**操作系统实习**

Lab4 虚拟内存 实习说明

本实习希望通过修改Nachos系统平台的底层源代码，达到“实现虚拟存储系统”的目标。

【背景描述】

目前，Nachos系统所实现的内存管理模块中，虽然存在地址转换机制，但是却没有实现真正的虚拟内存。

Nachos系统的内存分配必须在用户程序载入内存时一次性完成，故此，系统能够运行的用户程序的大小被严格限制(32 \* 128 B = 4KB，在Nachos系统中物理内存的页面大小PageSize为128 B，而每个用户程序最多只能获得32个物理页面，一旦超出则将导致系统出错)。

但是现代操作系统对内存管理的要求是支持多道程序，并且使得对任一程序而言，可用的内存空间应当是无限的，即需实现虚拟内存。同时还要提供存储保护机制。

【时间规划】

建议第一周完成 TLB异常处理、第二周完成分页式内存管理、第三周完成Lazy Loading.

【实习内容】

**一、TLB异常处理**

目前，Nachos系统对于内存的管理是基于软件模拟的TLB机制。其工作原理、异常处理、替换算法等方面，与分页式内存管理非常相像。

Exercise 1 源代码阅读

* 阅读code/userprog/progtest.cc，着重理解nachos执行用户程序的过程，以及该过程中与内存管理相关的要点。
* 阅读code/machine目录下的machine.h(cc)，translate.h(cc)文件和code/userprog目录下的exception.h(cc)，理解当前Nachos系统所采用的TLB机制和地址转换机制。

Exercise 2 TLB MISS异常处理

修改code/userprog目录下exception.cc中的ExceptionHandler函数，使得Nachos系统可以对TLB异常进行处理（TLB异常时，Nachos系统会抛出PageFaultException，详见code/machine/machine.cc）。

Exercise 3 置换算法

为TLB机制实现至少两种置 换算法，通过比较不同算法的置换次数可比较算法的优劣。

**二、分页式内存管理**

目前Nachos系统中，类Class Thread的成员变量AddrSpace\* space中使用TranslationEntry\* pageTable来管理内存。应用程序的启动过程中，对其进行初始化；而在线程的切换过程中，亦会对该变量进行保存和恢复的操作（使得类Class Machine中定义的Class Machine::TranslationEntry\* pageTable始终指向当前正在运行的线程的页表）。

Exercise 4 内存全局管理数据结构

设计并实现一个全局性的数据结构（如空闲链表、位图等）来进行内存的分配和回收，并记录当前内存的使用状

态。

Exercise 5 多线程支持

目前Nachos系统的内存中同时只能存在一个线程，我们希望打破这种限制，使得Nachos系统支持多个线程同时存在于内存中。

Exercise 6 缺页中断处理

基于TLB机制的异常处理和页面替换算法的实践，实现缺页中断处理（注意！TLB机制的异常处理是将内存中已有的页面调入TLB，而此处的缺页中断处理则是从磁盘中调入新的页面到内存）、页面替换算法等。

**三、Lazy-loading**

Exercise 7 我们已经知道，Nachos系统为用户程序分配内存必须在用户程序载入内存时一次性完成，故此，系统能够运行的用户程序的大小被严格限制在4KB以下。请实现Lazy-loading的内存分配算法，使得当且仅当程序运行过程中缺页中断发生时，才会将所需的页面从磁盘调入内存。

1. **Challenges（至少选做1个）**

Challenge 1 为线程增加挂起SUSPENDED状态，并在已完成的文件系统和内存管理功能的基础之上，实现线程在“SUSPENDED”，“READY”和“BLOCKED”状态之间的切换。

Challenge 2 多级页表的缺陷在于页表的大小与虚拟地址空间的大小成正比，为了节省物理内存在页表存储上的消耗，请在Nachos系统中实现倒排页表。

【实习建议】

1. 字节顺序

VMware虚拟机以及一般的PC机采用的是Little-Endian法字节顺序。

而Nachos模拟的处理器采用的则是Big-Endian法字节顺序。

请注意使用WordToMachine和ShortToMachine函数进行适当的转换。

1. 程序在地址空间的存放

在bin/noff.h中描述了程序的存放方式

* + - Struct segment代表了程序的一个分段。
    - Struct noffHeader中定义了程序的代码段、已初始化的数据段和未初始化的数据段。

1. 仔细阅读下列源代码文件：

code/machine/machine.h和code/machine/machine.cc

code/machine/translate.h和code/machine/translate.cc.

code/userprog/addrspace.h和code/userprog/addrspace.cc

code/userprog/exception.h和code/userprog/ exception.cc