Zachery Creech

Dr. Beck

COSC361

16 April 2021

Project 2

Modifications to the following files:

Makefile:

- Addition of 2 new terminal commands, “null_test” for testing the null pointer dereferencing, and “mem_test” for testing that mprotect() and munprotect() work correctly, including that fork() properly copies the page protections
  - _null_test\
  - _mem_test\
- Modification to CFLAGS so that the compiler does not generate trap 6 on null pointer dereference
  - CFLAGS = -fno-pic -static -fno-builtin -fno-strict-aliasing -O0 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer
- Modification to LDFLAGS so that processes are loaded into 0x1000 instead of 0
  - $(LD) $(LDFLAGS) -N -e main -Ttext 0x1000 -o $@ $^
    - Also did this to forktest, I wasn’t sure which ones had to be modified

null_test.c:

- A simple program that attempts to dereference a pointer pointing to NULL memory. It should generate trap 14

mem_test.c:

- A little more robust than the other test, it makes sure that mprotect() and munprotect() are working properly
- Also verifies that fork() properly copies page protections from parent to child

exec.c:

- When loading the program into memory, start process->sz at PGSIZE instead of 0 to reserve address 0x0 for null
  - sz = PGSIZE;

sysproc.c:

- Addition of two functions sys_mprotect(void) and sys_munprotect(void) to do trap security checking
  - See sysproc.c at the top

syscall.c:

- Addition of external sys_ function declarations for mprotect() and munprotect()
  - extern int sys_mprotect(void);
    extern int sys_munprotect(void);
- Addition of sys_ function tags in static syscalls[]
  - [SYS_mprotect] sys_mprotect,
    [SYS_munprotect] sys_munprotect,

defs.h:

- Addition of function signatures for mprotect() and munprotect()
  - int mprotect(void *, int);
    int munprotect(void *, int);

user.h:

- Addition of function signatures for mprotect() and munprotect()
  - int mprotect(void *, int)
    int munprotect(void *, int);

syscall.h:

- Define trap numbers for SYS_mprotect and SYS_munprotect
  - #define SYS_mprotect 22
    #define SYS_munprotect 23

usys.S:

- Addition of mprotect() and munprotect() to SYSCALL script list
  - SYSCALL(mprotect)
    SYSCALL(munprotect)

vm.c:

- Addition of function int mprotect(void *addr, int len)
  - See vm.c
  - Iterate through all pages from addr to addr+len and modify the PTE write bits to disable writing
    - *pte = *pte & ~PTE_W
- Addition of function int munprotect(void *addr, int len)
  - See vm.c
  - Identical to mprotect() except when modifying the PTE write bits
    - *pte = *pte | PTE_W

This project was generally easier than the first since I already knew how to implement system calls. Then it was really just a matter of figuring out what the functions should do, which was a bit difficult to grasp at first. However, once I figured out that vm.c contained a detailed breakdown of how paging works in xv6, it was not too difficult to put it all together. Reading mmu.h was also helpful for understanding how to use PGSIZE and PTE_W.

The null pointer dereferencing was somehow more confusing to me, despite it requiring very few code modifications. I tried modifying a few things in functions like argint and argstr, but then I started getting all kinds of weird traps. I was also confused somewhat by part of the wording of the project write-up: I thought I had to modify something in fork() rather than what fork() was calling. When copyuvm() finally led me to vm.c, everything made more sense.