NAME OF PROJECT:

================

Project 4 Memory Management

MEMBERS:

========

Brogan Bewley, Jaylen McKinney

STATEMENT:

==========

We have neither given nor received unauthorized assistance on this work.

VIRTUAL MACHINE (VM) INFORMATION:

=================================

NAME OF VIRTUAL MACHINE (VM): Centos7VM (under bbewley’s directory)

USERNAME:bbewley

PASSWORD: IloveCS4500

**Describe where the files can be found:**

Documents/Project4/Part1 & Documents/Project4/Part2

**Describe each file and the purpose it serves:**

In the Part1 directory, the main three files of note are va\_space.c, va\_space.ko, and the Makefile. The C file contains the coded portion for finding a process’ virtual address size using its PID as input. The Makefile is used to compile the code and the ko file is used to run the module.

In the Part2 directory, the main three files of note are va\_status.c, va\_status.ko, and the Makefile. The C file contains the code portion for finding whether or not a process’s associated address is in memory or on disk. The Makefile is used to compile the code and the ko file is used to run the code.

**Provide any special instructions to access or run your program:**

Part 1:

Run pgrep bash to get the PID of bash. Then, run make to build the project. Run sudo insmod va\_space.ko pid= (enter PID) to load the module into memory. To verify that the module has been loaded, run lsmod and look for va\_space. Use dmesg -T | tail to see the output of the module. To remove the module, run rmmod va\_space. Run dmesg -T | tail again to verify that the module was successfully removed.

Part 2:

Use pgrep bash to find a usable PID, and then pmap <PID from bash> to get a list of virtual addresses to use. Run make inside the Part2 directory to build the project. Then, run “sudo insmod va\_status.ko pid=<PID from earlier> address\_str=”0x<virtual address>” to load and run the program. To verify that the code has been loaded, run lsmod and look for va\_status. Then, run “sudo rmmod va\_status.ko” to unload the module. Finally, to check the output run “dmesg -T | tail” to get an output of whether the address is in memory or on disk and remove the module.

ROOT PASSWORD:

==============

IloveCS4500

DESCRIPTION AND REMEDIATION:

============================

Part 1:

(Jaylen)

The purpose of part one was to take the process ID and calculate its virtual address space. It took the pid as input and outputted the size of the process’s virtual address space. The Linux Kernel Module Programming Guide and the blog post “How The Kernel Manages Your Memory” helped out a lot when trying to figure out how to use the structs. One of the main issues of this part was the debugging. It took a while to figure out why I was getting unknown symbol errors for find\_vpid only to realize I was missing the module license. I also struggled to not get the entire VM to crash if there was a nonexistent pid being used. I had to do some more research to figure out the proper error code to use if this happened. I feel like I have gained a much better understanding of kernel programming after this project.

Part 2:

(Brogan)

The purpose of part 2 is to assess whether a virtual address for a specific project is in memory or not utilizing page table entries. The attached page helped with the page table walk for finding a valid page table. The main issue I ran into was getting an error of unknown symbols in the module which was referring to “find\_vpid”. I needed to utilize a module license which solved this issue. I actually realized this solution early on but accidentally added it to the wrong file which threw me off for a bit. I believe I also had to make some minor changes to some of the suggested code in the page table walk to have it run properly. Overall this project allowed me to get a more practical application to viewing page tables and looking over where they are in memory.