

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.

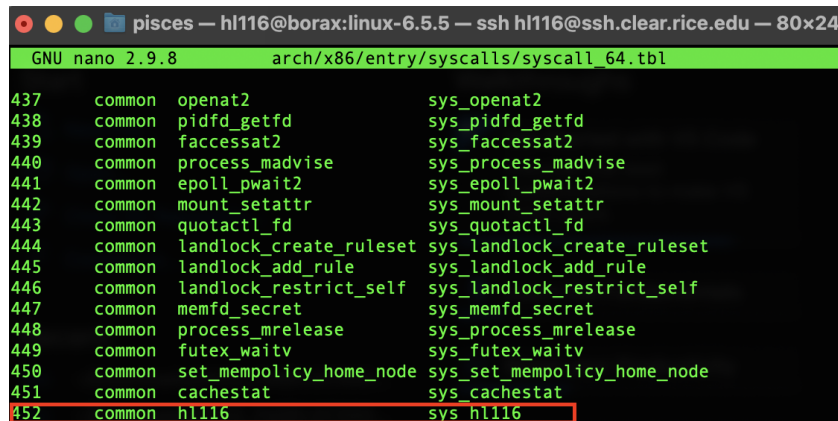
Finally, I prompted ChatGPT-3.5 to optimize my original text and made small changes based on its suggestions.

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

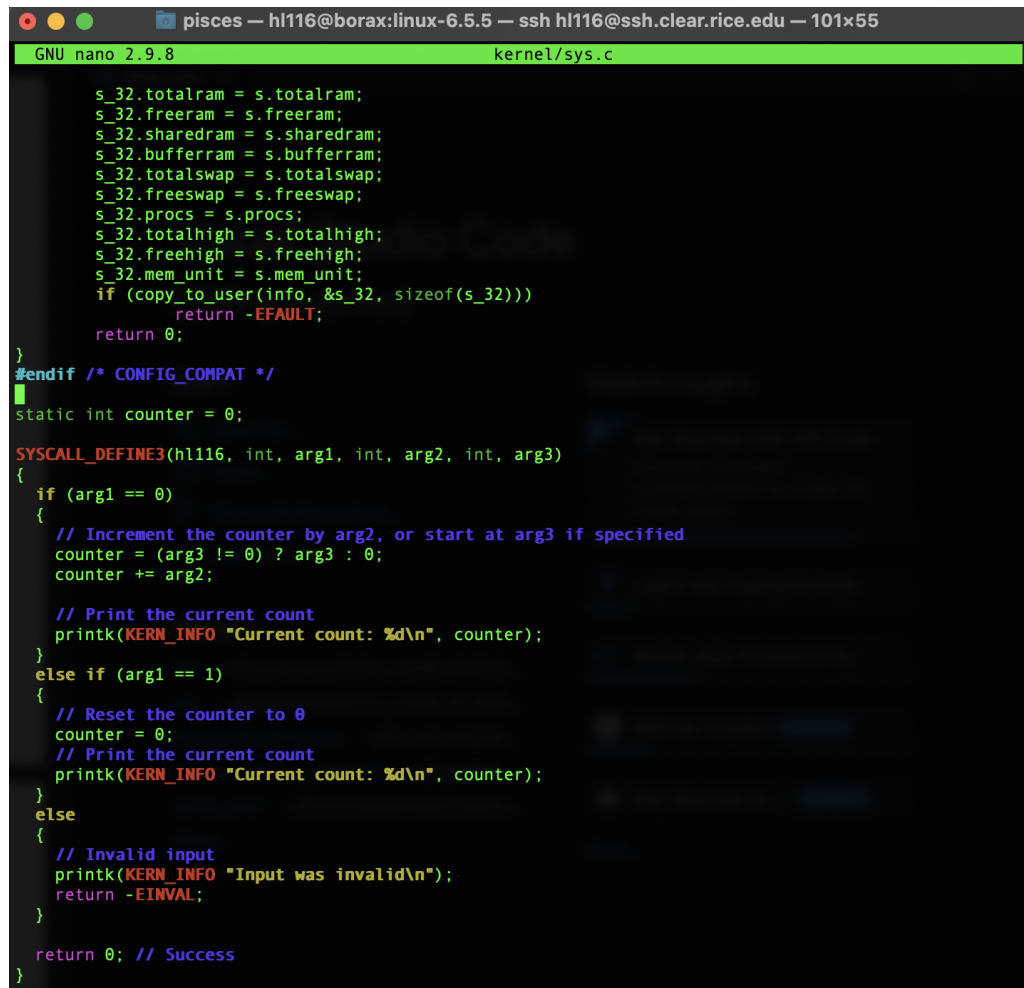
(4 points) Include a screenshot showing a portion or all of your system call code, and another screenshot showing a portion or all of your test file code.



```
GNU nano 2.9.8 arch/x86/entry/syscalls/syscall_64.tbl

437 common openat2 sys_openat2
438 common pidfd_getfd sys_pidfd_getfd
439 common faccessat2 sys_faccessat2
440 common process_madvise sys_process_madvise
441 common epoll_pwait2 sys_epoll_pwait2
442 common mount_setattr sys_mount_setattr
443 common quotactl_fd sys_quotactl_fd
444 common landlock_create_ruleset sys_landlock_create_ruleset
445 common landlock_add_rule sys_landlock_add_rule
446 common landlock_restrict_self sys_landlock_restrict_self
447 common memfd_secret sys_memfd_secret
448 common process_mrelease sys_process_mrelease
449 common futex_waitv sys_futex_waitv
450 common set_mempolicy_home_node sys_set_mempolicy_home_node
451 common cachestat sys_cachestat
452 common hl116 sys hl116
```

