Lab3 Page tables 实验要求

一、实验简介

本次 Lab 目的是了解虚拟内存和页表。修改页表以加速系统调用、检测哪些页表被访问过等。

二、实验准备

「1」准备 xv6 Lab3 源码.

「2」建议先阅读 xv6 book 的「Chapter 3」以及相关的代码文件.

> * ***kernel/memlayout.h***, which captures the layout of memory.
>
> * ***kernel/vm.c***, which contains most virtual memory (VM) code.
>
> * ***kernel/kalloc.c***, which contains code for allocating and freeing physical memory.

三、实验要求

3.1 代码实现

■ Part A: Speed up system calls

「该部分实现加速系统调用」

> Some operating systems (e.g., Linux) speed up certain system calls by <u>sharing data in a read-only region between userspace and the kernel</u>. This eliminates the need for kernel crossings when performing these system calls. To help you learn how to <u>insert mappings into a page table</u>, your first task is to implement this optimization for the getpid() system call in xv6.
>
> When each process is created, map one read-only page at <u>USYSCALL</u> (a virtual address defined in *memlayout.h*). At the start of this page, store a struct <u>usyscall</u> (also defined in *memlayout.h*), and initialize it to store the PID of the current process. For this lab, ugetpid() has been provided on the userspace side and will automatically use the USYSCALL mapping. You will receive full credit for this part of the lab if the ugetpid test case passes when running pgtbltest.

- You can perform the mapping in proc_pagetable() in *kernel/proc.c*.
- Choose permission bits that allow userspace to only read the page.
- You may find that mappages() is a useful utility.
- Don't forget to allocate and initialize the page in allocproc().
- Make sure to free the page in freeproc().

■ Part B: Print a page table

「该部分实现打印页表信息」

Write a function that prints the contents of a page table.

Define a function called **vmprint()**. It should take a pagetable_t argument, and print that pagetable in the format described below. Insert if(p->pid==1) vmprint(p->pagetable) in *exec.c* just before the return argc, to print the first process's page table.

「Print format」

The first line displays the argument to vmprint. After that there is a line for each PTE, including PTEs that refer to page-table pages deeper in the tree. Each PTE line is indented by a number of ".." that indicates its depth in the tree. Each PTE line shows the PTE index in its page-table page, the pte bits, and the physical address extracted from the PTE. Don't print PTEs that are not valid.

In the example, the top-level page-table page has mappings for entries 0 and 255. The next level down for entry 0 has only index 0 mapped, and the bottom-level for that index 0 has entries 0, 1, and 2 mapped.

「提示」

- You can put vmprint() in *kernel/vm.c*.
- Use the macros at the end of the file *kernel/riscv.h*.
- The function freewalk may be inspirational.
- Define the prototype for vmprint in *kernel/defs.h* so that you can call it from *exec.c*.
- Use **%p** in your printf calls to print out full 64-bit hex PTEs and addresses as shown in the example.

■ Part C: Detect which pages have been accessed

「该部分检测哪些页被 accessed」

Some garbage collectors (a form of automatic memory management) can benefit from information about which pages have been accessed (read or write). In this part of the lab, you will add a new feature to xv6 that detects and reports this information to userspace by inspecting the access bits in the RISC-V page table. The RISC-V hardware page walker marks these bits in the PTE whenever it resolves a TLB miss.

Your job is to implement pgaccess(), a system call that reports which pages have been accessed. The system call takes three arguments. First, it takes the starting virtual address of the first user page to check. Second, it takes the number of pages to check. Finally, it takes a user address to a buffer to store the results into a bitmask (a datastructure that uses one bit per page and where the first page corresponds to the least significant bit). You will receive full credit for this part of the lab if the pgaccess test case passes when running pgtbltest.

「提示」

- Read pgaccess_test() in *user/pgtlbtest.c* to see how pgaccess is used.
- Start by implementing sys_pgaccess() in *kernel/sysproc.c*.
- You'll need to parse arguments using argaddr() and argint().
- For the output bitmask, it's easier to store a temporary buffer in the kernel and copy it to the user (via copyout()) after filling it with the right bits.
- It's okay to set an upper limit on the number of pages that can be scanned.
- walk() in *kernel/vm.c* is very useful for finding the right PTEs.
- You'll need to define PTE_A, the access bit, in *kernel/riscv.h*. Consult the RISC-V privileged architecture manual to determine its value.
- Be sure to clear PTE_A after checking if it is set. Otherwise, it won't be possible to determine if the page was accessed since the last time pgaccess() was called (i.e., the bit will be set forever).
- vmprint() may come in handy to debug page tables.

提示非常详细，请同学们仔细参考。

3.2 问题回答

（1）在 Part A 加速系统调用部分，除了 getpid()系统调用函数，你还能想到哪些系统调用函数可以如此加速？

（2）虚拟内存有什么用处？

（3）为什么现代操作系统采用多级页表？

（4）简述 Part C 的 detect 流程。

四、实验预期结果

```
xv6 kernel is booting

hart 1 starting
hart 2 starting
page table 0x0000000087f6b000
..0: pte 0x0000000021fd9c01 pa 0x0000000087f67000
.. ..0: pte 0x0000000021fd9801 pa 0x0000000087f66000
.. .. ..0: pte 0x0000000021fda01b pa 0x0000000087f68000
.. .. ..1: pte 0x0000000021fd9417 pa 0x0000000087f65000
.. .. ..2: pte 0x0000000021fd9007 pa 0x0000000087f64000
.. .. ..3: pte 0x0000000021fd8c17 pa 0x0000000087f63000
..255: pte 0x0000000021fda801 pa 0x0000000087f6a000
.. ..511: pte 0x0000000021fda401 pa 0x0000000087f69000
.. .. ..509: pte 0x0000000021fdcc13 pa 0x0000000087f73000
.. .. ..510: pte 0x0000000021fdd007 pa 0x0000000087f74000
.. .. ..511: pte 0x0000000020001c0b pa 0x0000000080007000
init: starting sh
$
$ pgtbltest
ugetpid_test starting
ugetpid_test: OK
pgaccess_test starting
pgaccess_test: OK
pgtbltest: all tests succeeded
$
```

五、实验提交

（1）实验代码文件，提交前请 make clean.

（2）实验报告（中文即可，PDF 格式），需包含以下内容：

「1」实验实现的思路，相关代码截图，最终测试结果的截图.

「2」问题回答.

「3」实验中碰到何种问题，如何解决的.

「4」实验感想.

（3）报告截止日期：2022 年 11 月 8 日晚 22 点. 请按时提交.

（4）提交邮箱：各班级助教邮箱.

请将（1）、（2）放入一个文件夹，命名为：学号_姓名_oslab3.zip.

附：

助教邮箱

施恬安	302 室	21210240071@m.fudan.edu.cn
张卫华	204、205 室	15751006003@163.com
乔羽	202 室	2794953760@qq.com