**操作系统课程设计**

**实验内容、代码及实验报告提交要求**

**2022.7.18**

操作系统课程设计共有7个实验，除了实验1为环境准备，初步的测试运行及代码分析外，实验2-7涵盖了6个部分实验内容，分别为：**具有优先级的线程调度、信号量同步问题、基本文件系统扩展、文件的二级索引存储、系统调用与多道用户程序、虚拟内存。**

操作系统课程设计的得分，将主要考虑**平时上理论课的出勤情况(占比最小)，实验代码现场运行检查结果的情况(占比中等)，提交的代码及实验报告的情况(占比最大)。**

因个人原因，推迟代码现场运行检查，或延迟提交实验报告及代码的，将根据情况适当扣减分数。

一．实验内容

**实验1 Nachos实验环境准备、安装与源码分析(Lab1)**

**实验2 具有优先级的线程调度(Lab2)**

**实验3 使用信号量解决N线程屏障问题(Lab3)**

**实验4 基本文件系统扩展(Lab4)**

**实验5 具有二级索引的文件系统(Lab5)**

**实验6 系统调用与多道用户程序(Lab6)**

**实验7 虚拟内存(Lab7)**

实验1 Nachos实验环境准备、安装与源码分析(Lab1)

**实验内容：**

1. Nachos开发环境的安装测试(不含Linux系统本身及虚拟机软件的安装测试)。
2. Nachos实验代码框架(源码目录)的基本分析。
3. Nachos Makefile的基本分析。
4. 硬件机制模拟部分的实现原理分析，包括中断、时钟、CPU指令执行。

**注意：实验1没有代码的编写、演示验收及提交，因此分数完全来自于以上4部分实验内容在实验报告中的撰写质量情况。**

实验2 具有优先级的线程调度(Lab2)

**实验内容：**

1. 分析说明Nachos原有的线程调度策略。
2. 设计并实现具有静态优先级的非抢占式线程调度策略。
3. 以线程调试模式运行Nachos(./nachos -d t)，研究调试输出信息。上下文切换的次数与被测线程SimpleThread中打印输出的总行数一致吗？多余或缺少的上下文切换次数是什么原因造成的？请修改代码减少上下文切换的次数与被测线程SimpleThread中打印输出的总行数的差距。
4. 在实现了前面优先级调度的基础上，若要求实现优先级调度的老化(aging)，请给出在Nachos中**实现的具体方法**(不要求实现可运行的代码。在实验报告中用文字描述即可，必要时可在文字中结合关键代码片段、数据结构、对象等说明)。

**提示：List类中已有的SortedInsert方法可加以利用。**

**参阅：**

**code/lab2/n2**

**code/lab2/n2readme.txt**

**code/lab2/n2screen.txt**

**code/demo0/**

**《操作系统概念》第7版或第9版5.3.3节**

注1：若Lab2全部完成，演示提交的代码是3中完成的。

实验3 使用信号量解决N线程屏障问题(Lab3)

**实验内容：**

使用操作系统信号量机制，编写程序解决N线程屏障问题。

**包括：**

1. 分析说明Nachos的信号量是如何实现的。
2. 在Nachos中是如何创建及运行并发(而非线程自己主动调用Yield放弃CPU)线程的。
3. 先按“The Little Book of Semaphores”中3.6.4小节中的代码实现N线程屏障。用不同的随机数种子seed测试(./nachos -rs seed)，是否会发现有可能多个线程均判定自己为最后一个到达的线程，这个现象是什么原因造成的？该现象会导致N线程屏障出现与有题目要求不一致的错误码？
4. 请修改代码消除上面3中出现的现象。
5. 用不同的随机数种子测试，是否会发现各线程打印输出的rendezvous行的顺序，基本就是线程被创建的顺序(0,1,2…9)的现象？这是为什么，难道-rs选项没有起作用？试验在打印输出rendezvous之前加延迟(用软件空循环耗时)或Linux的sleep能否解决此问题，并解释为什么。
6. 请试着修改代码解决上面5出现的现象。提示：不用修改Nachos的核心实现代码，修改的是我们编写的N线程屏障的代码。