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



```
GNU nano 2.9.8 kernel/sys.c

s_32.totalram = s.totalram;
s_32.freeram = s.freeram;
s_32.sharedram = s.sharedram;
s_32.bufferram = s.bufferram;
s_32.totalswap = s.totalswap;
s_32.freeswap = s.freeswap;
s_32.procs = s.procs;
s_32.totalhigh = s.totalhigh;
s_32.freehigh = s.freehigh;
s_32.mem_unit = s.mem_unit;
if (copy_to_user(info, &s_32, sizeof(s_32)))
    return -EFAULT;
return 0;
}
#endif /* CONFIG_COMPAT */

static int counter = 0;

SYSCALL_DEFINE3(hl116, int, arg1, int, arg2, int, arg3)
{
    if (arg1 == 0)
    {
        // Increment the counter by arg2, or start at arg3 if specified
        counter = (arg3 != 0) ? arg3 : 0;
        counter += arg2;

        // Print the current count
        printk(KERN_INFO "Current count: %d\n", counter);
    }
    else if (arg1 == 1)
    {
        // Reset the counter to 0
        counter = 0;
        // Print the current count
        printk(KERN_INFO "Current count: %d\n", counter);
    }
    else
    {
        // Invalid input
        printk(KERN_INFO "Input was invalid\n");
        return -EINVAL;
    }

    return 0; // Success
}
```

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

```

GNU nano 2.9.8                               hl116_test.c

#include <syscall.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <errno.h>

#define syscall_num 452

int main(int argc, char *argv[])
{
    int reset_flag = 0;    // 0 = increment, 1 = reset
    int increment_value = 1; // Default increment value is 1
    int initial_value = 0; // Default initial value is 0

    // Check if there are at least 2 command-line arguments provided
    if (argc >= 2)
    {
        // Convert the first command-line argument (reset flag) from string to integer
        // atoi() function converts a string to an integer in C
        reset_flag = atoi(argv[1]);
    }

    // Check if increment_value is provided and positive
    if (argc >= 3)
    {
        increment_value = atoi(argv[2]);

        // Check if increment_value is positive
        if (increment_value < 0) {
            printf("Invalid increment value. Please use a positive integer.\n");
            return EINVAL; // Return EINVAL if invalid increment value is provided
        }
    }

    // Check if initial_value is provided and positive
    if (argc >= 4)
    {
        initial_value = atoi(argv[3]);

        // Check if initial_value is positive
        if (initial_value < 0) {
            printf("Invalid initial value. Please use a positive integer.\n");
            return EINVAL; // Return EINVAL if invalid initial value is provided
        }
    }

    printf("Calling counter syscall...\n");

    // Call the custom syscall (hl116) with provided arguments
    int result = syscall(syscall_num, reset_flag, increment_value, initial_value);

    // Check if syscall returns an error
    if (result < 0)
    {
        perror("Error calling custom syscall");
        return errno;
    }

    printf("Custom syscall returned: %d\n", result);

    // Return 0 to indicate successful execution
    return 0;
}

```

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.

```
bash-4.4# ./hl116_test
Calling counter syscall...
Current count: 1
Custom syscall returned: 0
bash-4.4# ./hl116_test 1
Calling counter syscall...
Current count: 0
Custom syscall returned: 0
bash-4.4# ./hl116_test 0
Calling counter syscall...
Current count: 1
Custom syscall returned: 0
bash-4.4# ./hl116_test 2
Calling counter syscall...
Input was invalid
Error calling custom syscall: Invalid argument
bash-4.4# ./hl116_test 0 0 0
Calling counter syscall...
Current count: 0
Custom syscall returned: 0
bash-4.4# dmesg | tail -n5
[ 15.170000] Current count: 1
[ 28.650000] Current count: 0
[ 35.600000] Current count: 1
[ 39.500000] Input was invalid
[ 58.790000] Current count: 0
bash-4.4#
```

Figure 4. Screenshot of first counter test terminal output

```
bash-4.4# ./hl116_test 0 1 1
Calling counter syscall...
Current count: 2
Custom syscall returned: 0
bash-4.4# ./hl116_test 0 1 2
Calling counter syscall...
Current count: 3
Custom syscall returned: 0
bash-4.4# ./hl116_test 0 5 4
Calling counter syscall...
Current count: 9
Custom syscall returned: 0
bash-4.4# ./hl116_test 0 99 11
Calling counter syscall...
Current count: 110
Custom syscall returned: 0
bash-4.4# ./hl116_test 1 99 11
Calling counter syscall...
Current count: 0
Custom syscall returned: 0
bash-4.4# ./hl116_test 2 99 11
Calling counter syscall...
Input was invalid
Error calling custom syscall: Invalid argument
bash-4.4# ./hl116_test 0 -1 0
Invalid increment value. Please use a positive integer.
bash-4.4# dmesg | tail -n10
[ 28.650000] Current count: 0
[ 35.600000] Current count: 1
[ 39.500000] Input was invalid
[ 58.790000] Current count: 0
[ 112.060000] Current count: 2
[ 115.990000] Current count: 3
[ 123.590000] Current count: 9
[ 137.130000] Current count: 110
[ 146.340000] Current count: 0
[ 156.400000] Input was invalid
bash-4.4#
```

Figure 5. Screenshot of second counter test terminal output