华南理工大学

《操作系统》课程实验报告

实验题目： 实验一：进程和线程的创建

姓名： 梁馨元 学号： 201730613410

班级： 17计科1班 组别： 无

合作者： 无

指导教师： 钟竞辉

|  |
| --- |
| **实验概述** |
| 【实验目的及要求】  实验目的：  1.进程和线程的创建  实验要求：  1．在linux下编写一个应用程序，命名为an\_ch2\_1b。这个程序不断地输出如下行：  Those output come from child,[系统时间]  另外写一个应用程序，命名为an\_ch2\_1a。这个程序创建一个子进程，执行an\_ch2\_1b。这个程序不断地输出如下行：  Those output come from parent,[系统时间]  观察程序运行的结果，并对看到的现象进行解释  2．在linux环境下编写一个控制台应用程序，程序中有一个共享的整型变量shared\_var，初始值为0；创建一个线程并使其立即与主线程并发执行。新创建的线程与主线程均不断地循环，并输出shared\_var 的值。主线程在循环中不断地对shared\_var 进行加1操作，即每次循环shared\_var 被加1；而新创建的线程则不断地对shared\_var 进行减1 操作，即每次循环shared\_var 被减1。观察程序运行的结果，并对你看到的现象进行解释。  3．提交源代码以及实验报告。  【实验环境】  操作系统：Linux |
| **实验内容** |
| 【实验过程】  一、进程的创建  1. 步骤和注意事项:   * 进程的创建使用fork函数，返回值为0则是子进程，否则返返回子进程pid * exec系列有很多个，本实验选择使用execv(path, args)：第一个参数是可执行文件的路劲，第二个是传入的参数 * 注意，exec系列的args部分需要以NULL结尾 * 此外，由于本实验要求文件需要按类型分别放入bin和src文件夹，执行时只能在一级目录执行（不能进入bin再执行），否则会因路劲错误而不能执行。 * 使用time.h下的函数得到系统时间，具体方法见代码 * 使用sleep可以让效果更明显一点   2. 实验结果和分析  1）an\_ch2\_1b 结果（部分截图）  2）an\_ch2\_1a结果    由上图可见，第一个程序成功打印系统时间，而第二个程序交替执行父子进程，这是因为进程的执行是交替的(当然也存在两个进程同时打印)。每次的结果也略有不同（可能是父进程先打印，也可能是子进程先打印）。这是因为系统fork之后得到的pid可能是子进程也可能是父进程。  二、线程的创建  1.步骤和注意事项   * pthread\_create函数创建线程 * pthread\_create通过传入function pointer告诉线程执行什么函数 * 注意，pthread\_create也可以给线程执行的函数传入参数，但是必须以(void \*)类型传入，而后在threadfunction中通过类型转换再恢复参数原来的类型 * pthread\_exit函数退出线程 * pthread\_join函数让主线程等待它创建的线程结束 * 为了更好得体现线程的工作，尝试多创建了几个线程   2.实验结果和分析  1）lab结果    图1    图2  2）lab\_1结果    图3    图4  由上图可见，所有线程都是分享同一个变量的（shared\_var的地址不变），这是因为shared\_var作为local var是被所有线程的分享的。除了local var以外，放在heap的变量（比如用malloc创建的变量）也是由所有线程共享。  此外，线程的执行也是交替进行的，和进程一样。图1和图2的结果差别只是因为后者使main和child线程的创建时间几乎一致。于是两个线程基本只能在0,-1,1之间徘徊了。  图3，看上去shared\_var的变动不是连贯的，也就是中途会有从76到-1的跃变。这是因为线程在对shared\_var做加减法并写回内存和从内存获得变量并打印出来有时间差。如果线程几乎都是同时创建同时执行函数，那结果就会有“跃变“。修改了创建线程的时间间隔之后问题就没有了（如图4所示）另外仔细看图4的结果，线程可能同时读取到一个shared\_var值，也就说明可能有幻读、脏读等情况。那是因为没有对共有变量进行加锁使得线程互斥。 |
| **小结** |
| 通过本实验，学会了创建进程和线程。实验的过程也颇有心得。  首先是第一个实验“使程序不断打印时间”。一开始想当然得使用while(1)死循环，并没有大碍，可以ctrl+c终止进程。但第二部需要用父进程执行子进程的时候，发现直接在终端使用ctrl+c并不能终止子进程，然后变成了真正的死循环，只能通过ps -aux或者top找回这个脱缰之马，并用kill杀死进程。这是因为父进程先于子进程退出之后，子进程就成了孤儿进程，会把init进程当做父进程。认贼作父的后果就是不能简单使用ctrl+c结束进程了。所以为了让后续实验省心，就让函数打印100次就结束了。  然后在使用exec函数的时候，一开始也不能让子进程执行。后来使用getcwd函数获得当前工作目录，醒悟到是path没有给正确的问题。后来坚持在一级目录调用bin的可执行文件，就没有问题了。  接着是第二个实验“创建线程”，遇到了pthread\_\*系列函数undefined reference的编译器报错。这是因为需要在编译时加上-pthread参数。醒悟到这点之后，我感觉每次编译的过程其实挺烦人的，就写了一个Makefile，在CFLAGS把-pthread加上，一劳永逸。  最后一点心得：man is your friend。不知道函数怎么用，man一下就知道，比google一下还快：） |
| **指导教师评语及成绩** |
| 评语：  成绩：           指导教师签名：                                                 批阅日期： |