CSC 4730 Fall 2023 Project 4

Add a guard page to xv6.

Objectives

As a result of executing this project the student will:

- Understand how a NULL dereference is handled in a modern OS.
- Understand how programs are loaded in a Linux / Unix environment.
- Understand how an OS written in a higher level language manipulates the memory management unit (MMU).
- Understand the in-memory layout of a process.
- Have experienced a case of *"none of it works until all of it works"*.

Overview

Presently, `main()` is located at address 0 in the virtual address space of a process. This means, when the common programming error of dereferencing a NULL pointer, the process can read program code - probably NOT what the programmer intended. In modern OSs, processes would not be able to write to program code. Why?

Your project is to move `main()` up to the next page of RAM and mark the vacated 0th page as invalid. In this way, page 0 becomes a "guard" page catching NULL dereferences.

Deadlines

You and your assigned partner have 7 days to complete this assignment plus one grace day. If you have not turned anything in by 11:59 PM on the eighth day, you will win a grade of zero. Therefore, to receive partial credit, hand in something before the project expires!

Details

Current memory layout

XV6 currently lays out process memory like this (starting at address 0 at top):

Contents
code
stack
heap

Your job is to make programs start beyond address zero (one page to be exact) so that a dereference of any address in the first page of memory will result in death.

The new layout should be like this:

Like this:

Contents
guard page
code
stack
heap

where the code starts at the first page boundary, not on the zeroth page.

Suggested sequence of steps

Make a copy of XV6

Do not defile the virgin.

Test program

The test program is very short. Declare a pointer, initialize it with NULL (must be in C not C++). Dereference the pointer. Boom. At least in Linux. Try this to verify.

Then, move the source code to the right place so that it is included in XV6. Test this. No crash should occur.

Figure out how xv6 sets up a process page tables

Look at how `exec()` works to better understand how address spaces get filled with code (loaded from disk) and initialized. That will get you most of the way.

Also look at `fork()`, in particular the part where the address space of the child is created by copying the address space of the parent. What needs to change (in `fork()` itself or a function `fork()` calls)?

Detect and correct any other dependencies

The rest of your task will be completed by looking through the code to figure out where there are checks or assumptions made about the address space. Hint: Answer the question:

```
How do I set the address of `main()` to be at 0x1000?
```

Cliff

As soon as you begin making changes, attempting to test can result in an OS crash. It is possible that nothing works until all of it works.

What to Hand In

Execute a `make clean`. Then zip the entire xv6 folder. Test the zip file to ensure you have done this correctly. Hand in the zip file.

Partners

- You must use only the partner I assign you.
- Only 1 person should hand in code. The code should clearly state who the partners are in a comment at the top of the main source file.
- The non-code-submitting partner must submit a text file "partner.txt" that states who the partners are.
- Failure to list partners correctly as described above removes 5 points from your grade.

Grading

The class will receive a rubric when grading is complete that describes errors and penalties.

Both partners get the same grade without exception.