**深 圳 大 学 实 验 报 告**

**课程名称： 操作系统**

**实验项目名称：实验一 并发程序设计**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 罗秋明**

**报告人：谭嘉豪 学号： 2020152087 班级： 数计班**

**实验时间： 2023年3月9日**

**实验报告提交时间： 2023年3月23日**

**教务部制**

|  |
| --- |
| **一、实验目的与要求**   1. **加深对进程的创建、运行、撤销过程的直观认识；** 2. **掌握通过操作系统的用户接口（命令行和系统函数）控制进程状态的方法；** 3. **了解多进程在多核处理机上的并发执行过程；**   **二、实验内容**   * 1. **可以使用Linux或其它Unix类操作系统；**   2. **学习该操作系统提供的命令行启动、撤销进程的方法；**   3. **学习该操作系统提供的系统调用接口（借助于库函数的形式间接调用）启动和撤销进程；**   4. **利用该操作系统提供的工具观测这些程序的并发执行过程以及状态转换过程。**   **三、实验步骤及说明**   1. **学习top、ps、pstree和kill命令的使用** 2. **top命令实时监控系统进程状态**   **命令格式：top [选项]**  **选项说明：**   * **-d 秒数：指定top命令每隔几秒更新。默认为3秒。** * **-i：使top命令不显示任何闲置或僵死进程。** * **-p：通过指定监控进程ID来仅仅监控某个进程的状态。**   **操作说明：**   * **P：以CPU使用率排序（默认）。** * **M：以内存使用率排序。** * **N：以PID排序。** * **Q：退出top。**   **查询字段解释，以下以实例说明：**  **第一行为任务队列信息，其中：**   * **23:30:57表示系统当前时间。** * **up 11 min表示当前系统运行时间为11分钟。** * **1 user表示当前登录了1个用户。** * **load average: 0.00，0.11，0.14表示系统在之前 1 分钟，5 分钟，15 分钟的平均负载。一般认为小于 1 时，负载较小。如果大于1，系统已经超出负荷。**     **第二行为进程信息，其中：**   * **Tasks：279 total表示系统进程总数为279。** * **1 running表示正在运行的进程数为1。** * **278 sleep表示睡眠的进行数为278。** * **0 stopped表示正在停止的进程数为0。** * **0 zombie表示僵尸进程数为0。**   **第三、四、五行分别为CPU信息、物理内存信息、交换分区信息。**   1. **Ps命令查看系统当前进程状态**   **命令格式：ps [选项]**  **选项说明：**   * **a：列出带有终端的所有用户的进程。** * **x：列出当前用户的所有进程。** * **u：面向用户友好的显示风格。** * **-e：列出所有进程。** * **-u：列出某个用户关联的所有进程。** * **-f：显示完整格式的进程列表。**      1. **Pstree查看系统进程树**   **命令格式：pstree [选项]**  **选项说明：**   * **-p：显示进程的PID。** * **-u：显示进程的所属用户。**      1. **Kill命令终止进程**   **命令格式：kill [选项] 进程号 或 killall 进程名称**  **选项说明：**   * **-9：强迫进程立即停止。**  1. **编写如下一段简单的C程序代码，创建子进程，并使用上述命令进行查看。**   **对上述C程序代码进行编译并且在后台执行后，使用ps命令查看进程信息。**  **可以发现，共创建了4个进程，父进程2921创建了两个子进程，分别是2922、2923，其中2922又创建了1个子进程2924。**  **使用pstree -p [进程号]命令可以更加直观地查看系统进程树的情况。**  **通过使用kill命令将进程终止，再次使用ps命令查看，发现自行创建的进程已经停止。**  **再次运行C程序代码，进程号改变。**  **使用/proc/PID/maps命令，了解输出信息。**  **可以观察到对应线程ID的线程的虚拟栈空间、虚拟变量、虚拟动态共享变量、虚拟系统的地址。**  **使用/proc/PID/status命令，了解输出信息。**  **可以观察到对应线程ID的线程的一些信息，包括Name、PID、PPID等。**   1. **使用fork()创建子进程** 2. **父进程创建十个子进程**   **编写以下C程序代码，思路：循环10次，当前进程为子进程时，使用break语句跳出循环，不再创建新的子进程（指当前进程的子进程），当前进程为父进程时，继续创建新的子进程（指父进程的子进程）。**  **使用ps命令与/proc文件系统命令查看创建的进程的PID与PPID，其中3136是父进程，而3137-3146是3136的子进程。**  **使用pstree命令可以更加直观地观察进程树。**   1. **使用fork()函数创建嵌套子进程**   **编写以下C程序代码，思路：循环10次，当前进程为父进程时，使用break语句跳出循环，不再创建新的子进程（指当前进程的子进程），当前进程为子进程时，继续创建新的子进程（指父进程的子进程）。**  **使用ps命令与/proc文件系统命令查看创建的进程的PID与PPID。**  **使用/proc文件系统查看其中一些进程的关系。**  **使用pstree命令查看进程树。**     1. **使用fork()函数创建树型结构的进程**   **编写以下C程序代码，思路：每个父进程必须创建并且只能创建两个子进程，且迭代三次。**    **运行后，发现创建了15个进程，每个进程最多只有两个子进程，一个父进程，与预期相符合。