University of Texas at Arlington

CSE 3320 - Spring 2020
Operating Systems
Project 3 - Memory Management [*]

Instructor: Jia Rao

Points Possible: 100
Handed out: April 8, 2020
Due date: 11:59 pm, Friday, May 8, 2020

Introduction

The outcome of this project is to implement a series of system calls in Linux kernel to report memory management statistics. The objective of this project is to get familiar with the following memory management components:

1. Process virtual address space.
2. Page tables.
3. Linux page cache.

Project submission

For each project, create a gzipped file containing the following items, and submit it through Blackboard.

1. A report that briefly describes how did you solve the problems and what you learned.
2. The Linux kernel source files that your modified or created.

Assignment description

This project reviews the key memory management concepts we studied in class: virtual memory, page tables and the page cache. Virtual memory often refers to the process address space assigned to user-space processes. Such virtual addresses need to be translated into physical addresses with the help of page tables.

[*]Disclaimer: This assignment is adapted from projects developed by Dr. Jason Nieh at Columbia University and Dr. Kai Shen at Rochester University.

The page cache is used to temporarily store recently read or written pages in memory. Read the chapter 10.4 in our textbook to get an overview of Linux's memory management.

You tasks

This project consists of three parts. You are going to write three system calls to collect memory management statistics.

Part 1 (50 pts)

Write a system call to report statistics of a process's virtual address space. The system call should take a process ID as input and outputs the following information about the process

1. The size of the process's virtual address space.
2. Each virtual memory area's access permissions.
3. The names of files mapped to these virtual memory areas.

Write two user-level programs to test your system call. One test program just calls the new system call and report the calling process's statistics. The other test program should create multiple threads and report information about individual threads. The purpose of the second test program is to study if threads share the same address space.

Hints

The Linux kernel uses the *memory descriptor* data structure to represent a process's address space. The memory descriptor `struct mm_struct` is defined in `<linux/mm_types.h>` and included in a process's `task_struct`. In `mm_struct` the `mm_users` field is the number of processes using this address space and the `mm_count` field is the reference count for this `mm_struct`. The `vm_area_struct` describes a single memory area over a contiguous interval in an address space. All the virtual memory areas together form a process's virtual address space. To calculate the size of a virtual address space, one only needs to sum the sizes of individual virtual memory areas (VMA). The VMAs in a process can be accessed in two ways. `mmap` in `mm_struct` points to a sorted linked list of all VMAs of a process. `mm_rb` points to a red-black tree of VMAs, which can be used to search for a given VMA. You can verify your result using `pmap`.

Part 2 (50 pts)

Given the above virtual memory areas used by a process, write a system call to report the current status of a specific address. The system call takes a virtual address of a process and outputs the following information:

1. If the data in this address is in memory or on disk.
2. If the page which this address belongs to has been referenced or not.
3. If the page which this address belongs to is dirty or not.

Hints

It is helpful to read subsection of Chapter 2: Page table handling of the reference book *Understanding the Linux Kernel (ULK)* (3rd edition). The page descriptor `struct page` defined in `linux/mm_types.h` contains information (i.e., the `flags` field) about the page. You need to figure out how to obtain a reference to the page descriptor given a virtual address and read to information from the page descriptor. To test if an address is in memory, you need to test the `present` flag of the address's corresponding page table entry. Note that Linux uses multi-level page tables, you might need multiple steps to reach the page table entry of a given virtual address.