# 实验二 进程创建与并发

# 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…..

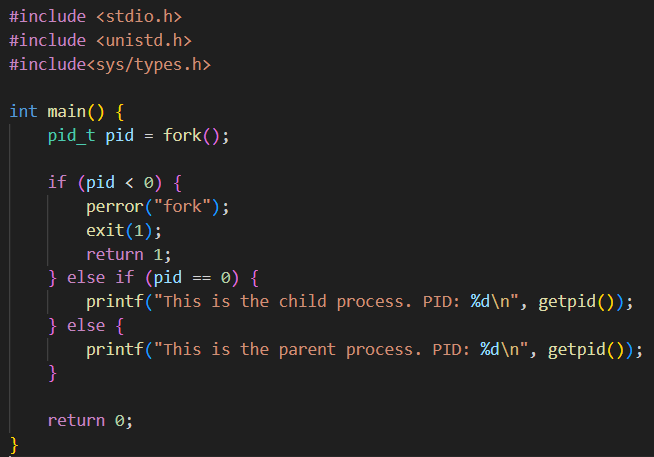
一段时间后，父进程杀死子进程，然后结束自己的进程，分别显示：

子进程1被父进程杀死

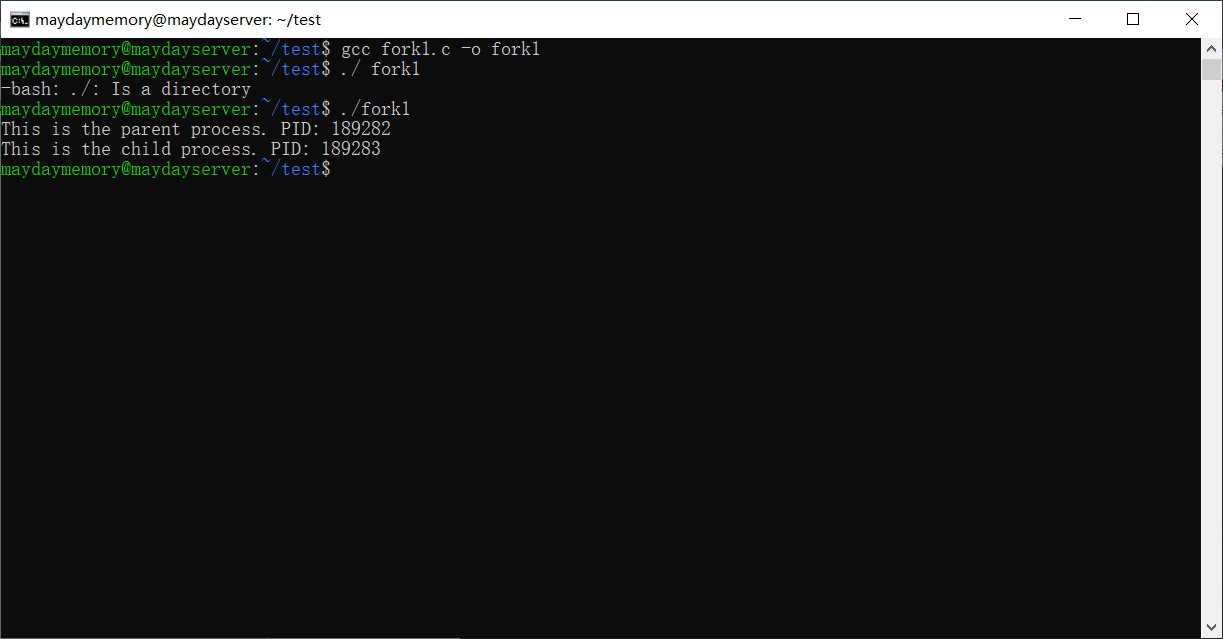
子进程2被父进程杀死

父进程结束

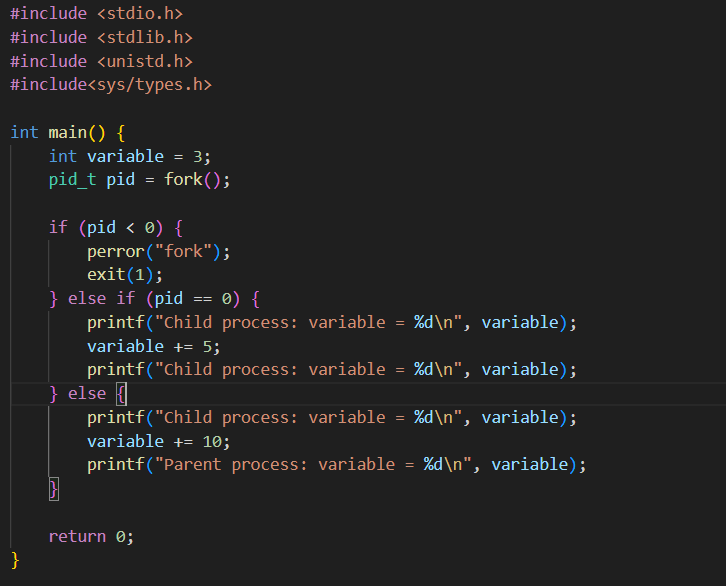
# 实验结果



利用fork()函数创建子进程。



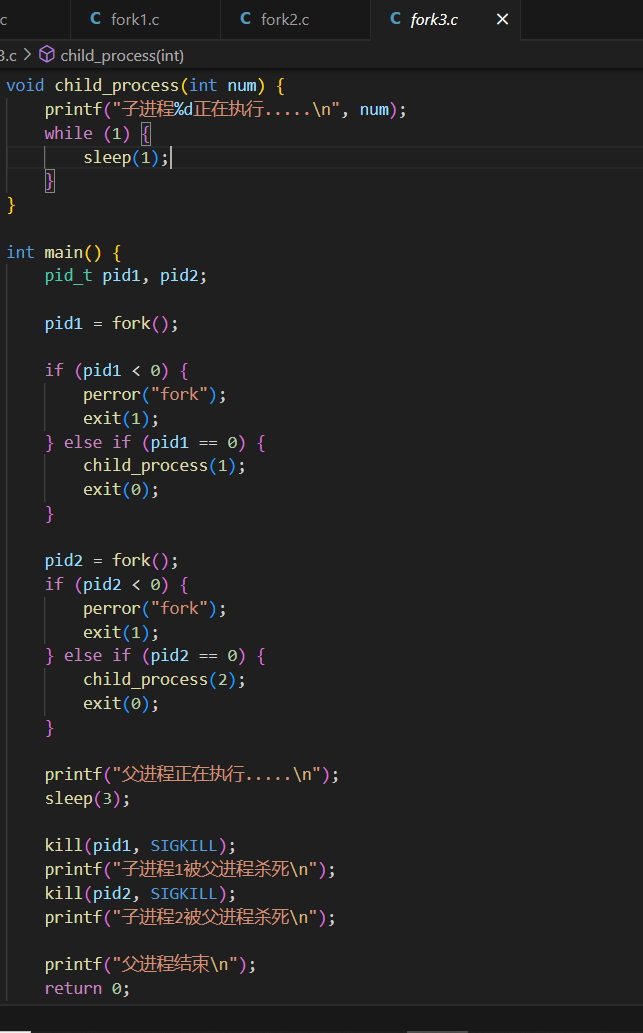
子进程调用fork（）返回0，父进程和子进程都输出自身的pid



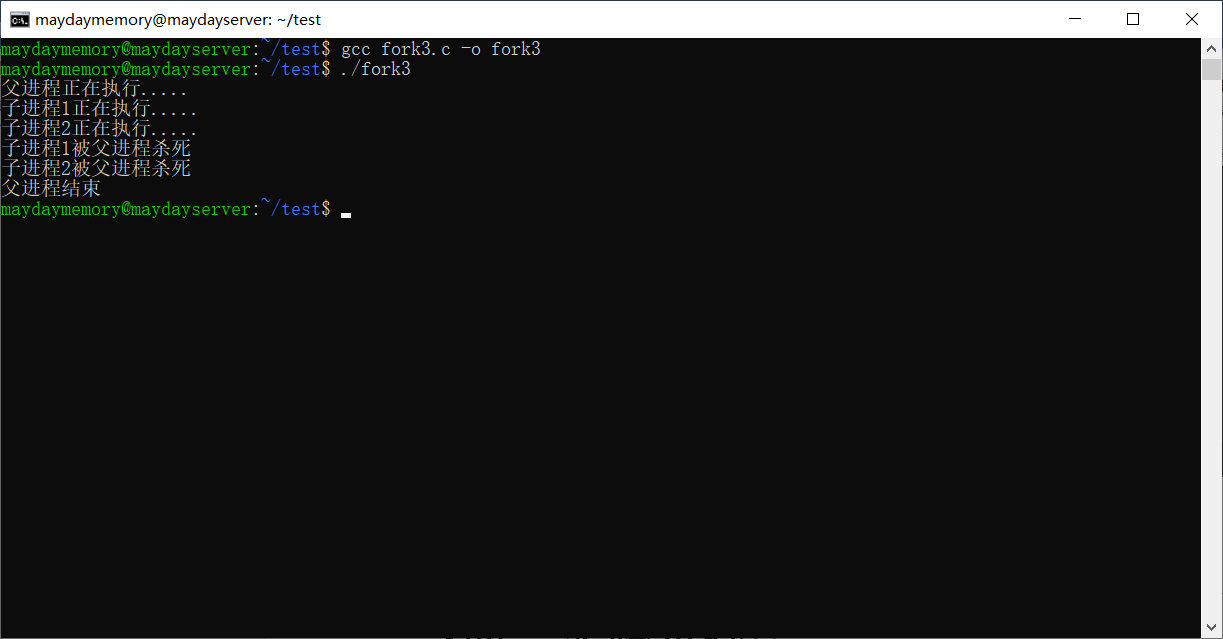
考察fork()函数创建的子进程与父进程之间的同名变量是否为临界资源。



子进程会复制父进程的变量，进程间的变量互不影响



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



进程并行运行，由父进程结束子进程

# 实验小结

通过本次实验，我研究了 fork() 函数在 Linux 环境下的应用，并验证了父子进程之间的变量独立性。成功地使用 fork() 函数创建了两个子进程，实现了父进程与子进程的并发执行，并通过输出各自的执行消息，清晰地观察到了多进程并发的效果。在实验中，我定义了一个变量，并在子进程中对其进行了修改，结果表明子进程对变量的修改并不会影响父进程中的变量，验证了每个进程拥有独立的内存空间，父子进程之间的同名变量并不是临界资源。父进程在等待一段时间后，成功地使用 kill() 函数终止了子进程，这帮助我理解了进程的终止和资源管理机制。