**  **使用ps命令与/proc文件系统命令查看创建的进程的PID与PPID。**  **使用/proc文件系统查看部分进程。**  **使用pstree命令查看系统进程树。**   1. **孤儿进程与僵尸进程** 2. **孤儿进程**   **编写如下代码实现孤儿进程，思路：使用getchar()函数使子进程阻塞，父进程正常运行即可。**  **使用ps命令查看孤儿进程PID与其PPID，原孤儿进程2163的父进程为2162，后来可以看到孤儿进程2162并没有被init进程收养，反而是被1457进程收养了。**  **同时使用pstree命令查看进程树。**  **正常时：**  **父进程结束后：**  **使用pstree -p命令查看转移后的情况。**  **使用top -p pid命令查看进程1457的情况，可以发现进程1457是一个systemd进程，而非init进程，通过上网查阅资料得知，桌面版的Ubuntu采用的是孤儿进程被最近的祖先节点收养的方式。**   1. **僵尸进程**   **编写如下代码实现僵尸进程，思路：使用getchar()函数使父进程阻塞，子进程正常运行即可。**  **使用ps j命令查看僵尸进程PID与其PPID，可以观察到进程4054是进程4055的父进程，原两个进程都是正常的，后来子进程已经变为了僵尸进程。**     1. **创建多个线程，在各个线程中打印出堆栈变量的地址**   **通过查找pthread.h库的源代码发现可以通过以下函数得到为线程所设置的堆栈地址。**  **发现该函数需要一个pthread\_attr\_t类型的参数，因此查找pthread.h库的源代码以找到对应的函数获得该参数。**  **编写如下C程序代码，创建5个线程中，并在各个线程中打印出堆栈变量的地址。**  **运行并输出，可以观察到线程所属进程的PID，自身的TID，堆栈地址，堆栈大小8M = 8 \* 1024 \* 1024。**  **使用cat /proc/3915/maps观察到子线程的栈实际上是在主线程中的“堆区”的一部分，且每个子线程的栈空间默认为8M。**  **同时可以观察到主线程的栈空间地址则不同，两者在不同的空间分别存放。**   * 1. **创建相同数量的进程与线程，比较**   **编写如下代码，分别创建50个进程、50个线程，比较进程控制块开销的差异、内存vma描述符开销的差异。**  **首先比较进程控制块的差异。**  **在fork-5 运行前、运行中和结束后各执运行一次cat /proc/slabinfo |grep task\_struct，观察进程控制块的数量。**  **可以看出，当fork-5运行时，运行前后产生了约50个task\_struct。**  **在pthread-5 运行前、运行中和结束后各执运行一次cat /proc/slabinfo |grep task\_struct，观察进程控制块的数量。**  **可以看出，当pthread-5运行时，运行前后产生了约50个task\_struct。**  **结论：线程作为调度执行单位，进程控制块 PCB（task\_struct）还是需要的，这是最小资源的一部分，在这方面的资源开销，进程和线程都是一样的。**  **接下来，比较内存、vma描述符开销的差异。**  **在fork-5 运行前、运行中和结束后各执运行一次cat /proc/slabinfo |grep mm\_struct，观察mm\_struct的数量。**  **在pthread-5运行前、运行中和结束后各执运行一次cat /proc/slabinfo |grep mm\_struct，观察mm\_struct的数量。**  **结论：同一进程的线程间共享进程的内存空间，因此共用一个内存描述符 mm\_struct，创建n个进程需要n个mm\_struct，但是在进程的主线程存在的前提下创建n个线程则不需要新创建任何 mm\_struct。**  **在fork-5 运行前、运行中和结束后各执运行一次cat /proc/slabinfo |grep vm\_area\_struct，观察vm\_area\_struct的数量。**  **在pthread-5 运行前、运行中和结束后各执运行一次cat /proc/slabinfo |grep vm\_area\_struct，观察vm\_area\_struct的数量。**  **通过观察可以发现，当创建的进程与线程的数量i相同时，创建线程所需的vm\_area\_struct远少于创建进程。**  **结论：创建线程的开销远比创建进程小。**   * 1. **自行设计一个C语言小程序，完成最基本的shell角色**   **给出命令行提示符、能够逐次接受命令；对于命令分成三种，内部命令（实现help命令给出用法、exit命令退出shell）、外部命令（即磁盘上的可执行文件）以及无效命令（不是上述两种命令）。**  **思路：使用一个循环使shell一直运行，内部命令help打印信息与exit跳出循环即可。外部命令需要使用fork()函数，父进程作为新的执行指令的进程，且需要wait(NULL)语句等待子进程执行当前指令，需要使用到exec函数族，子进程执行完外部指令后立即结束。**  **依次测试三种命令。**  **四、感想及其他**  **通过本次实验，我学习了以下指令如top，ps，pstree，kill，/proc文件系统等命令的用法，并在正式实验中加以运用，了解了fork()函数的使用、僵死进程与孤儿进程，比较了相同数量下的进程与线程的创建的资源开销（线程的开销较小），通过/proc文件系统查看主线程与子线程的堆栈空间的位置差异。** |

深圳大学学生实验报告用纸

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2023年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。