**CS2106 Operating Systems**

**Lab 2 – Fork and Pipe Answer Sheet**

|  |  |
| --- | --- |
| Student ID 1: A0168721B | Name 1: Ang Wei Ming |
| Student ID 2: A0126400Y | Name 2: Quek Yang Sheng |
| Student ID 3: | Name 3 : |

**Question 1.** (5 marks)

Parent’s process ID: 18670

Child’s process ID: 18671

Child’s parent’s process ID (as reported by child): 1555 (upstart)

Parent’s parent’s process ID (as reported by parent): 18267 (bash)

Parent of the parent: bash

**Question 2.** (3 marks)

The web browser cannot connect to the web server because in the deliverHTTP function, the process is waiting to read input from the client and telnet is keeping the connection open without sending an input. Therefore the server cannot continue and exit the function to listen to other connections.

**Question 3a.** (5 marks)

The modifications I made, and my explanations for them are:

if(fork() == 0) {

deliverHTTP(connfd);

exit(0);

}

When the process accepts a connection, it forks a new process to handle the connection while the parent process continues to listen to incoming connections. Once the child finish executing deliverHTTP function, it exits.

**Question 3b.** (7 marks)

The maximum number of connections I can make is:

Less than 4194303 for 64 bit system and 32767 for 32 bit system

This is because whenever there’s an incoming connection, it is passed on to a new child process to handle the request. Therefore, the number of connections I can make is the number of child processes that can be forked. (Assuming that there is no limit on the sockets that can be created or resources available) There is a limit of 7769 processes that can be created in the Ubuntu VM for a non superuser, found using ulimit -u. However, since the webserver is running on a superuser environment, it is limited to the maximum number imposed on the OS found in kernel.pid\_max.

**Question 4a.** (3 marks)

We must close the ends of the pipe we are not using because the reading process will only stop reading when it encounters an EOF, which is triggered only when the all output end of the pipe is closed. Therefore, the reading process have to close the output end of the pipe before it starts reading until the writing process closes the output end. The writing process will only stop if it finishes writing the data into the pipe or when all input end of the pipe is closed. In the case when the reading process closes halfway while reading, data will fill up the pipe till it’s full and the writing process will not know to stop waiting because it’s input end of the pipe is still opened.

**Question 4b.** (5 marks)

This statement is false because…

**Question 5.** (7 marks)

The modifications I made, with explanation, are: