CS4500 Operating Systems
Section 001
Project 2
Kernel Module Process Information and System Call Analysis

Josh Manchester, Ronan Hella
Date of Submission: 23 Feb 2025

Statement: “We have neither given nor received unauthorized assistance on this work.”

1. Virtual Machine Information

Directory and Name of Virtual Machine:

Name: josh_ronan under `jmanches` profile

Directory: `/home/josh/Public/Project_01/project_02`

Username: `josh`

Password: `KMJNkmjn9090`

Path to Source Code:

`/home/josh/Public/Project_01/project_02`

(Subdirectories: part_1, part_2, part_3)

2. Description

2.1 How We Solved the Problems

- **Part 1 (HelloWorld Kernel Module):**

  We created a simple kernel module that prints "Hello world!" when loaded and "Goodbye world!" when removed. We fixed issues in the provided `Makefile` by correcting spacing and using the proper object file declaration (e.g., "`obj-m += hello.o`").

- **Part 2 (Print Self Kernel Module):**

  We implemented a module that uses the current macro to identify the current process and prints its name, PID, and state. The module then traverses the parent process chain until reaching `init`. Adjustments were made to accommodate updates in header files and structure definitions.

- **Part 3 (Print Other Kernel Module):**

  We developed a module that takes a process PID as an argument and prints the target process's information (name, PID, state), then traverses the process's parent chain until reaching PID 1. We used module parameters and the function `pid_task(find_vpid(target_pid), PIDTYPE_PID)` for process lookup. Corrections in the `Makefile` (using the proper `.o` extension) were also applied.

- **Part 4 (Kernel Modules and System Calls Questions):**

  What is the difference between a kernel module and a system call?

  - A **kernel module** is dynamically loadable code that extends the functionality of the kernel (e.g., drivers, file systems) without requiring a reboot.
  - A **system call** is a fixed, statically defined interface within the kernel that allows user-space applications to request services (like file I/O, process management) from the kernel.

    Reflection on a 20-year-old kernel module example:

- Although the basic concept of kernel modules remains the same, modern kernels have significantly updated APIs, security measures (e.g., module signing), and improved stability and performance.
- Legacy examples may fail to compile or run due to changes such as renamed structure fields (e.g., `task_struct` state field) and stricter policies. However, these changes are beneficial overall as they improve system security and reliability.

2.2 What We Learned

- We gained hands-on experience with writing and compiling Linux kernel modules.
- We learned to pass parameters to modules and retrieve process information from within the kernel.
- We improved our troubleshooting skills by debugging `Makefile` issues and adapting to changes in kernel APIs.
- We developed a deeper understanding of how kernel modules extend the functionality of the Linux kernel and the differences between modules and system calls.

2.3 How to Run Our Code

For each part, follow these steps:

a. **Open a terminal and navigate to the corresponding directory:**

For Part 1:
```
cd /home/josh/Public/Project_01/project_02/part_1
```

For Part 2:
```
cd /home/josh/Public/Project_01/project_02/part_2
```

For Part 3:
```
cd /home/josh/Public/Project_01/project_02/part_3
```

(Note: Part 4 does not require code execution.)

b. **Compile each module using:**
```
make
```
(This produces the .ko file for the module.)

c. **Load the module:**

For Part 1:
```
sudo insmod hello.ko
```

For Part 2:
```
sudo insmod print_self.ko
```

For Part 3 (include the process PID parameter):
`sudo insmod print_other.ko target_pid=1234`
(Replace 1234 with a valid process PID, e.g., from `pgrep bash`.)

d. **View the module output by running:**
`sudo dmesg -T | tail`

e. **Remove the module with:**

For Part 1:
`sudo rmmod hello`

For Part 2:
`sudo rmmod print_self`

For Part 3:
`sudo rmmod print_other`

(After each removal, check the output with `sudo dmesg -T | tail` to verify removal.)

f. **Clean up build artifacts using:**
`make clean`

3. Screenshots of Output

3.1 Code Output Screenshots

- **Part 1 Output:**

```
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_1$ make
make -C /lib/modules/6.8.0-53-generic/build M=/home/josh/Public/Project_01/project_02/part_1 modules
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-53-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-13 (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
  You are using:           gcc-13 (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
  CC [M]  /home/josh/Public/Project_01/project_02/part_1/hello.o
  MODPOST /home/josh/Public/Project_01/project_02/part_1/Module.symvers
  CC [M]  /home/josh/Public/Project_01/project_02/part_1/hello.mod.o
  LD [M]  /home/josh/Public/Project_01/project_02/part_1/hello.ko
  BTF [M] /home/josh/Public/Project_01/project_02/part_1/hello.ko
Skipping BTF generation for /home/josh/Public/Project_01/project_02/part_1/hello.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-6.8.0-53-generic'
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_1$ sudo insmod hello.ko
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_1$ sudo dmesg -T | tail -1
[Sun Feb 23 14:48:03 2025] Hello, world!
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_1$ sudo rmmod hello
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_1$ sudo dmesg -T | tail -1
[Sun Feb 23 14:48:16 2025] Goodbye, world!
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_1$ sudo rmmod hello
rmmod: ERROR: Module hello is not currently loaded
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_1$ make clean
make -C /lib/modules/6.8.0-53-generic/build M=/home/josh/Public/Project_01/project_02/part_1 clean
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-53-generic'
  CLEAN   /home/josh/Public/Project_01/project_02/part_1/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-6.8.0-53-generic'
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_1$
```

- **Part 2 Output:**

```
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_2$ make
make -C /lib/modules/6.8.0-53-generic/build M=/home/josh/Public/Project_01/project_02/part_2 modules
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-53-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-13 (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
  You are using:           gcc-13 (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
  CC [M]  /home/josh/Public/Project_01/project_02/part_2/print_self.o
  MODPOST /home/josh/Public/Project_01/project_02/part_2/Module.symvers
  CC [M]  /home/josh/Public/Project_01/project_02/part_2/print_self.mod.o
  LD [M]  /home/josh/Public/Project_01/project_02/part_2/print_self.ko
  BTF [M] /home/josh/Public/Project_01/project_02/part_2/print_self.ko
Skipping BTF generation for /home/josh/Public/Project_01/project_02/part_2/print_self.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-6.8.0-53-generic'
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_2$ sudo insmod print_self.ko
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_2$ sudo rmmod print_self
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_2$ sudo dmesg -T | tail
[Sun Feb 23 14:51:36 2025] Unloading print_self module
[Sun Feb 23 14:51:53 2025] Current Process Name: insmod, PID: 41123, State: 0
[Sun Feb 23 14:51:53 2025] Parent Process Name: sudo, PID: 41122, State: 1
[Sun Feb 23 14:51:53 2025] Parent Process Name: sudo, PID: 41121, State: 1
[Sun Feb 23 14:51:53 2025] Parent Process Name: bash, PID: 39557, State: 1
[Sun Feb 23 14:51:53 2025] Parent Process Name: sshd, PID: 39554, State: 1
[Sun Feb 23 14:51:53 2025] Parent Process Name: sshd, PID: 39482, State: 1
[Sun Feb 23 14:51:53 2025] Parent Process Name: sshd, PID: 6652, State: 1
[Sun Feb 23 14:51:53 2025] Parent Process Name: systemd, PID: 1, State: 1
[Sun Feb 23 14:51:57 2025] Unloading print_self module
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_2$ make clean
make -C /lib/modules/6.8.0-53-generic/build M=/home/josh/Public/Project_01/project_02/part_2 clean
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-53-generic'
  CLEAN   /home/josh/Public/Project_01/project_02/part_2/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-6.8.0-53-generic'
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_2$
```

- **Part 3 Output:**

```
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_3$ make
make -C /lib/modules/6.8.0-53-generic/build M=/home/josh/Public/Project_01/project_02/part_3 modules
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-53-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-13 (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
  You are using:           gcc-13 (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
  CC [M]  /home/josh/Public/Project_01/project_02/part_3/print_other.o
  MODPOST /home/josh/Public/Project_01/project_02/part_3/Module.symvers
  CC [M]  /home/josh/Public/Project_01/project_02/part_3/print_other.mod.o
  LD [M]  /home/josh/Public/Project_01/project_02/part_3/print_other.ko
  BTF [M] /home/josh/Public/Project_01/project_02/part_3/print_other.ko
Skipping BTF generation for /home/josh/Public/Project_01/project_02/part_3/print_other.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-6.8.0-53-generic'
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_3$ pgrep bash
39557
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_3$ sudo insmod print_other.ko target_pid=39557
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_3$ sudo rmmod print_other
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_3$ sudo dmesg -T | tail
[Sun Feb 23 14:51:53 2025] Parent Process Name: sshd, PID: 6652, State: 1
[Sun Feb 23 14:51:53 2025] Parent Process Name: systemd, PID: 1, State: 1
[Sun Feb 23 14:51:57 2025] Unloading print_self module
[Sun Feb 23 14:55:51 2025] print_other: Searching for PID 39557
[Sun Feb 23 14:55:51 2025] print_other: Target Process: bash (PID: 39557, State: 1)
[Sun Feb 23 14:55:51 2025] print_other: Parent Process: sshd (PID: 39554, State: 1)
[Sun Feb 23 14:55:51 2025] print_other: Parent Process: sshd (PID: 39482, State: 1)
[Sun Feb 23 14:55:51 2025] print_other: Parent Process: sshd (PID: 6652, State: 1)
[Sun Feb 23 14:55:51 2025] print_other: Parent Process: systemd (PID: 1, State: 1)
[Sun Feb 23 14:56:03 2025] print_other: Unloading module
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_3$ make clean
make -C /lib/modules/6.8.0-53-generic/build M=/home/josh/Public/Project_01/project_02/part_3 clean
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-53-generic'
  CLEAN   /home/josh/Public/Project_01/project_02/part_3/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-6.8.0-53-generic'
josh@josh-virtual-machine:~/Public/Project_01/project_02/part_3$
```

Interpretation: The output demonstrates that our modules are correctly interacting with the kernel. The process information is retrieved and printed as expected, confirming that our implementation meets the assignment requirements.