University of Colorado at Colorado Springs

CS4500/5500 - Spring 2016
Operating Systems
Project 3 - Memory Management *

Instructor: Jia Rao

Points Possible: 100
Handed out: April 6, 2016
Due date: 11:59 pm, Saturday, April 23, 2016

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 to jrao@uccs.edu via email. Include “CS4500/5500_Spring2016_ProjectX” in the title of your email, where “X” is the project number.

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.

Each team should specify a virtual machine (VM) that the instructor can login to check the project results. Name the VM as “CS4500/5500_LastNameoftheFirstMember_LastNameoftheSecondMember” and create a new user account called instructor in your VM. Place your code in the home directory of the instructor account (i.e., /home/instructor). Make sure the instructor has access to your code. In your submission email, include your password for the instructor account.

*Disclaimer: This assignment is adapted from projects developed by Dr. Jason Nieh at Columbia University and Dr. Kai Shen at Rochester University.
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. 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 (35 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 (35 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.

Part 3 (30 pts)
Page cache in Linux temporarily saves recent accessed data in RAM. As we learned in class, if the amount of free memory falls below a threshold, the OS needs to reclaim (or evict) some of the cached data in the
page cache. Linux divides physical memory into multiple zones and maintains two LRU lists for each zone: the **active_list** and **inactive_list**. The OS only reclaims unused pages from the inactive list. When needed, pages can be moved from between the two lists based on their reference history. In this part, you are asked to write a system call that reports the following statistics about Linux page cache:

1. the current number of pages in the active list over all memory zones.
2. the current number of pages in the inactive list over all memory zones.
3. the current number of pages in the active list whose reference bit is set over all memory zones.
4. the current number of pages in the inactive list whose reference bit is set over all memory zones.

**Hints**
The memory zone descriptor **struct zone** defined in `linux/mmzone.h` contains pointers to the two LRU lists\(^1\) First, you need to get a reference to the individual memory zones, from where you can access the LRU lists. Macro `for_each_zone` can be used to iterate over all zones. A page’s status is saved in its page flags defined in `linux/page-flags.h`. It is helpful to read the Chapter 17: Page Frame Reclaiming in the reference book ULK.

---

\(^1\)Linux actually has five LRU lists in each zone for anonymous memory, file data and unevictable data.