

****

《实用操作系统》实验报告

（二）

**姓　　名 ：宋泽涛**

**学　　号 ：25120222201292**

**学 院 ：信息学院**

**专 业 ：软件工程**

**2024年 9 月**

# 实验目的

* 加强对进程概念的理解。
* 掌握进程的创建。
* 进一步了解并发执行的实质。

# 实验环境

* + PC + Linux Red Hat操作系统
  + GCC

# 实验内容

* + 熟悉linux环境的基本使用命令，熟悉使用vi、gdb、gcc等编程工具或软件
  + 利用fork()函数创建子进程。
  + 考察fork()函数创建的子进程与父进程之间的同名变量是否为临界资源。
  + 利用fork()函数编写一个程序，要求父进程创建两个子进程，父进程、子进程并发执行，输出并发执行的消息，如：

父进程正在执行…..

子进程1正在执行…..

子进程2正在执行…..

一段时间后，父进程杀死子进程，然后结束自己的进程，分别显示：

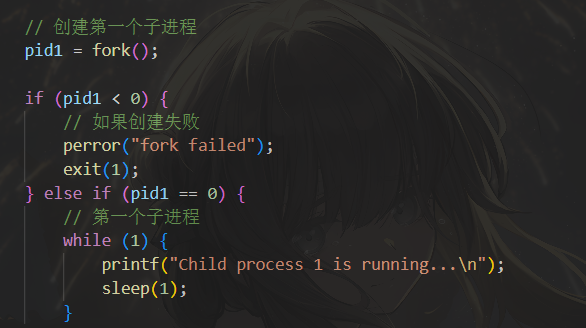
子进程1被父进程杀死

子进程2被父进程杀死

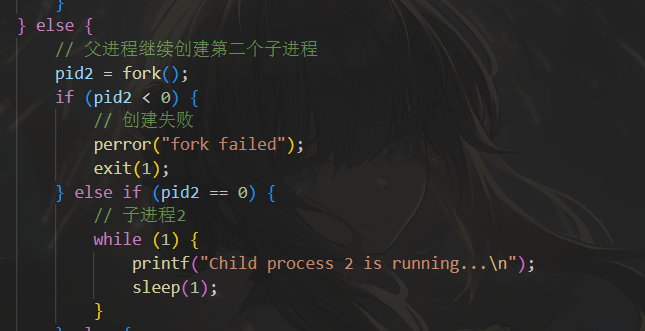
父进程结束

# 实验具体实现

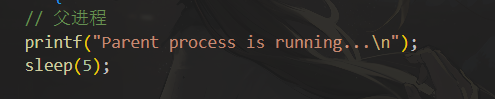
1. 首先编写如下c语言程序，在其中调用 fork() 函数创建第一个子进程



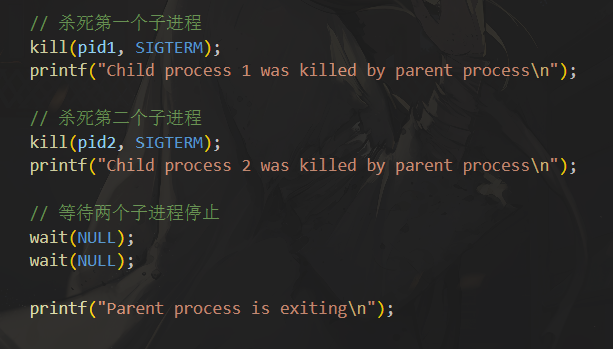
1. 创建第二个子进程



1. 父进程睡眠5s 保证两个子进程都运行一段时间



1. 然后父进程杀死两个子进程



1. **fork()函数创建的子进程与父进程之间的同名变量是否为临界资源？**

答：fork()函数创建的子进程与父进程之间的同名变量**不是临界资源**。原因是当fork()创建子进程时，子进程会复制父进程的整个地址空间，包括所有变量的值和内存状态，但这只是一个拷贝，子进程和父进程各自拥有自己独立的变量和内存空间。以上面的代码为例，调用 fork()函数之后，父进程和子进程各自拥有自己独立的内存空间，同名变量会被复制，而不是共享。pid1和pid2在父进程和子进程中是独立存在的，每个进程对这些变量的操作仅影响自己进程的内存空间，不会影响其他进程。

