**实验二 进程的控制**

# 姓 名徐昊博 学 号21013134 成绩

实验时间2023年10月26日 指导教师(签名)

**（诚信声明：本实验报告内容，均由本人亲自上机完成。 签名：\_\_\_\_\_\_\_\_\_\_\_\_\_）**

一．实验目的

1.熟悉和理解进程和进程树的概念，掌握有关进程的管理机制

2.通过进程的创建、撤销和运行加深对进程并发执行的理解

3.明确进程与程序、并行与串行执行的区别

4.掌握用C 程序实现进程控制的方法

二．实验工具与设备

已安装 Linux 操作系统的计算机。

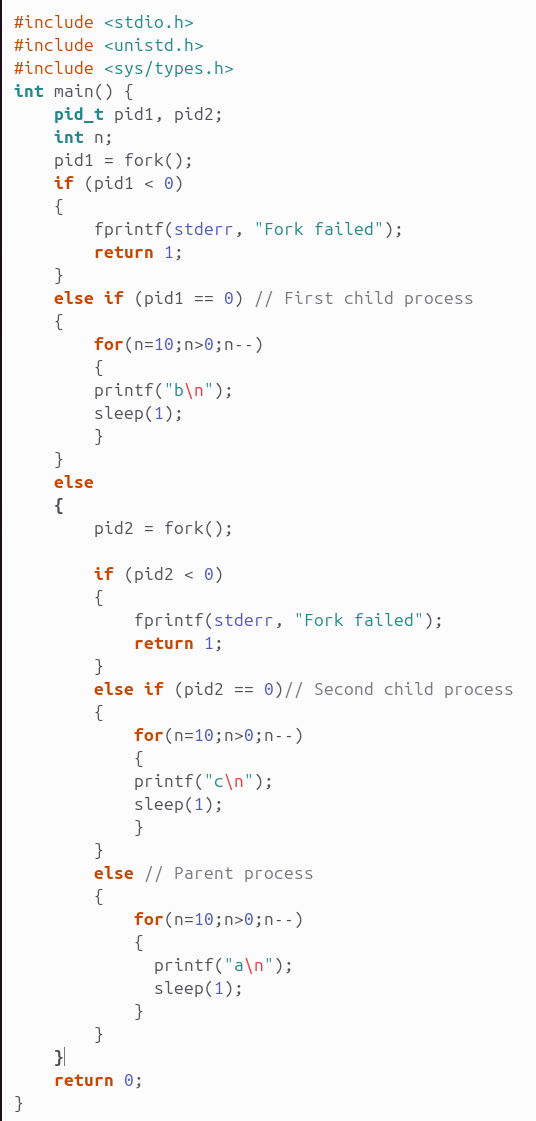
1. 实验内容

## 1. 了解系统调用 fork( ), exec 系列函数, exit( ), waitpid( )的功能和实现过程

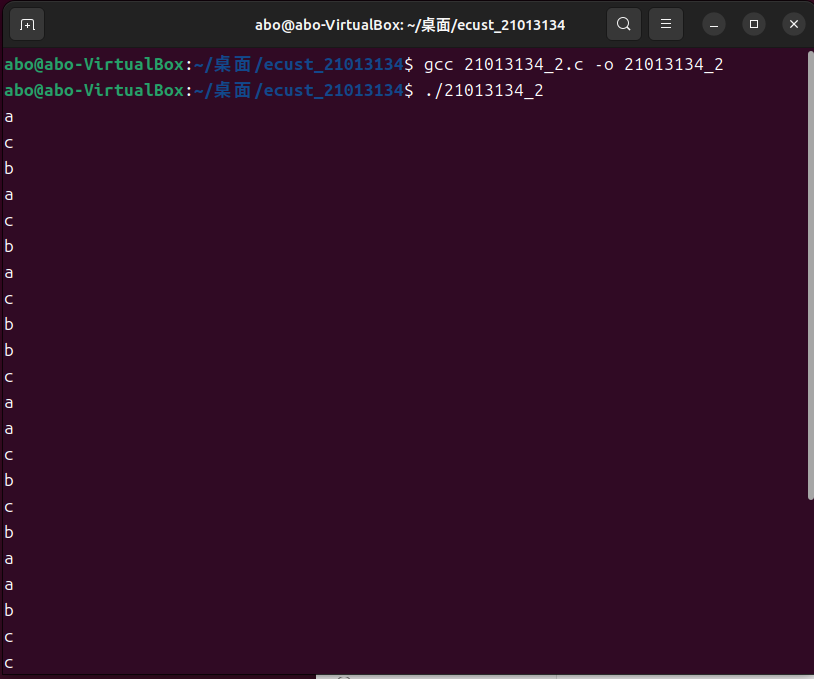
## 2．程序设计

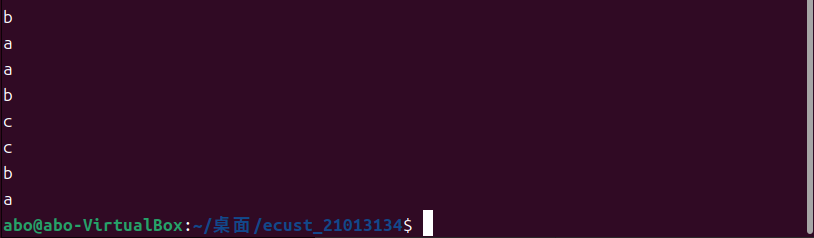
（1）编写一段程序，使用系统调用 fork( )创建两个子进程。当此程序运行时，在系统中有一个父进程和两个子进程活动。让每一个进程在屏幕上显示一个字符：父进程显示字符’a’；子进程分别显示字符’b’和’c’。观察屏幕上的显示结果，并分析原因。

