FALL 2023
MOBILE AND EMBEDDED SYSTEM DESIGN AND APPLICATION
(COMP/ELEC 424/553)

Assignment 1: Kernel Hacking

Shaun Lin (hl116), S01435165

The goal of this project was to add a custom system call to the Linux kernel implementing a counter with **reset, increment, and initialize functionality**. To achieve this, my approach was to first modified arch/x86/entry/syscalls/**syscall_64.tbl** to add my custom syscall **hl116**, and second I modified kernel/**sys.c** to implement the counter for my custom syscall function. Then, I designed a test program **hl116_test.c** to test the syscall.

My first challenge was trying to add a function to kernel/sys.c, it was difficult at first because I was unfamiliar with kernel coding. Additionally, my second challenge was not familiar with user-mode Linux (UML), I tried to run my test program on CLEAR until I realized I needed to run it within UML using the provided "/linux rootfstype=hostfs rw init=/bin/bash" command. In this project, I learned a lot and got more familiar with Linux kernel development. Here's an interesting idea that we can invent a system call to direct integration with chatGPT. For example, "sys_chatgpt" call could take a question string as an input argument and return chatGPT's response directly within the terminal without needing to open a browser or application. We can design an "AI kernel" that creates a clean interface between an AI and the kernel!

I asked chatGPT how to create a simple counter system call in the Linux kernel. It provided a 6-step process to modify the syscall table, declaring a syscall function, implementing the counter logic in kernel/sys.c, updating Makefile, recompiling the kernel, and testing with a C program. The steps seem accurate overall. However, chatGPT's solution initializes the counter variable **inside the syscall function**, whereas **my implementation used a static variable to persist the count between calls**. Also, this project should be running and testing in user-mode Linux (UML) which chatGPT didn't mention. In conclusion, chatGPT provided a great overview, but there are some differences from my real-world implementation. The AI gave a textbook response, while my solution required debugging real kernel code which was more challenging but also more intrigued me as a hands-on kernel hacking project.

Acknowledgements

I acknowledge the use of ChatGPT-3.5 to generate a few initial scripts for this project.

I entered the following prompt: “Hi ChatGPT, I working on a ELEC553 Embedded System assignment, need to make a counter system call in linux kernel/sys.c, assume it take 3 arguments of type int, when first input argument is 0, and second input argument is 1, the counter should increase by second input argument, how to implement this?” I used the output to help me quickly understand the steps for modifying the kernel, and use portions of the output as a starting script. I modified the output generated, discarding several suggestions and replacing them with my own ideas based on the research and reading I completed.

Additionally, I was confused about project requirements 2.(b)”Code commented in detail”, so I prompted ChatGPT-3.5 “Add annotation to my code.”, and based on the output made a few changes.

Finally, I prompted ChatGPT-3.5 to optimize my original text and made small changes based on its suggestions.

Prompt: "Hi ChatGPT, I working on a ELEC553 Embedded System assignment, need to make a counter system call in linux kernel/sys.c, assume it take 3 arguments of type int, when first input argument is 0, and second input argument is 1, the counter should increase by second input argument, how to implement this?". Open AI, ChatGPT-3.5, October 2, 2023.

Prompt: "Add annotation to my code.". Open AI, ChatGPT-3.5, October 2, 2023.

Appendix A

(4 points) Include a screenshot showing a portion or all of your system call code, and another screenshot showing a portion or all of your test file code.

