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：  独立实验要求参考示例程序建立并发进程的方法，编写一个父子协作进程，父进程创建一个子进程并控制它每隔3秒显示一次当前目录中的文件名列表。  首先要先分支创建子进程，调用库函数fork()即可，返回值为0代表目前为子进程；返回值大于0代表子进程号。    随后需要判断当前进程是父进程还是子进程。这需要用到pid变量，如果为父进程，则等待子进程执行完毕；如果是子进程，循环打印文件名列表，直到接到键盘中断信号，则执行完毕，消亡。  第一种情况，父进程：    第二种情况，子进程：    这里需要注意的是，子进程是每隔3秒打印一次文件名列表，因此每次执行完毕”bin/ls”命令后，需要sleep(3)。  另外需要注意的是，如果子进程中采用了示例程序中的execve(char\* filename,char\* args)函数，是无法做到循环打印的效果的。这是因为exec族函数会复制当前程序，保留进程号执行，一旦执行完毕则进程退出，这与题目中要求每隔3秒相悖，于是采用另一种system call的方式，即system(char\* args)函数。  最后，键盘中断信号控制子进程是否继续循环执行。    volatile关键字是可以确保running变量可以在其他线程中得到更改。  在执行实验代码部分上，我采用了山东大学云计算平台提供的linux实例，编写了Makefile,将代码上传至实例中的数据域中。（以下截图是编译后的产物，一开始只有Makefile,pctl.h,pctl.c）    在控制台中gmake，对代码进行编译。  然后在cd到这个文件夹下，执行./pctl。（静态图片里看不出来这是每隔三秒打印的，但我会上传一个录屏，那个里面就能看到了）    可以看到结果中，反复输出了Makefile pctl pctl.c …这些文件名。  同时,^c代表ctrl+C,键盘中断信号发出后，退出了子进程，随后父进程等待子进程结束，父进程输出子进程状态码后也退出了。  到这里整个独立实验要求的功能就都已经实现了。  需要注意的是，有些代码的编写方式是在循环中反复调用fork，然后调用execve函数，也就是每个子进程执行一遍ls命令就结束了，消亡。这样是有悖于题目要求中的“一个子进程”的，因此不能用这种方式编码。  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  实验采用两种测试方式。第一种是ls，第二种是ls -a。（把具体信息也打印出来了）  第一种：    第二种：    实验中我通过信号状态(running变量)和库函数waitpid(int pid,int status,int sig)控制父子进程的running/waiting状态，前者控制子进程的执行，后者控制父进程的执行，从而达到了控制父子进程并发执行或串行执行的效果。键盘中断信号是用户输入的（ctrl+C），控制是否退出子进程。 | | | |
| 结论分析与体会：  实验中遇到了一个问题，即，如果调用示例程序中的execve的话，会执行一次打印文件名列表后会直接退出子程序。查阅文档后，发现exec族函数会复制程序，保留进程号，执行后直接退出，不再返回原进程了。    之后在查阅库函数文档后，我换成了一个可以同时进行system call并且可以返回原进程的函数，system函数。    这样问题就解决了。  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。  代码优化上还有可以改进的地方，例如，我可以模仿示例程序中，增加缺省命令模块以及用户自定义函数模块，使用户可以自己附加命令行参数并执行，而非指定执行我给出的命令（“/bin/ls”和”/bin/ls -a”）  实验中我通过信号状态(running变量)和库函数waitpid(int pid,int status,int sig)控制父子进程的running/waiting状态，前者控制子进程的执行，后者控制父进程的执行，从而达到了控制父子进程并发执行或串行执行的效果。键盘中断信号是用户输入的（ctrl+C），控制是否退出子进程。 | | | |