**参阅：**

**"The Little Book of Semaphores V2.2.1 -Allen B. Downey 2016.pdf"，3.6.4**

**code/lab3/n3**

**code/lab3/n3readme.txt**

**code/lab3/n3screen.txt**

**“操作系统课程设计 指导教程 -张鸿烈 2012.pdf”，P39，2.2.4中的说明**

**code/demo1/**

注1：若Lab3全部完成，演示提交的代码为6中完成的。

实验4 扩展文件系统(Lab4)

**实验内容：**

1. 扩展Nachos的基本文件系统。Nachos的文件系统是一个简单并且能力有限的系统，限制之一就是文件的大小是不可扩展的。通过扩展，使得文件的大小是可变的。在扩展写入文件内容时，一边写入，一边动态调整文件的长度及所占用的数据扇区。
2. 增加Nachos文件的最后修改时间，并在执行./nachos -D命令时显示。Nachos文件头中存储文件最后修改时间，时间值是从UTC 1970年1月1日00:00:00来的秒数(精确到1秒)，占用原来numSectors的存储位置(从磁盘存储空间效率上考虑，文件头中已经有了文件长度字节数，无需再存储文件内容占用的扇区数)。

**参阅：**

**操作系统课程设计 指导教程 -张鸿烈 2012.pdf，4.2节，pp.84-89**

**man 2 time**

**man ctime**

**man 2 stat**

**code/lab4/n4a、n4b、n4c、n4d**

**code/lab4/n4areadme.txt、n4breadme.txt、n4creadme.txt、n4dreadme.txt**

**code/lab4/n4ascreen.txt、n4bscreen.txt、n4cscreen.txt、n4dscreen.txt**

注1：仅普通文件的文件头最后修改时间字段有意义，并在执行./nachos -D命令时显示其时间。对其他文件头对象，在执行./nachos -D命令时不显示时间即可。

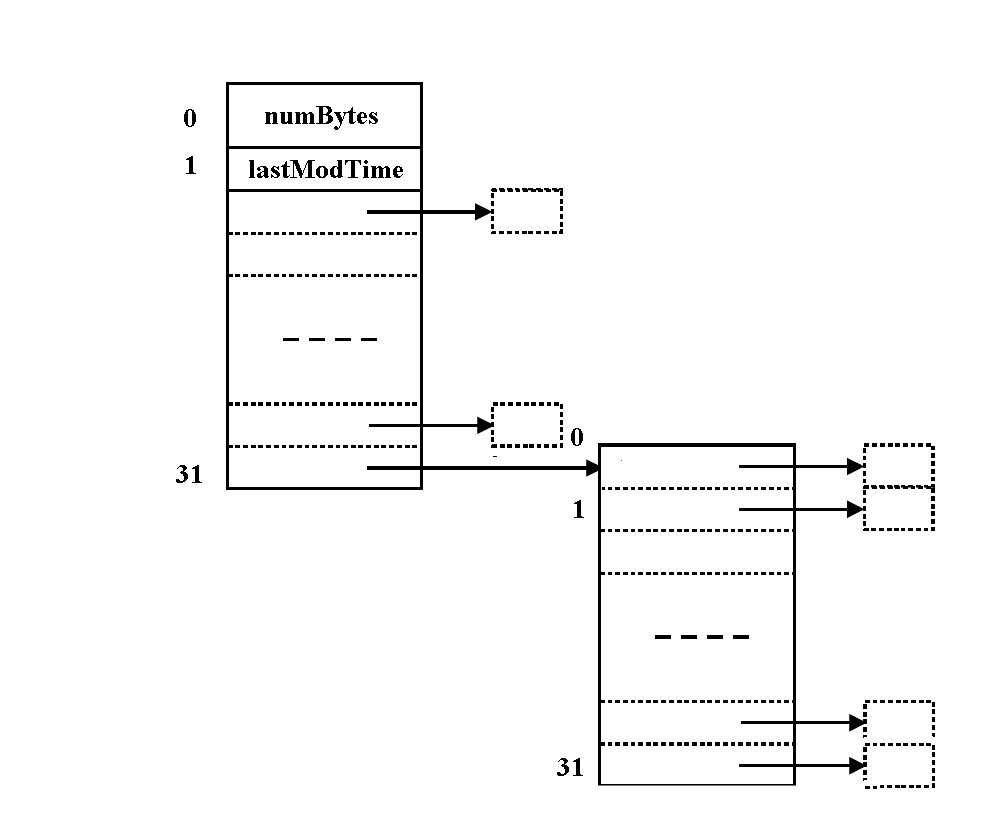
注2：若Lab4全部完成，演示提交的代码为带有文件最后修改时间的。

注3：对一般的OS，一个100字节的文件，open后lseek到偏移50处，write 10字节，close后，文件长度还是100字节，不会截短到60字节。这在实现Nachos的-hap命令行选项时需要注意。