1. **利用fork()函数编写一个程序，要求父进程创建两个子进程，父进程、子进程并发执行，输出并发执行的消息，如：**

**父进程正在执行…..**

**子进程1正在执行…..**

**子进程2正在执行…..**

**一段时间后，父进程杀死子进程，然后结束自己的进程，分别显示：**

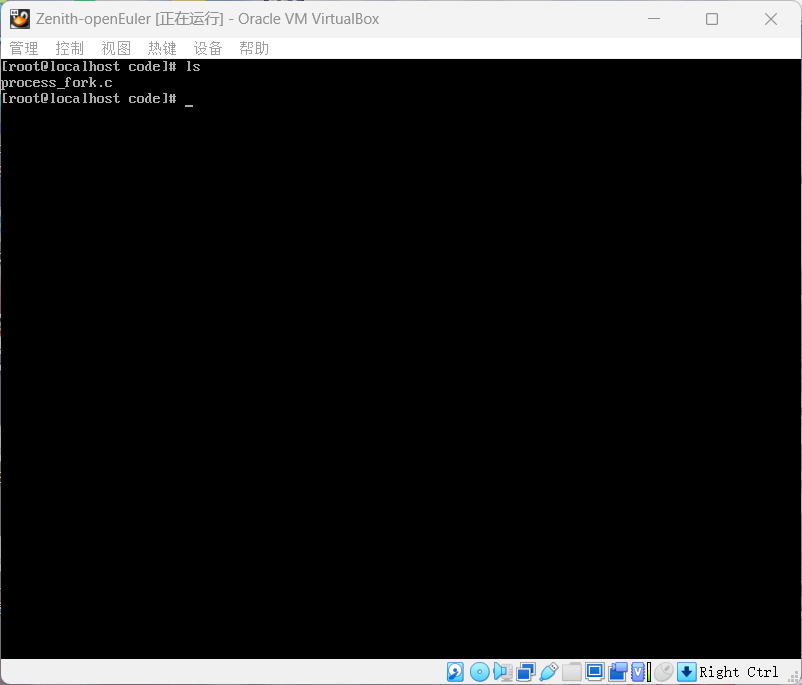
**子进程1被父进程杀死**

**子进程2被父进程杀死**

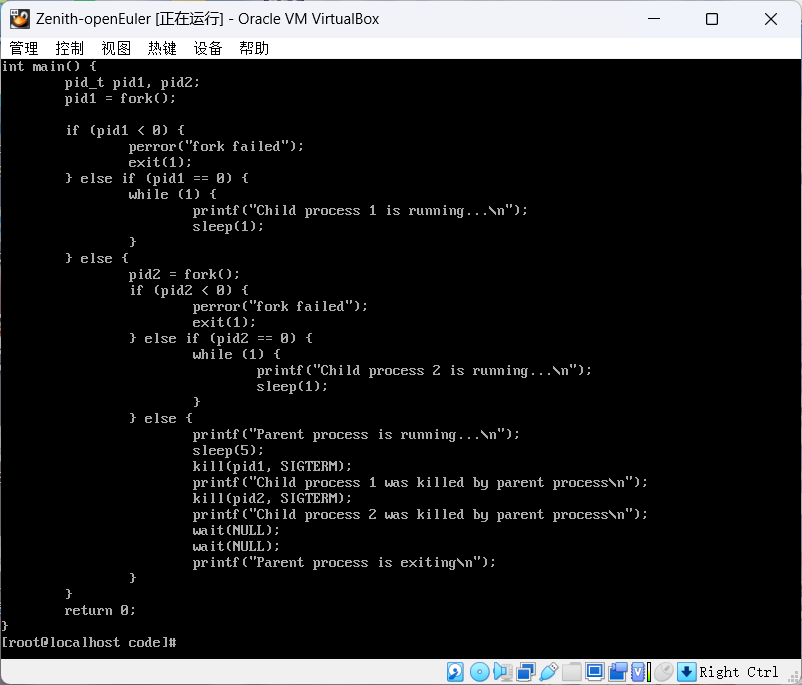
**父进程结束**

在Linux虚拟机中执行效果如下：

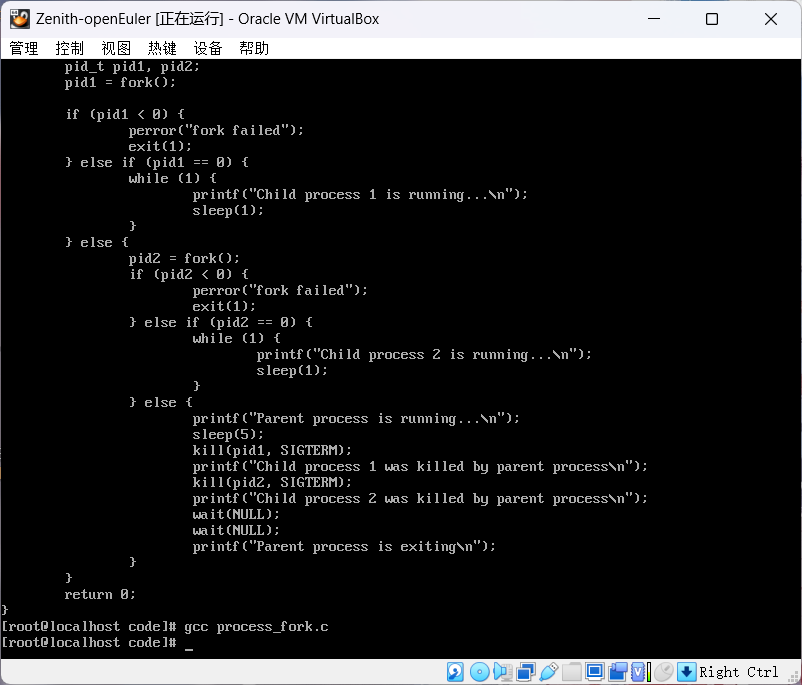
1. 定位到code文件夹



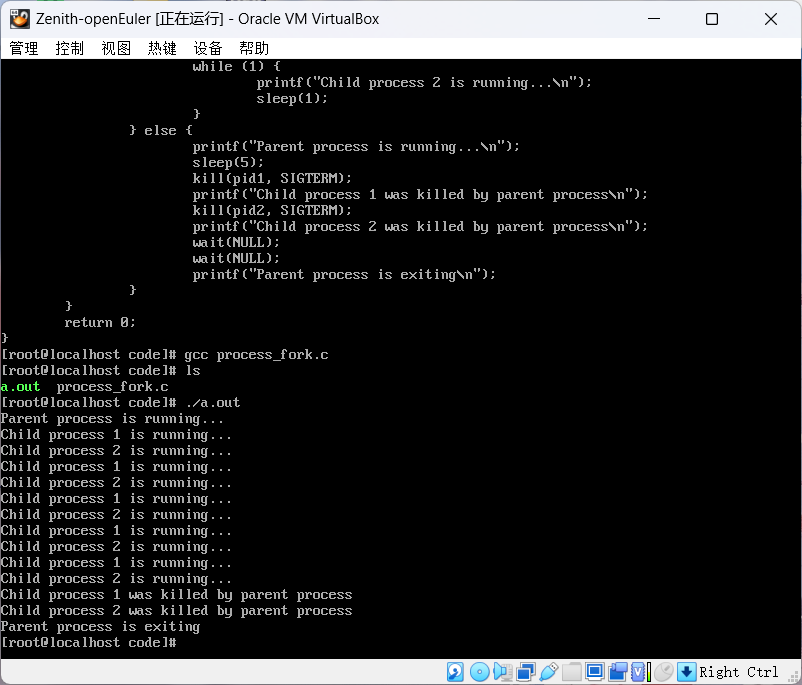
1. 事先已经将编写好的c代码通过vim存放到了code文件夹中，这里仅查看



1. 使用 gcc 命令生成可执行程序



1. 执行生成的可执行程序



# 实验分析与总结

1. 通过本次实验，进一步加深了对进程概念的理解。利用 fork() 函数创建的子进程和父进程是独立的，可以并发执行。父子进程在系统中获得了平等的调度机会，这展示了进程并发执行的特点。通过输出消息和延迟显示，能够明显观察到父进程与子进程的并行运行效果。
2. 通过实验体会到进程与程序的区别。虽然子进程在创建时是父进程的复制品，但它们是两个 独立 的进程，拥有不同的 PID 和独立的执行空间。父子进程可以在相同的代码基础上执行不同的操作，并且互相不会干扰。
3. 本次实验通过编写并运行 fork() 函数创建的父子进程程序，全面掌握了进程的创建、并发执行以及进程间信号通信的操作方法。通过深入理解进程的独立性和并发机制，增强了对操作系统中进程管理的认识