CS4375-13948 Fall 2023 Homework Report

Franz Reyes Submitted on 12/01/2023

Fareyes4@miners.utep.edu

**UG HW5: Anonymous Memory Mappings for xv6**

Task 1.

1. In the implementation of mmap() and munmap() for the private program, mmap() is used to request memory mappings, and munmap() is employed to release or unmap memory. The program initially aborted due to an unhandled page fault when attempting to access unmapped memory.
2. In order to fix this, modifications were made to the usertrap() function in kernel/trap.c. These changes included acquiring the process lock before accessing proc struct fields, checking for load or store faults, verifying the fault address within a mapped memory region, and allocating a physical memory frame using kalloc().
3. The debugging process involved addressing challenges such as working page faults, trying correct lock usage, managing memory allocation, and understanding the pre existent conditions.

A screenshot of a computer program

Description automatically generated

Task 2.

1. Two memory copy functions were updated: uvmcopy() is for copying private mapped regions, keeping changes separate between processes, while uvmcopyshared() is for copying shared mapped regions, allowing changes by any process to be seen by all.
2. In the modified fork() function, added code ensures proper copying of mmr tables for both private and shared regions. For private regions, changes by the parent or child won't be shared, and for shared regions, changes are visible to all.
3. The testing results, with different outcomes for prodcons1.c and prodcons2.c, likely reflect how shared memory regions are handled, showing the intended modifications in Task 2.

A screenshot of a computer screen

Description automatically generated

Task 3.

1. before implementing part b, the prodcons3 program likely produces incorrect results due to the absence of proper handling for shared memory writes by processes other than the original mapper. This can lead to inconsistencies when multiple processes attempt to write to the shared memory region simultaneously. For part b, after implementing the extra credit modification in usertrap() to address shared memory writes,
2. Running prodcons3 should now shows correct results, with shared memory being correctly managed across processes. Difficulties that came up in this task involved understanding and modifying the trap handler to handle shared memory scenarios, but these challenges were overcome through careful debugging and code review.

**Summary:**

In this assignment, I learned about memory mapping, where processes allocate and share memory. Lazy allocation, where memory is only allocated when accessed, helps optimize resources. I was also able to understand shared memory and its handling that allows changes in one process to be seen by others. Modifications to the fork() function that caused proper memory inheritance in new processes. Through the tasks I was able to figure out how addressing issues like page faults and race conditions.