实验5 具有二级索引的文件系统(Lab5)

**实验内容：**

1. Nachos系统原有的文件系统只支持单级索引，最大能存取30 \* 128 = 3840字节大小的文件。本实验将在理解原文件系统的组织结构基础上扩展原有的文件系统，在Lab4的基础上，设计并实现具有二级索引的文件系统。
2. 为Nachos增加命令行选项-DI。执行./nachos -DI时显示Nachos磁盘的以下信息：Nachos磁盘的总体大小，已使用空间大小，空闲空间大小，普通文件数目，全部普通文件的总字节数，全部普通文件占用的空间大小(不包括文件头占用的，但加上普通文件数据扇区的内碎片)，总内碎片字节数(仅计普通文件数据扇区造成的)。
3. 若要求为Nachos文件增加rwx权限(可读，可写，可执行)，请给出在Nachos中**实现的具体方法**(不要求实现可运行的代码。在实验报告中用文字描述即可，必要时可在文字中结合关键代码片段、数据结构、对象等说明)。



二级索引文件头i-node设计：

上图中，在扇区长度为128字节，int长度为4字节，均为Nahcos在X86平台上的默认值。构建具有二级索引的i-node，文件头原先的2-30项还是直接索引，最后的第31项指向二级索引块，这个二级索引块存放新的索引条目，有32项(数组下标为0~31)。

实现二级索引后的文件最大长度为：（29 + 32）\* 128 = 7808字节。

二级索引块是动态产生的，当文件大小不需要它时，一级索引块的最后一项可设置为-1，此时不存在二级索引块。当文件大小增长到一级索引无法支持时，再分配一个新的块存二级索引，并将二级索引块的扇区号存入一级索引块的第31项。

**参阅：**

**code/lab5/n5**

**code/lab5/n5readme.txt**

**code/lab5/n5screen.txt**

注1：若Lab5全部完成，演示提交的代码为带有-DI命令行选项的。

实验6 系统调用与多道用户程序(Lab6)

**实验内容：**

1. 扩展现有的class AddrSpace的实现，使得Nachos可以实现多道用户程序。按照实验指导书中的方法，完成class AddrSpace中的Print函数。实现Nachos系统调用：Exec()，一个用户程序启动另一个用户程序。注意本实验要求实现的Exec()系统调用，是在另一个地址空间运行指定的另一个用户程序，新程序并没有覆盖调用者的地址空间。这与Unix/Linux的系统调用exec()不同。
2. 在Nachos中增加并实现一个新的系统调用：PrintInt()，在用户程序中打印一个整数值。
3. 在实现了多道用户程序的基础上，若要求在Nachos中实现与Unix/Linux 的fork()/exec()功能类似的Nachos系统调用Fork()/Exec()，及写时复制 (copy-on-write) 机制，请给出在Nachos中**实现的具体方法**(实现时假定有足够的物理内存，无需页面置换。不要求实现可运行的代码。在实验报告中用文字描述即可，必要时可在文字中结合关键代码片段、数据结构、对象等说明)。

**参阅：**

**操作系统课程设计 指导教程 -张鸿烈 2012.pdf，pp.53-60，pp.64-68**

**code/lab6/n6**

**code/lab6/n6readme.txt**

**code/lab6/n6screen.txt**

**man fork**

**man exec**

注1：实现PrintInt系统调用需要修改code/userprog/syscall.h及code/test/start.s。在对原文件备份后，这两个文件可**原位修改**。

Lab6涉及2个用户进程，其源码文件分别命名为code/test/exec.c及code/test/halt2.c，其代码分别为：

/\* exec.c: Simple program to running another user program. \*/

#include "syscall.h"

int main()

{

SpaceId pid;

PrintInt(12345);

pid = Exec("../test/halt2.noff");

Halt();

