山东大学 软件 学院

**操作系统** 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：2020级5班 |
| 实验编号：实验三 | | | |
| 实验题目：进程调度算法实验 | | | |
| 实验学时：4 | | 实验日期：2022年4月20日 | |
| 实验目的：  加深对进程调度概念的理解，体验进程调度机制的功能，了解 Linux 系统中进程 调度策略的使用方法。 练习进程调度算法的编程和调试技术。 | | | |
| 硬件环境：  宿主机：lntel(R) Core(TM) i5-10210U CPU @ 1.60G Hz 2 .1 1 G Hz  虚拟机：ubuntu-20.04.1-desktop-amd64 | | | |
| 软件环境：  宿主机：Windows 10 家庭中文版  虚拟机：Code::Blocks | | | |
| 实验步骤与内容：  1.分析并编写示例实验代码，并运行:    运行结果如下：  首先输入参数：10 5 -10 0 0 0 得到结果【注意要用sudo命令获得超级管理员权限】：    再输入 10 5 18 0 0 1 得到结果【注意要用sudo命令获得超级管理员权限】  ：    经验证，发现实验结果和实验大纲结果一致。  2.分析独立实验要求：  **设有两个并发执行的父子进程，不断循环输出各自进程号、优先数和调度策 略。进程初始调度策略均为系统默认策略和默认优先级。当某个进程收到 SIGINT 信号时会自动将其优先数加 1，收到 SIGCSTP 信号时会自动将其优先数减 1。请编 程实现以上功能。**  （SIGINT：程序终止(interrupt)信号, 在用户键入INTR字符(通常是Ctrl-C)时发出，用于通知前台进程组终止进程。  SIGTSTP：停止进程的运行, 但该信号可以被处理和忽略. 用户键入SUSP字符时(通常是Ctrl-Z)发出这个信号）  编写独立实验代码：    输入指令：gcc main.c sudo ./a.out  执行，并在执行过程通过ctrl-c,ctrl-z不断键入信号 | | | |
| 结论分析与体会：  【对大纲代码的理解】  初始化阶段：  创建变量i用于循环等；创建数组pid用于记录需要创建的三个子进程进程号；创建sched\_param结构数组p用于存储调度参数结构，主要包含调度优先数。同时创建数组pol、pri存储调度策略以及调度优先级。  运行阶段：  首先是三次for循环：  创建子进程，并将子进程号赋予给pid[i]；如果创建成功，则判断是否处于父进程阶段（pid[i]>0），如果是则做如下事情：  由于输入的第1，2，3个参数（存到了argv[]数组中）是进程的优先级，输入的第4，5，6个参数是进程的调度策略（argc>i+4），所以：  如果输入了第4/5/6位，则把它存到pol中作为对应子进程的调度策略，如果没有输入则默认是SCHED\_OUTER调度；如果输入了第1/2/3位则讲对应的进程优先级存入，如果没有输入则默认优先级为10；  随后将刚刚读到的调度策略和调度优先级利用sched\_setscheduler（） 和 setporiority（）方法设置对应子进程的调度策略和优先级。  如果是子进程（pid=0），则进入else判断语句分支：  首先睡眠三秒，然后进入for循环：进行十次输出，把子进程的pid、第几次输出、优先级以及一个cal方法输出（每隔一秒打印一次）。  每个子进程做如上几件事。  对于父进程，最后循环输出了三个子进程的进程号以及调度策略随后先行退出。  分析结果：  如果输入是sudo ./a.out 10 5 -10 0 0 0  说明三个子进程的优先级分别为10，5，-10，并且三个进程的调度策略皆为0，即默认的分时调度策略；因此在输出时，在工作量相同时，优先级高的（第三个）会更早地输出。  如果输入是 sudo ./a.out 10 5 18 0 0 1  说明三个子进程的优先级分别为10，5，18，但三个进程的调度策略分别为分时、分时和FIFO，可以看出虽然 11308 进程其优先数最大，但由于其调度策略为先进先出，因此总是首先得到调度。  【独立实验分析】  在实验开始设置四个函数和两个变量：  其中child变量存储子进程优先级，father变量存储父进程优先级；其中fathersigint和fathersigstp处理父进程信号，如果是中断信号则让father++并给予设置，如果是stp信号则father—并基于设置；childsigint和childsigstop同理。  随后进入主函数，通过fork创建子进程，在子进程和父进程的判断语句中分别检测信号并设置相应的函数，随后开始50次的循环，不断播报进程的pid、优先级以及调度策略。  【结论分析】  Linux 调度程序是抢占的、基于优先级的算法， 具有两个独立的优先级范围： 从0 、99的real-time 范围和从100 ～ 140 的nice 范围。这两个范围映射到全局优先级， 其中数值越低表明优先级越高。  一个可运行的任务被认为适合在CPU 上执行， 只要它在其时间片中具有剩余的时间。  当任务耗尽其时间片后， 它被认为是到期了， 不适合再执行， 直到所有其他任务都耗尽了它们的时间片。内核在运行队列数据结构中维护所有可运行任务的列表。由于对SMP 的支持， 每个处理器维护它自己的运行队列， 并独立地调度它自己。每个运行队列包括两个优先级队列一一活动的和到期的。活动队列包括所有在其时间片中尚有剩余时的任务，而到期队列包括所有己到期的任务。每个优先级队列都有一个根据其优先级索引的任务列表，调度程序从活动队列中选择最高优先级的任务来在CPU 上执行。对于多处理器系统，这意味着每个处理器从其自己的运行队列中调度最高优先级的任务。当所有任务都耗尽其时间片（即活动队列为空）时，两个优先级队列相互交换，到期队列变为活动队列，反之亦然。 | | | |