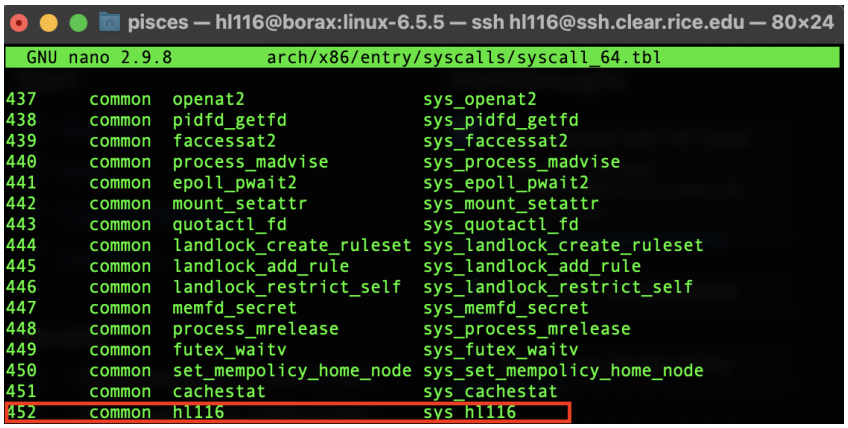

Figure 1. Screenshot a portion of arch/x86/entry/syscalls/syscall_64.tbl

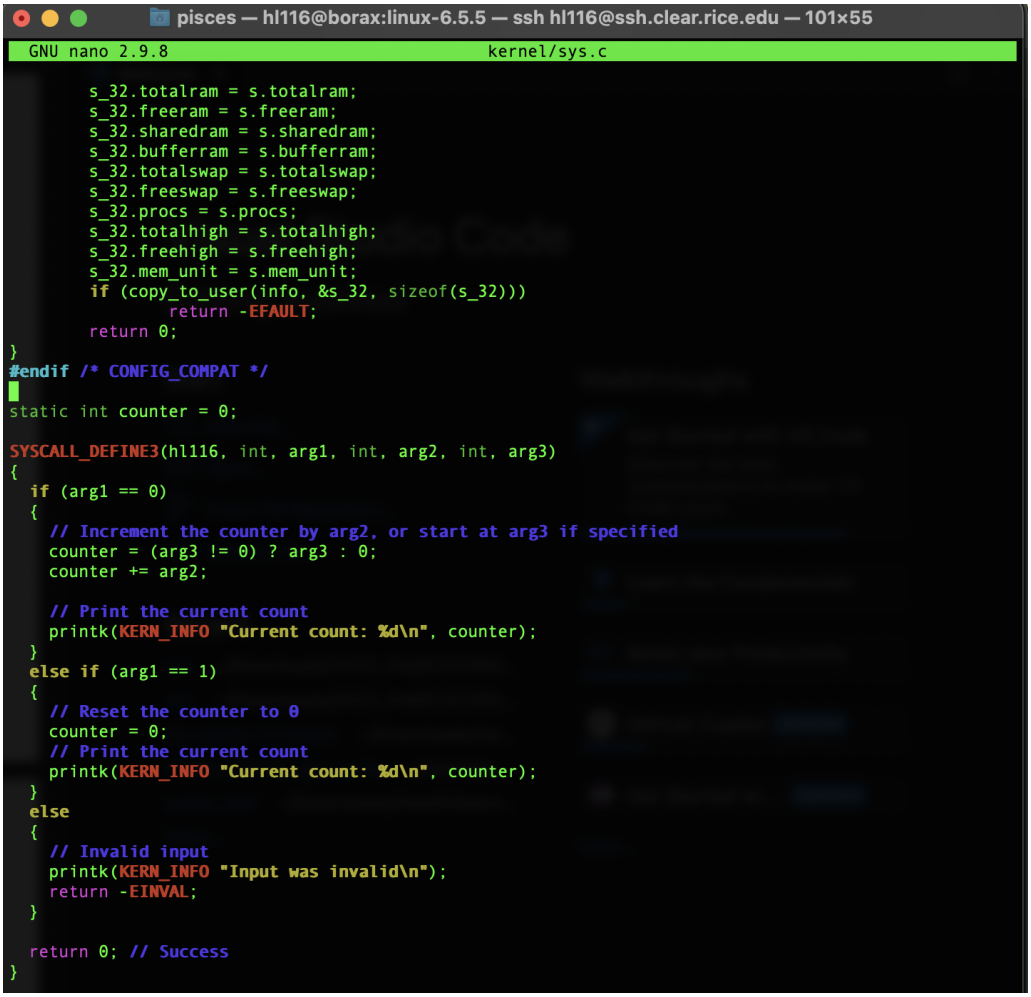

Figure 2. Screenshot a portion of kernel/sys.c

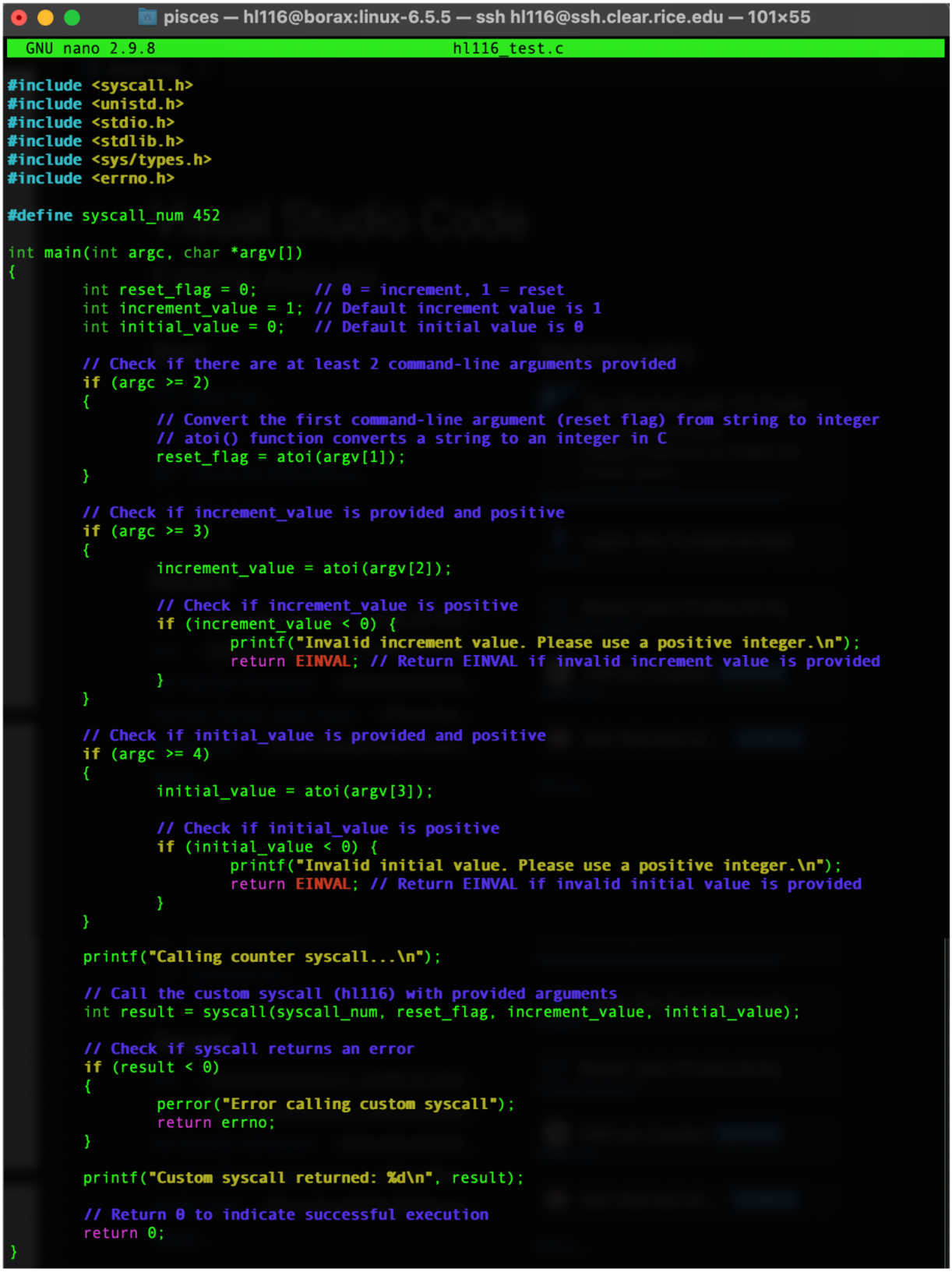

Figure 3. Screenshot all of hl116_test.c