/\* not reached \*/

}

/\* halt2.c: Just do a "syscall" that prints a number and shuts down the OS. \*/

#include "syscall.h"

int main()

{

PrintInt(67890);

Halt();

/\* not reached \*/

}

实验7 虚拟内存(Lab7)

**实验内容：**

1. 在未实现虚拟内存管理之前，Nachos在运行一个用户进程的时候，需要将程序运行所需全部内存空间一次性分配。虚拟内存实现将突破物理内存限制。本实验核心任务为根据理论学习中涉及的对换（Swapping）技术，在Lab6的基础上，设计并实现用户空间的虚拟内存管理。
2. 用户进程的帧数采用固定分配(建议5帧)，局部置换。
3. 实现“纯按需调页”(pure demand paging)。
4. 页置换算法可以采用LRU、增强型二次机会、二次机会、FIFO等算法之一，或自己认为合适的其他算法(不包括随机置换)。
5. 对class Statistics进行调用及修改，以便在程序结束时打出页故障次数及将牺牲页写入交换空间的次数。
6. 使用lab7目录中的示例程序n7(若lab7额外实现了多种算法，可用自己的lab7)，测试用户程序用同样ARRAYSIZE参数值的sort，但不同的页置换算法(详见code/lab7/n7readme.txt)多次运行n7。不同页置换算法运行结束时显示的user ticks数是否一样？解释这是为什么？
7. 最优页置换算法(OPT)有最低的页故障率，但需要未来的页面引用信息，因此不能用于实际环境，主要用于评估其他页置换算法的性能。在前述1-5实现的基础上，给出在Nachos中获得最优页置换算法页故障次数的**具体实现方法**(不要求实现可运行的代码。在实验报告中用文字描述即可，必要时可在文字中结合进关键代码片段、数据结构、对象等说明)。

**参阅：**

**操作系统课程设计 指导教程 -张鸿烈 2012.pdf，pp.60-63**

**操作系统概念 第7版或第9版 第9章相关部分**

**code/lab7/n7**

**code/lab7/n7readme.txt**

**code/lab7/n7screen.txt**

注1：要求用耗费用户内存比较多的code/test/sort.c作为虚拟内存的用户测试程序。可根据需要适当修改ARRAYSIZE值的大小，但不要修改除此之外的其他sort.c代码。

注2：可以让Nachos直接运行用户程序sort。也可以先运行一个Nachos用户程序，然后第一个Nachos用户程序用Exec()系统调用来运行sort。二者在演示验收时只需演示一个即可。

二．关于组队

根据项目的总体难度、工作量及本学期课程的安排，建议每组4人(不能超过4人)，也可3人。特殊情况请与老师提前联系。

同学们根据自己的时间、精力、知识、能力、特长、团队协作精神等情况，自行选择队友，并进行合理的分工协作。

组队请勿跨越课程的“头”(课号)。比如软工20.1-2这2个班之间可以自由组队，软工20.3-4这2个班之间可以自由组队，但软工20.1不能与软工20.3之间进行组队。

