

Figure 1: Latency Results

3 Throughput

Throughput can be described as the width of the pipe, or the amount of data that can be transferred within a given amount of time. Each throughput values corresponds with the amount of time it took to run `spidey.py` and `thor.py` on each filetype. The throughput values are as follows in Table 2 and are displayed in Figure 2.

Type	Throughput (single)	Throughput(forking)
File	3934	10,549
Directory	6827	4096
Script	699	1959

Table 2: Throughput (bytes per second

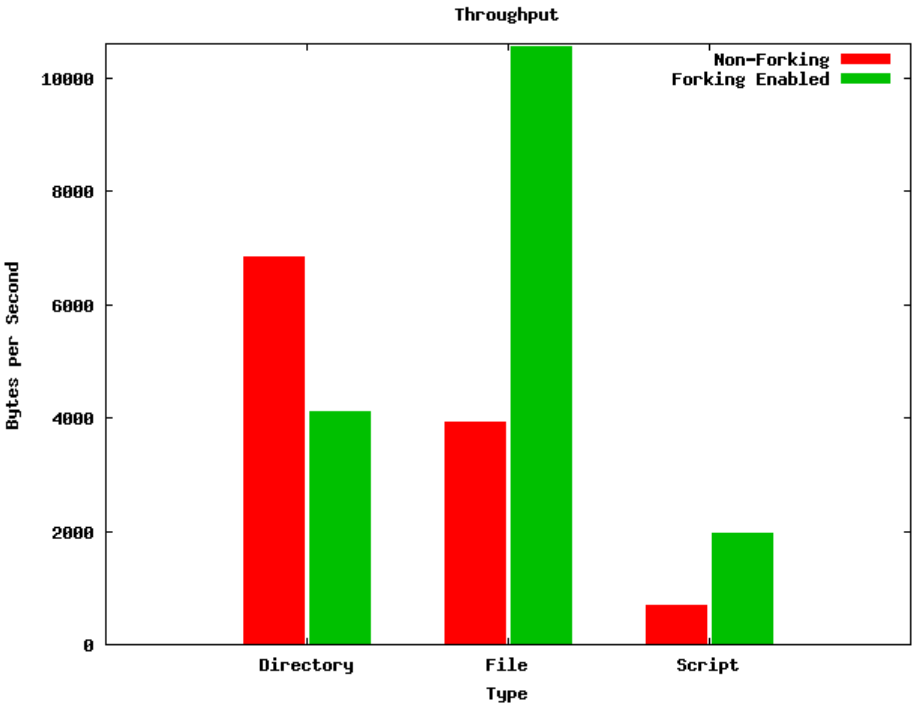

Figure 2: Throughput Results

4 Analysis

For latency, the script takes the most amount of time to run in the single mode because the program has to run another program rather than just list contents like it does when requesting a file of directory. As seen in Table 1 the forking mode runs in less time for a file and script. This is because multiple processes can be run at the same time in forking mode while in single mode the program can only handle one process at a time. The forking mode for the directory and script takes a longer amount of time because the program only needs to execute one process to list all the contents of a directory, thus making the forking mode unnecessary. The throughput test in single mode and forking mode shows that the program can run the most on directories, then files, and then executable. The throughput is higher for forking mode because multiple processes can be run at once which would speed up the process thus making more data being able to be transferred.

5 Conclusions

From the overall assignment we learned Object-Oriented Programming in python, how to interact with different sockets, how to use python to code in different languages (e.g. using python to make the HTML for `spidey.py`'s web page), and how to parse complicated strings. We also learned that all the work that we just did can be implemented in about 4 lines of code :). We also learned that socket programming is very interesting, while complicated and at times difficult to understand, the coolest thing of all time was seeing our requests appear on the terminal when someone else opened up the web page. From the experiments, we learned that forking mode helps speed up the requests for files and scripts. As the size of the file grows, the amount the forking helps the process increases dramatically.