山东大学 软件 学院

**操作系统** 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202100300063 | 姓名：李彦浩 | | 班级：软件菁英班 |
| 实验编号：实验*1* | | | |
| 实验题目：进程控制实验 | | | |
| 实验学时：2 | | 实验日期：2023.04.05*x* | |
| 实验目的：  加深对于进程并发执行概念的理解。实践并发进程的创建和控制方法。观察和 体验进程的动态特性。进一步理解进程生命期期间创建、变换、撤销状态变换的过 程。掌握进程控制的方法，了解父子进程间的控制和协作关系。练习 Linux 系统中 进程创建与控制有关的系统调用的编程和调试技术 | | | |
| 硬件环境：  ThinkPad 8核 Intel Core i5-8265U 1.60GHz 8G内存  山大云算力平台 devlinux 1核 1G内存 | | | |
| 软件环境：  山大云算力平台 devlinux | | | |
| 实验步骤与内容：  实验要求掌握进程生命周期中各状态的变换过程以及父子进程的控制协作关系。  因此首先要在父进程中创建子进程，然后将其挂起（子进程进入waiting状态），随后父进程通过信号（键盘或其他输入，这里采用的模拟键盘信号）  唤醒子进程并发执行。  随后分为两种情况。第一种是父进程并不等待子进程的结束自己先行结束；第二种情况是父进程等待子进程结束再结束执行。  在实验模型中，我们采用命令行参数数组的方式来区分以上的两种情况，如果在命令行中附加了执行时的参数，那么将采用第二种情况；否则，采用第一种情况。  数据结构上，使用pid存储子进程号；status存储子进程返回状态,args[]数组存储缺省命令（用于第一种情况，因为我们要让子进程继续执行，体现出父进程先行结束），SIGINT存放一个模拟键盘中断的函数。  具体算法实现上，分两部分。  第一部分，对于第一种情况，（即如果命令行中没有附加的参数）那么创建子进程后执行pause()函数，让它等待父进程的键盘中断信号。随后我们用一种手段来模拟用户在键盘上发出唤醒信号，即kill(pid,SIGINT)来唤醒子进程。随后父子进程并发执行。子进程执行args[]中的缺省命令，父进程先行结束执行。之后子进程成为“孤儿进程”，由初始化进程清理其退出状态。      第二部分，对于第二种情况，（即如果命令行中有附加的参数）父进程在创建子进程后等待子进程的执行，但同时子进程因为执行了pause()函数而进入waiting状态等待唤醒，此时两个进程都被挂起，我们可以从用ctrl+z和ps -l的方式进入后台查看进程状态。    发现两个进程确实如预测的结果一样，都被挂起了。这个时候无法通过被挂起的父进程模拟一个键盘中断信号来唤醒子进程，只能通过手动唤醒的方式。即fg,ctrl+c让父子进程返回前台并继续执行。  此时子进程接到了键盘唤醒的信号，继续执行命令行传递进来的参数，同时切换自己的状态为running。    而父进程也接到了键盘唤醒信号，父进程也开始执行，但执行到waitpid(pid,&status,0)，也即需要等待子进程执行结束，于是父进程又进入了等待状态。    最终子进程按照命令行传递的参数执行完毕，父进程等待状态结束进入执行状态。父进程执行结束后消亡，并清理父子进程的退出状态。  项目文件共有四个。链接编译c文件的Makefile文件，代码主体部分pctl.c文件，代码头文件（同时定义了键盘唤醒函数）pctl.h文件，以及源码最终编译的产物pctl.o文件。它们统一放置在算力平台实例下的home/usr文件夹下。  本次实验实际投入学时2h,主要时间均用于代码的编写、调试与执行上  实验采用两种方式进行测试，一种是命令行缺省参数，另一种是命令行附加参数。运行结果如下。  第一种：    第二种：    实验中我通过信号状态控制父子进程的running/waiting状态，从而达到了控制父子进程并发执行或串行执行的效果。信号可以是用户输入的（模拟键盘输入），或者是某一进程执行的返回值，（子进程执行完毕返回退出状态status）利用这二者可以分别控制父/子进程的状态与父进程的状态。 | | | |
| 结论分析与体会：  实验中遇到了一个问题，即，在命令行缺省参数执行pctl时，fork后父进程先执行，随后子进程不执行默认的命令(/bin/ls -a)。而是被挂起了。下图中子进程成为了孤儿进程，它的父进程本来应该时306，但此时显示父进程是1。也即init进程（初始化进程），并且由于执行了pause函数子进程一直被挂起。    这是因为在实际执行中，父进程率先执行，在子进程还没有执行pause时已经向子进程发送了SIGINT信号。解决的方式也比较简单，我们先让父进程沉睡一段时间，（这个时间要足够长，让子进程执行pause函数），然后再发送SIGINT信号即可。    实验结果在上一个部分中已经列出，与预测的结果完全一致，完全复现了设计目标。  代码部分上还可以做进一步的改进（功能附加），例如，可以在子进程得到唤醒后在while(1)里每时隔三秒（sleep(3)）执行/bin/ls -a（或-l）命令，直到得到下一个键盘信号退出执行，同时唤醒父进程继续执行。  实验中我通过信号状态控制父子进程的running/waiting状态，从而达到了控制父子进程并发执行或串行执行的效果。信号可以是用户输入的（模拟键盘输入），或者是某一进程执行的返回值，（子进程执行完毕返回退出状态status）利用这二者可以分别控制父/子进程的状态与父进程的状态。 | | | |