请保持组队的稳定性，一个队，一旦进行了首次现场演示验收，就不可改变队员人数及人员。

每个队，每个实验只需提交一份实验报告。每个队，全部实验提交一份含代码及所有实验报告的压缩包。

三．实验报告提交要求

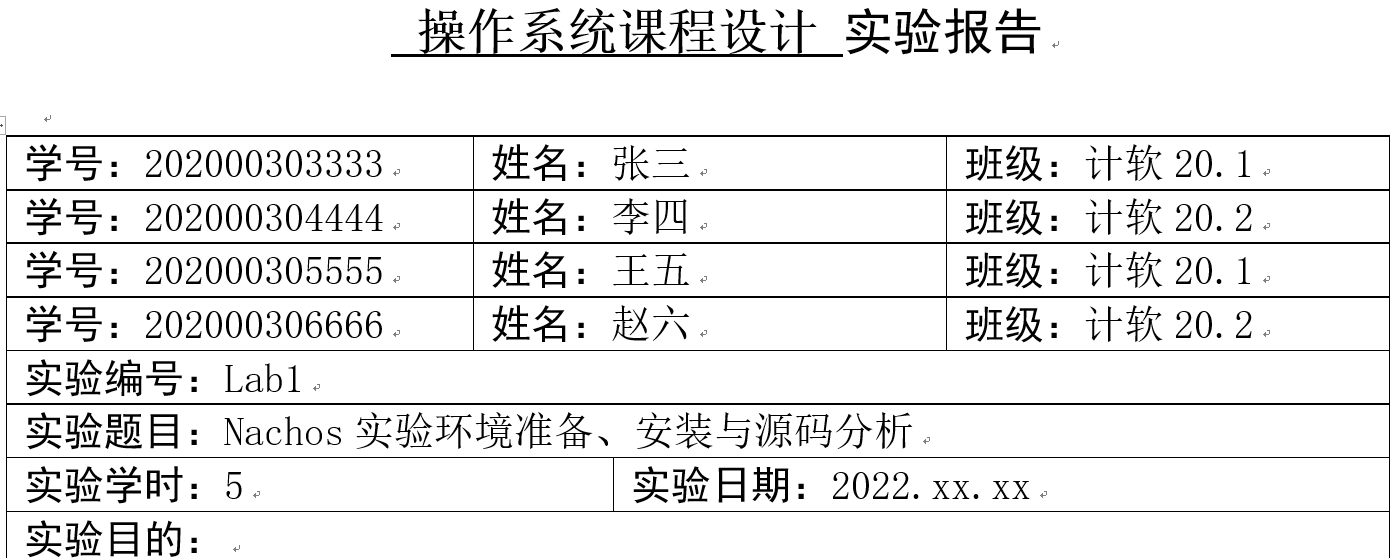
每个实验写一份实验报告(包括实验1)，文件名以组内任一同学的名字命名均可。实验报告文档命名格式为：

