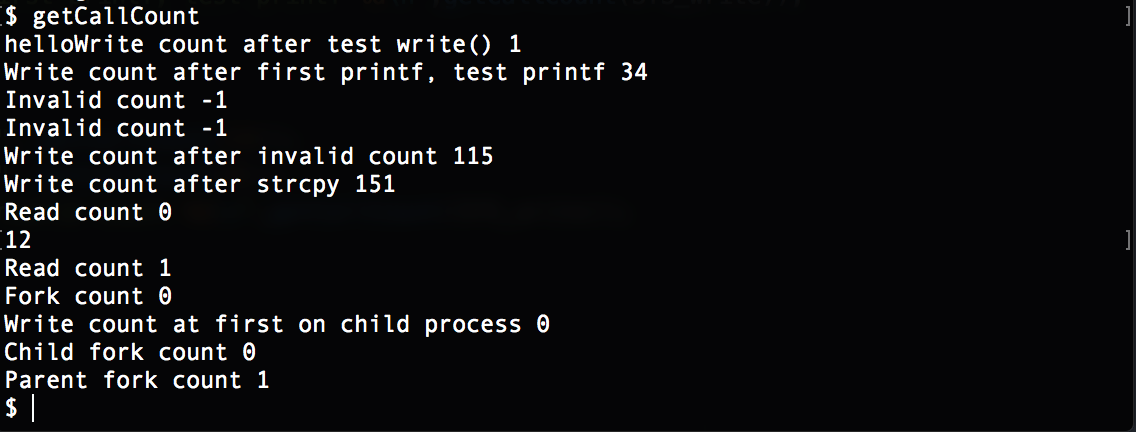
**TEST CASES AND RESULTS**

To perform my tests, I wrote a program in xv6 called **getCallCount.c**. This program will be then included in the Makefile (under UPROGS) so that when compile xv6, the program will also be compiled and output an executable.

Inside the program I would perform tests to make sure the **getCallCount** system call runs correctly.

In my program I performed four test cases for my new system call. The image below is my result in my test program.



* **The write() system call:**

Both **write()** and **printf()** would call the write system call. So to check if **getCallCoun**t works, first I run the **write()** function to the standard output. The **write()** function would then print to the standard output “hello”. After that, I would run the getCallCount() to check how many times write() system call has been invoked. The result **1** indicated that when running the **write()** function, the system only call **write()** once.

* **The printf() function:**

Using the **printf()** function means I will perform a test on the **write** system call. **printf()** would call a write system call for each of the character in its output string. So in line 4 of the image, the test printf would yield 34, which means the write system call has been executed 34 times. It is because the string “*Write count after test write() 1*” contains 32 characters, the \n, along with the write() system call would makes 34. The getCallCount() does not included the count of the **printf()** it is in (“*Write count after first printf, test printf 34*”) because the counter would only increment once the system call is done.

* **The invalid system call**:

In my test case I ask the getCallCount to return the count for system call number 0 and 23. Because both of them are invalid system call, so the function would return -1 indicate an invalid call.

* **The strcpy() function:**

The **strcpy**() does not invoke any system call. After checking for the invalid I run **strcpy()** and the write system call still only executed 151 times. The difference of 36 since the last time I **getCountCal**l the **write** system call is from the **printf** above, not from the **strcpy().** Furthermore, the read system call’s also still at 0, which means strcpy also does not do anything with write and read or any other system call.

* **The read system call:**

My test will use **read()** system call. The read system call will read from the standard input to the string **re**. After calling read I check the count for the **read()** and it goes to 1, which mean my **getCallCount** runs correctly.

* **The fork system call:**

The fork system call would fork from the parent process. That’s why in the child, the fork counter is 0 because no fork has been called from the child process. On the other hand, the parent process runs fork once so the fork counter for the parent is 1.

Also to test **if the counter will reset on each new process**, I also check the write system call counter on the child process, which is 0. It means that the program successfully reset the counter array.

This is my test program:

