Nathaniel Fenoglio

1001634050

Lemuel Rivera

1000842851

**General Process Outline for Project 3 Including Problems and Solutions Encountered Along the Way**

**Assignment 1**

* We struggled with implementing a new practice system call to test that we still had the procedure in place for successfully modifying Makefile to compile a new system call source code file, adding new system call number and info to arch/x86/entry/syscalls/syscall\_64.tbl table, declaring a new system call in the include/linux/syscalls.h header file, then executing the correct command line commands to recompile the kernel with the new system call.

We did not realize at first that the current kernel version name at some point had changed from 4.14.217 to 4.14.217+ we are guessing that this happened when an update was performed. Choosing to boot into 4.14.217+ and modifying references to the kernel name resolved the problem.

* We found the information needed for the structure of how a process’s memory is accessible through the way the Linux kernel organizes it through reading “Understanding the Linux Kernel” and examples found online. The information that we needed for the assignment was to be found in the below structure paths.
  + task\_struct->active\_mm->mmap the vm\_area\_struct start of the virtual memory linked list for a process
  + task\_struct->active\_mm->map\_count contains the number of memory regions for a process
  + task\_struct->active\_mm->mmap.vm\_flags unsigned long with bits that represent the different file permissions of a virtual memory area
  + task\_struct->active\_mm->mmap.vm\_start virtual memory start address
  + task\_struct->active\_mm->mmap.vm\_end virtual memory end address
  + task\_struct->active\_mm->mmap.vm\_next virtual memory nodes next value, used to traverse the virtual memory linked list of a process
  + used one\_mem\_area.vm\_file->f\_path.dentry->d\_iname to get name of file of the virtual memory address if it exists
* When executing the second test program “test\_virt\_address\_stats\_threads.c”, we found that the different threads printed the exact same memory information when calling the virt\_address\_stats system call. This is what was expected as threads share the same memory address space of the process that they exist within.

**Assignment 2**

* We were unable to get sys\_virt\_address\_status.c to compile until we added p4d extra level of multilevel page directory. This extra level of the page directory was not discussed in the book and we are guessing that it was added to the Linux kernel after the book was written.
* Pathway through multi-level page table to take to get to the page table entry and access the flags:
* global directory -> p4 directory -> upper directory -> middle directory -> table
* We found the macros needed to obtain the page table entries needed to traverse the multiple levels of the 4-level page directory.
* pgd\_t \*pgd = pgd\_offset(memory, mem); //page global directory
* p4d\_t \*p4d = p4d\_offset(pgd, mem);
* pud\_t \*pud = pud\_offset(p4d, mem); // page upper directory
* pmd\_t \*pmd = pmd\_offset(pud, mem); // page middle directory
* pte\_t \*ptep = pte\_offset\_kernel(pmd, mem); // page table entry pointer
* And the macros needed to read the different flags for
  + data = pte\_present(pte); if the data is in memory or on disk (Present flag value)
  + ref = pte\_young(pte); if the page that the memory belongs to has been referenced (Referenced flag value)
  + dirty = pte\_dirty(pte); if the page is dirty (modified since last write) (Dirty flag value)

Honor Code

I pledge, on my honor, to uphold UT Arlington’s tradition of academic integrity, a tradition that values hard work and honest.

effort in the pursuit of academic excellence. I promise that I will submit only work that I personally create or contribute to

group collaborations, and I will appropriately reference any work from other sources. I will follow the highest standards of

integrity and uphold the spirit of the Honor Code.

Nathaniel Fenoglio

Lemuel Rivera