**实验n \_学号\_班级\_姓名.docx (或.doc)**

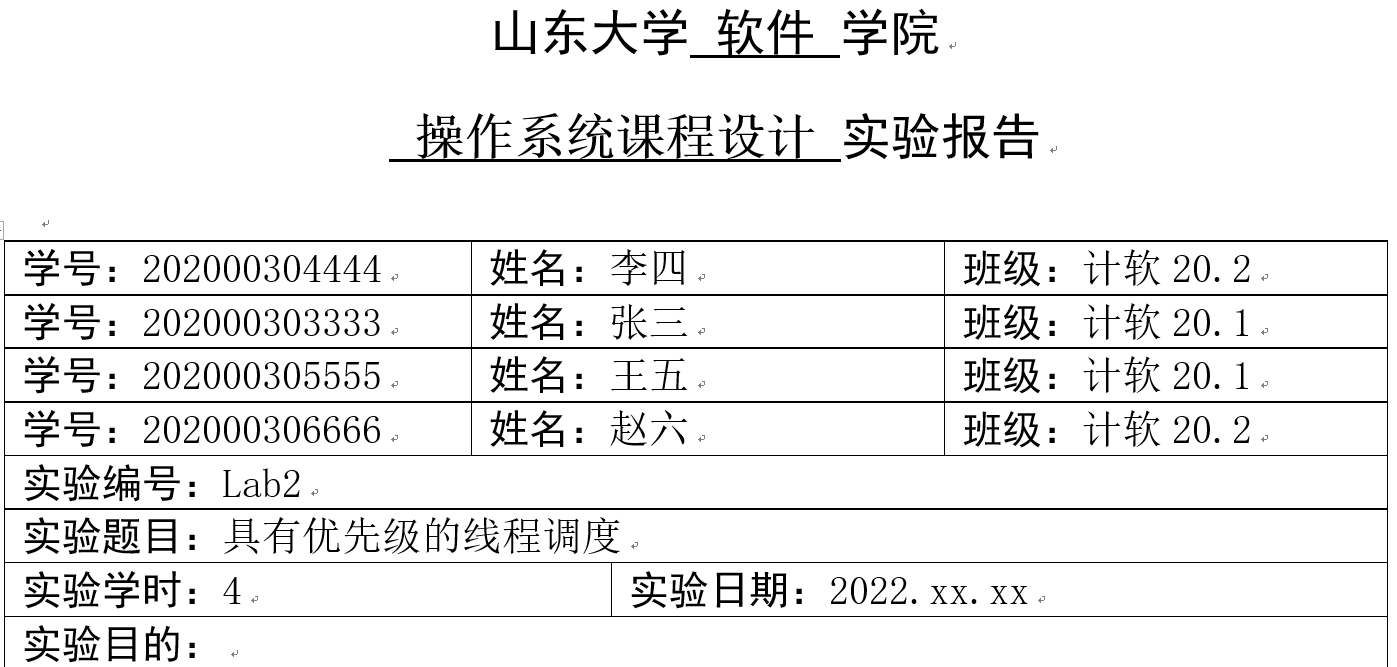
例如：

**实验1\_202000303333\_计软20.1\_张三.docx**

对一个组的具体某一个实验，**贡献值按本实验报告内头部的名字排序决定，且仅用于本实验**。例如下面这个组的实验1，张三得分最高：



而同一组的实验2的实验报告内头部可能为：



这样对该组的实验2，李四得分最高。

**各实验报告与完成的最终Nachos代码打成一个压缩包一起提交，请勿单独提交各个实验报告。**

四．代码及压缩包提交要求

各个实验的代码放在规定的目录中(code/lab2至code/lab7，包含源码及已编译好的可执行程序)，但syscall.h及start.s这两个文件保存在原来的目录中即可。并保证无需手动修改任何内容，make即可重新成功编译，并可实际测试运行成功。

最后将所有实验报告与最终代码打包成一个压缩包（限 .zip, .rar, .tar.gz三种格式之一）,压缩包文件名包含**所有组员**的班级代码、学号、姓名。请勿压缩包内嵌套压缩包。

压缩包文件命名格式：

**学号1\_班级代码1\_姓名1\_学号2\_班级代码2\_姓名2\_学号3\_班级代码3\_姓名3\_学号4\_班级代码4\_姓名4.zip (或.rar,.tar.gz)**

详细情况参考网盘中的压缩包模板：

**202000303333\_1\_张三\_202000304444\_2\_李四\_202000305555\_1\_王五\_ 202000306666\_2\_赵六.zip**

注：因为压缩包文件名有可能比较长，因此班级代码采用最简短的缩写，对应关系如下：

**班级代码 班级**

1 软件工程20.1班

2 软件工程20.2班

…

压缩包原则上只能提交一次。若提交后发现有重大缺陷，确实需要修改的，可重新提交，但提交时间按最后一次提交为准(关系到提前提交压缩包奖励分项，及延迟提交压缩包扣除分项)。

**以上提交规定务请严格执行，如果出现命名格式等问题，将酌情扣减成绩，问题严重的提交将不被接受。**

拷贝他人代码或实验报告属于学术诚信问题，一经发现将严肃处理。

五．分值计算

1．每个Lab的个人得分

Lab1的分数完全由实验报告的撰写情况决定。

Lab2-7每个实验演示验收通过了所有要求实现的功能且提交了代码的即得30分。其余70分看实验报告写作情况及提交的代码质量。

每队每个Lab排名第二及以后的同学，得分比排名第一的得分略少一点。

2．个人总分计算

个人实验总分 = ( Lab1 + Lab2 + Lab3 + Lab4 + Lab5 + Lab6 \* k6 + Lab7 \* k7

+ 提前提交压缩包奖励分

- 延迟提交压缩包扣除分 ) / d

个人总分 = 个人理论课考勤分 \* 0.1 + 个人实验总分 \* 0.9

其中：

k6 = 0.5。

k7 = 0.3。

提前提交压缩包奖励分、延迟提交压缩包扣除分，均在15分左右。

压缩包原则上只能提交一次。若提交后发现有重大缺陷，确实需要修改的，可重新提交，但提交时间按最后一次提交为准。

d <= 5.8。

理论课考勤分满分100分。

注：

实验开始后，会公布两个时间点：提前验收并提交压缩包时间点 A，及正常验收并提交压缩包时间点 D。

在A点之前，通过演示验收，并完成压缩包上传的，组内所有同学会得到提前提交压缩包奖励分。

在D点之后，才完成压缩包上传的(不管通过演示验收的日期)，组内所有同学将会被扣除延迟提交压缩包扣除分。

在A点与D点之间，完成压缩包上传的(不管通过演示验收的日期)，不加分，也不扣分。

必须先通过所有实验的演示验收，才能上传提交压缩包，且压缩包内容要与演示验收时的代码一致，否则按无效上传处理。