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： 实验一Context Switch in Nachos | | | |
| 实验学时：4 | | 实验日期： 2018/10/9 | |
| 实验目的：  understand how context switch is realized in Nachos by tracing the Nachos test program in both C++ and the machine code levels. | | | |
| 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：    通过分析和查看源码，可知程序的执行情况：   1. 首先执行main.cc中的main函数，进行主线程main的创建和初始化。 2. 调用threadtest.cc中的ThreadTest函数 3. ThreadTest函数执行Fork函数创建新进程forked thread并执行SimpleThread函数 4. SimpleThread函数循环五次，每一次循环调用Yield函数让出CPU给另一线程执行。所以程序输出为主线程和forked thread交替输出。 5. 具体执行情况如下：   9[FR8RV`3ZJN{X~B`W0MT%0  **问题一：**  trace the execution of Nachos and observe the executions of  (a) context switch function SWITCH()  (b) function ThreadRoot()  using gdb and   1. 进入../threads/通过make指令进行编译生成nachos执行文件 2. 输入gdb nachos 指令启动debug模式逐条执行nachos 3. 输入（gdb）b SWITCH 和 (gdb) b ThreadRoot 命令为SWITCH和ThreadRoot方法添加断点 4. 输入（gdb）run 命令运行 5. 输入 （gdb）c ，（gdb）ni,（gdb）s等命令查看运行情况     具体执行情况如下：  mmexport1539316700197  **问题二**  What are the addresses of the following functions in your Nachos:  i. InterruptEnable()  ii. SimpleThread()  iii. ThreadFinish()  iv. ThreadRoot()  and describe how did you find them.  通过gdb print 命令,可以打印出每个方法的地址。  ZU)D2U@MJDHQV]6SL)4OZ(H  四个方法的地址分别为：  InterruptEnable 0x804a2dc  SimpleThread: 0x804a495  ThreadFinish: 0x804a2c2  ThreadRoot : 0x804bad4  **问题三**  What are the addresses of the thread objects for  i. the main thread of the Nachos  ii. the forked thread created by the main thread  and describe how did you find them.   1. 通过分析源码可以分析出，main thread 是main函数调用的Initialize()函数创建并初始化的。Forked thread 是在ThreadTest()函数中通过Fork()函数创建并初始化的 2. 所以通过gdb b 为Initialize()函数和ThreadTest()函数设置断点用gdb n命令逐行执行。 3. 执行到线程创建方法后，调用print方法输出线程地址 4. 具体执行情况如下：   **main thread:**  mmexport1539316710951  Initialize函数执行到 currentThread = new Thread(“main”);创建了主线程。  main thread的地址为0x8054af0  **forked thread：**  **mmexport1539316694319**  ThreadTest函数执行到Thread \*t = new Thread(“forked thread”);时线程被创建。  forked thread线程地址为 0x8054b50  **问题4：**  When the main thread executes SWITCH() function for the first time, to what address the CPU returns when it executes the last instruction ret of SWITCH()? What location in the program that address is referred to?   1. SWITCH函数负责进程上下文的切换，用汇编语言实现。保存之前Thread的寄存器信息，将新的Thread加载到寄存器中。 2. 可以通过（gdb）layout asm 指令查看SWITCH汇编代码具体执行情况 3. 并不断调用gdb ni 命令逐条执行。 4. 可以通过（gdb）i r 命令输出寄存器的具体数值 5. 通过查看SWITCH源码和执行情况可知SWITCH的返回地址存储在eax寄存器中        1. 执行到 movl \_PC(%eax),%eax # restore return address into eax 输出寄存器eax的数值就是SWITCH方法的返回地址。   **具体执行情况如下：**        通过观察可知SWITCH执行到最后地址为0x0804bad4，这个地址指向ThreadRoot函数  **问题5：**  When the forked thread executes SWITCH() function for the first time, to what address the CPU returns when it executes the last instruction ret of SWITCH()? What location in the program that address is referred to?   1. 与问题四操作基本相同，程序第二次执行到SWITCH函数就是forked thread第一次执行SWITCH函数。 2. 具体执行情况如下：       可以得出forked thread第一次执行SWITCH函数的返回地址0x0804916f,这个地址指向Scheduler 的run函数。 | | | |
| 结论分析与体会：  通过实验基本掌握了Nachos线程切换的具体实现，SWITCH函数的汇编实现需要仔细的阅读并配合注释才能理解，老师上课的所教授的知识需要好好的理解和消化，需要在实战时不断地增强理解。  SWITCH函数的具体实现主要是保存上个线程的寄存器信息并加载新线程的数据到寄存器中。  因为对于c++，汇编语言的语法不够熟悉，遇到很多困难不过通过本次实验不断的熟悉了语言的语法，对于之后的实现有很大的帮助。 | | | |