Assignment 2

P2 - Prefork server

Submitted To - Dr. Hari Babu K.
Name - Chirag Singhal, Harshan B.
ID NOs - 2018A7PS0219P, 2018A7PS0166P

Design

This exercise develops a pre-forking model for a web server which maintains a process pool to handle client requests. Server binds to a port and creates child processes. Each child process of the server acts as a server and accepts a connection request and sleeps for one second then sends a dummy reply. Server maintains a minimum number of idle child processes `minSpareServers` and checks that at any point its not more than `maxSpareServers`. Each child handles a fixed number of requests before it is killed and a new child is created, called recycling.

The communication between the child processes and server happens using UNIX domain sockets. Whenever the number of idle child processes is less than `minSpareServers` server adds 1 process to the process pool and waits for one second and then if it's still less it will continue spawning child processes exponentially to the server pool till its 32 processes per second and the spawn rate becomes constant after that.

Users can press `Ctrl + C` to print the number of children currently active, and for each child how many clients it has handled.

Every child communicates with the parent with UNIX domain sockets created by `socketpair`. The parent uses I/O multiplexing across all these sockets to receive and send status messages and to self-regulate. Every process in the process pool can have 4 statuses:

- INIT
- IDLE
- BUSY
- EXIT

INIT status indicates that the child was successfully forked but waits for acknowledgement from the parent. The parent acknowledges the newly forked child by sending an IDLE status message.

IDLE status indicates that the child is a part of the spare servers and that it can handle new requests. The IDLE status is used by the parent to regulate the number of spare servers.

BUSY status indicates that the child has accepted a connection request and is busy handling the request. Parent uses this status to remove this process from the spare process pool. After the request is handled, the child sends an IDLE status message to signal that it is ready to handle more requests.

EXIT status is sent by the child to the parent to signify that it has completed `MaxRequestsPerChild` requests and that this process will exit naturally. Once the parent receives this status, it will create a new child process to

occupy its slot, which is called recycling. Recycling is done to make sure the spare process pool does not keep diminishing.

Status messages:

`@@` status messages are printed by the parent for every change happening in the process pool, i.e number of connections received and number of processes which became IDLE, and it is set/cleared using the `VERBOSE` flag in the code.
`#define VERBOSE 1` is the default parameter.

>> status messages are printed when a change is done to the process pool by the parent in the process of regulation, i.e. number of idle/busy/total processes and the action taken by the parent to regulate and the status of the action done.

~~ status message is printed when Ctrl+C is pressed. This prints the number of connections handled till now by all the current processes in the pool.

How to Run:

The following command compiles the executable.
```
$ make compile
```

You can run the program by the following command:
```
$ ./prefork_server.o <minSpare> <maxSpare> <maxReqChild>
```

To exit the process you can press Ctrl + Z, and terminate the process using
```
$ kill <pid>
```

Assumptions and Limitations:

- Each child handles requests sequentially and not concurrently.
- The HTTP request is assumed to be of 2000 bytes max size.
- The child responds with a dummy response (the exact request message is sent back as a response) 1 second after receiving the message. The dummy response is not according to the HTTP standards and will register as a `Non-2xx` response by any benchmarking tool. `ab` (apache benchmark) is used for testing this program.