(2 points) Include two screenshots of terminal output showing your counter working on one test or two.

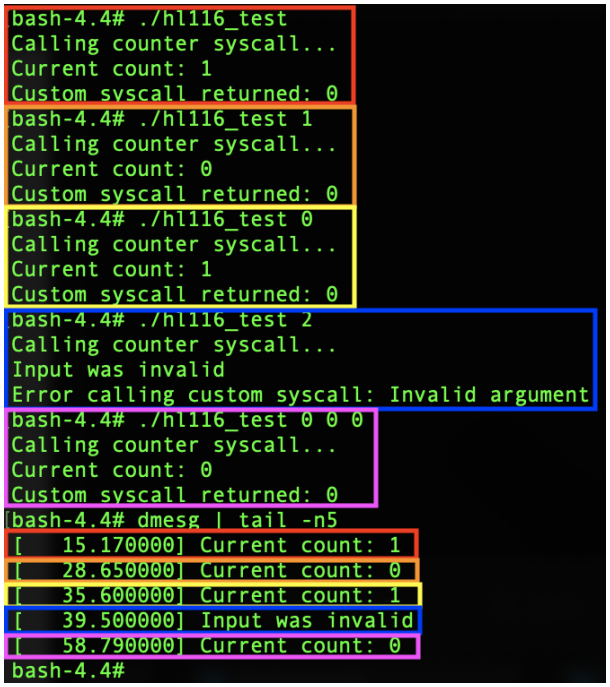

Figure 4. Screenshot of first counter test terminal output

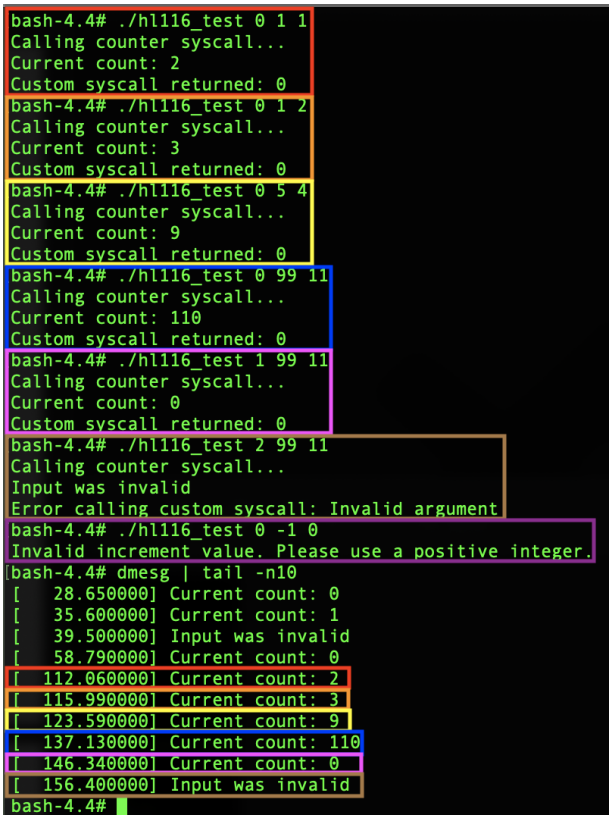

Figure 5. Screenshot of second counter test terminal output