山东大学 软件 学院

**操作系统课程设计** 实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：计软20.5 |
| 实验编号：Lab2 | | | |
| 实验题目：具有优先级的线程调度 | | | |
| 实验学时：4 | | 实验日期：2022.11.05 | |
| 实验目的：  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)。 | | | |
| 硬件环境：  HUAWEI matebook14 2020笔记本  Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz CPU  8GB内存  512GB SSD | | | |
| 软件环境：  宿主机：Windows 10 21H2 64位  虚拟机软件：VMware Workstation Pro 16.2.4 build-17966106  Linux：Ubuntu 14.04.6 LTS Desktop i386 (Trusty Tahr)  gcc/g++：(Ubuntu 4.8.4-2ubuntu1~14.04.4) 4.8.4  MIPS交叉编译器：gcc-2.8.1-mips.tar.gz  Nachos：Nachos-3.4-UALR-2022 | | | |
| 实验步骤与内容：   1. Nachos原有的线程调度策略：   执行中的程序的地址空间表现在存储它的内存空间中：  TEXT，从外存中装入的可执行文件的二进制代码  DATA，由全局的和静态变量组成的程序数据区，分为初始化的和未初始化的两部分，初始化的数据是从二进制可执行文件中装入的。  STACK：函数局部变量存储区，由函数的调用而增长，函数的返回而缩小  HEAP：动态分配变量存储区  寄存器集，如PC寄存器包含着可执行代码下一条指令地址，SP寄存器指向当前栈。    （图：类Thread）  其中，线程的状态有如下：    当线程改变时，相应statues也对应改变。  Nachos中，线程调度由定义在scheduler的类中由一个全局对象完成，这个类的方法提供了线程和进程的所有调度功能。当系统首次启动时，首先建立一个Scheduler类的全局实例对象的引用，由它负责完成线程或者进程的调度任务。    （图：Scheduler.h）  类中仅有一个私有对象指向list对象，list对象是放置READY状态的就绪队列。函数ReadyToRun将一个线程推入队列尾，同时函数FindNextToRun从队列返回出队线程的指针。Run函数更换下一个线程并设置状态为RUNNING。  Nachos线程调度时维护一个就绪线程队列，当一个线程可以占用处理机时，调用ReadyToRun方法将这个线程放入就绪线程队列，并把线程状态改为就绪态。FindNextToRun方法根据调度策略取出下一个应运行的线程并把这个线程从就绪队列中删除。  根据源代码，不难发现现有的线程调度策略为先进先出策略（FIFO）：  **readyList->Append((void \*)thread);**    （图：FindNextToRun方法）  随后，由于涉及到较多的代码操作，因此在32位系统中下载VSCode并进行后续操作：       1. 设计并实现具有静态优先级的非抢占式线程调度策略。   此处设置数字越大，优先级越低。  首先根据ReadyToRun方法判断出原有的调度方式为FIFO，随后开始修改相应代码：   1. 修改Thread类，为Thread类新增属性priority表示线程优先级。注意要给优先级一个默认值：     其中注释头有“jxy”字样的注释为本人做的注释。  随后为Thread类添加新的构造函数和getPriority方法获得优先级：      随后在.cc代码中实现新构造函数：     1. 修改Scheduler类，在ReadyToRun方法中修改调度方式为非抢占式优先级调度：   主要思想为在添加新线程时，调用list类中的插入排序函数，将此线程按照优先级大小插入到readyList中，调度时先调度优先级小的。  下方扩展了抢占式优先级调度，其主要思想为将线程加入到就绪队列前比较其与当前线程的优先级，如果其优先级大，则将当前线程利用Yield函数放入就绪队列，保存上下文后将优先级高的线程优先运行。  至此，实现了非抢占式和抢占式优先级调度。   1. 修改threadtest文件，进行测试：     主要修改为在文件中加入三个不同优先级的子进程，分别为1，2，3优先级，同时修改打印信息，打印优先级。利用make命令编译后运行：    可以发现，在父进程启动后，具有较高优先级的进程1，2先运行，随后低优先级的3和父进程（默认优先级99）后运行。   1. 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。   以线程调试模式运行，得到相应结果：    打印信息过长，因此只截取了片段。可以发现，上下文切换次数多余打印的行数，且主要出现在下列位置：    当线程2结束了最后一次打印后，它没有立刻结束，而是先切换到了线程1，因为线程1在结束最后一次循环后没有结束而是先然后线程1结束后，再切换为线程2，线程2结束。主要原因为线程2结束后，由于线程1优先级高，所以线程1优先运行。但是此时线程2没有需要运行的函数，所以可以先结束。  因此定位到先前代码**，发现在测试代码中每次for循环打印后立刻调用下一个进程，而不是判断是否此进程结束**。因此做如下修改进行判断：    再次运行，发现先前问题消失：     1. 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。   优先级调度的老化指的是随着时间的流逝，提高等待的线程的优先级。当涉及到Nachos中代码的实现时，具体的思路如下：  首先要在Thread类中添加方法setPriority（）可以修改线程优先级，随后设置全局方法，主要完成如下任务： 在系统启动后便开始计时，每隔一段时间（如10min）便提高一次等待队列中线程的优先级。当有新线程插入时，若有一个足够老的线程，那么它的优先级会随着时间的流逝提高到极高，因此会优先执行老线程，而不会导致有“饿死”的现象发生。 | | | |
| 结论分析与体会：  通过本次实验，我对线程、线程调度以及优先级调度方法及其具体实现有了更加深入的了解，同时也对操作系统内部代码结构以及调用方式、数据流动方式有了深入的认识，同时我也意识到操作系统是一个复杂且有序运行的系统，只有通过缜密的设计和充分的考虑才能设计出一款优秀的高效的操作系统。  在对问题的解决上，我学会了熟练运用调试工具，并且学会了在Linux中对VSCode的安装与使用，取得了较大的进步。 | | | |