山东大学 软件 学院

操作系统课程设计 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201600301079 | 姓名： 崔玉峰 | | 班级： 2016级软件4班 |
| 实验题目：  Laboratory 7: Extension of AddrSpace  Laboratory 8: System Calls Exec()and Exit() | | | |
| 实验学时：4 | | 实验日期： 2018/11/13 | |
| 实验目的：  **Laboratory 7:**  In this lab, you are required to  • extend the current implementation of class AddrSpaceso that Nachos can run multiple user programs.  • complete the print function for AddrSpaceas mentioned in Lab 6.  This lab will get you ready to implement the nachos system calls Exec()and Exit()in Lab 8  **Laboratory 8:**  In this lab, you are required to implement two Nachos systems calls: Exec(), Exit(). | | | |
| 硬件环境：  PC | | | |
| 软件环境：  虚拟机：VMWARE  操作系统：ubuntu 16.04.5 32位 | | | |
| 实验步骤与内容：    **Laboratory 7:**   1. **查看addrspace.cc的定义:**   **图片1**    可以看到addrspace原本的定义只是单纯的，逻辑地址等于物理地址，这无法满足我们的需求。需要借助BitMap对每个页进行合理的分配。    对于可执行文件代码和数据的实际地址读取也仅仅通过逻辑地址查找，所以需要将逻辑地址映射成物理地址在进行使用。   1. **创建BitMap** 2. **在**addrspace.h中声明一个静态的BitMap,类的所有对象共享一个bitmap。      1. 在addrspace.cc中将bitmap实例出来：        1. **更改addrspace.cc 物理地址的分配：** 2. 在AddrSpace创建时通过bitmap分配空闲页      1. 在AddrSpace被销毁时通过，将物理页收回：      1. 逻辑地址映射成物理地址：   **图片2**  **Laboratory 8:**     1. **查看源码：** 2. **nachos的系统调用，会被../test/start.s中被转化，然后被Mippim.cc的OneInstruction（）方法处理：**      1. **其中Syscall的处理，调用了machine.cc中的RaiseException方法,并且这个情况的处理使用return返回没有用break结束，对于其他进行PC+1操作的代码没有执行，所以需要添加AdvancePC方法；**        1. **machine.cc中的RaiseException方法，会调用ExceptionHandler将异常抛出，交给exception.cc处理：**   **图片3**     1. **进入exception.cc的异常处理函数：**     可以看到它只处理了系统调用Halt，对于Exce和Exit都还没有实现这些都需要我们进行实现。   1. **修改系统调用处理函数：**   修改ExceptionHandler()方法使其可以处理多种系统调用情况：     1. **SpaceID的实现：** 2. **可通过一个Bitmap 来为每个AddrSpace分配一个独一无二的SpaceID。** 3. **改变AddrSpace的定义添加一个SpaceId公有属性，并添加一个静态Bitmap \*spaceIds负责分配SpaceId；**      1. **在构造方法中分配SpaceId，在析构方法中释放SpaceId:**          1. **Exec(filename)的实现：** 2. **通过查看../userprog/progtest.cc 中的StartProcess（）方法可以看到通过调用./nachos -x filename 的方法是如何执行可以执行文件：**      1. 问题关键点就在于，filename的获取，通过filename正处于内核之中，通过借助ExceptionHandler()方法的注释可知系统调用的第一个参数的地址被放在R4寄存器之中。      1. 具体实现filename获取：     通过ReadRegister()方法读取r4寄存器的值，即filename的 地址，然后通过ReadMem()方法将整个filename字符串读出。   1. AdvancPC方法实现：      1. Exce()方法具体实现   首先读出文件名filename，然后执行，代码与StartProcess（）方法一致，最后将SpaceId写回寄存器r2，最后调用AdvancePC进行pc+1。       1. **Exit()的实现：**        1. **测试结果：** 2. **在../test文件夹中创建exec.c文件：**      1. **halt.c文件：**      1. **通过make 生成exce.noff,和halt.noff** 2. **到../lab7-8文件中修改进行make并执行./nachos -x ../test/exec.noff -d m:**       **mmexport1542196011883mmexport1542196015200** | | | |
| 结论分析与体会：  通过实验完成对与Exec()方法和Exit()进行了实现，并且测试通过，对nachos下实现系统调用的流程以及如何处理编写系统调用都有了了解，具体实现主要代码，已在上方贴出。大部分的实验该考虑的问题，实验指导书上已经给出了，主要的难点在于对于原来nachos代码是如何实现系统调用的，然后不断测试后最终完成了实验。 | | | |