实验代码如下：



根据题目要求，在上述代码中首先定义了两个进程变量pid1和pid2，然后使用fork（）命令新建了一个进程，当进程变量小于0时产生进程错误，当变量等于0时调用了子进程，第一个子进程要求输出1，为了研究进程之间的调度关系和资源管理，将该子进程重复10次，利用循环来实现，同时使用sleep（1）函数控制各个进程之间的输出顺序，在每次循环后都挂起1秒钟，当进程变量大于0时调用了pid1的父进程，按照题目要求在该父进程下再创建一个进程pid2，用同样的方法对pid2取值进行分析，当小于0的时候报错，等于零时调用第二个子进程输出c，为了研究进程之间的调度关系和资源管理，将该子进程重复10次，利用循环来实现，最后变量大于0时输出父进程内容，输出a，也同样重复10次，代码编写后进行编译运行结果如下：





可以看到程序在一开始输出的内容为acb。即先执行子进程再执行父进程，但随后出现了随机输出例如abc或者cba等现象，原因在于这三个进程是并行运行的，而Ubuntu操作系统调度这些进程的顺序是不确定的。操作系统可能会根据当前系统的负载、进程的优先级等因素来决定进程的调度顺序。因此，无法确定哪个进程会首先执行或者在何时执行。

代码中，每个进程都有一个循环，在循环内部会打印字符并休眠一秒。因为这些进程是并行运行的，所以它们的输出顺序可能会是随机的，具体取决于操作系统的调度决策。

因此，当观察到的输出现象是 "acb" 或 "cba" 或其他随机顺序时，这是因为进程调度的随机性导致的，操作系统会根据内部的调度算法来决定进程的执行顺序，而不是按照代码中的先后顺序执行。

（2）编写一段程序，使用系统调用 fork( )创建一个子进程。子进程通过系统调用 exec 系列函数调用命令 ls，调用 exit( )结束。而父进程则调用 waitpid( )等待子进程结束，并在子进程结束后显示子进程的标识符，然后正常结束。

代码如下所示：

# 

运行两次后结果如下：

# 

注意到显示子进程的标识符的时候运行两次前后PID值有变化，这是因为当程序运行时，操作系统会为每个进程分配一个唯一的进程ID（PID）。这个ID可以被用来标识和管理这个特定的进程。在上面的代码中，`fork()`系统调用会创建一个新的进程，它是调用进程（也就是父进程）的一个副本。子进程和父进程会继续执行`fork()`之后的代码。

由于每次运行程序时，操作系统都会为新的进程分配一个新的PID，所以即使代码完全相同，每次运行程序时，子进程的PID也会不同。这种分配PID的方式有助于操作系统跟踪和管理每个正在运行的进程。

# 思考题

## 怎样用 C 程序实现进程的控制？当首次调用新创建进程时，其入口在哪里？

## 当在C程序中调用`fork()`函数时，操作系统会创建一个新的子进程。在这个时刻，父进程和子进程各自有各自的地址空间、变量和文件描述符。`fork()`函数返回两次：对于父进程，返回值是新创建子进程的进程ID（PID），而对于子进程，返回值是0。通过检查`fork()`的返回值，我们可以确定代码是在父进程还是子进程中执行。

## 在父进程中，我们可以使用`wait()`或`waitpid()`来等待子进程的结束，并获取子进程的状态信息。这样可以确保父进程在子进程执行完毕之前不会继续执行其后的代码。

## 当首次调用新创建进程时，子进程会从`fork()`之后的位置开始执行。这意味着在`fork()`之后的代码会在父进程和子进程中各自执行一次。因此，我们可以在程序中根据`fork()`的返回值来区分父进程和子进程，从而执行不同的代码逻辑。

1. **系统调用 fork( )是如何创建进程的？系统调用 exit( )是如何终止一个进程的？**

## 系统调用fork()用于创建一个新的进程。它的工作方式如下：

## 1. 当调用fork()时，操作系统会创建一个新的进程，该进程是调用进程（父进程）的一个副本。

## 2. 新的进程会继承父进程的大部分属性，包括内存布局、打开的文件描述符、信号处理程序等。但是，子进程会有自己独立的进程ID（PID）。

## 3. 子进程会从父进程的执行位置开始执行，然后父进程和子进程会并发执行，各自在自己的地址空间中运行。

## 4.fork()返回两次。在父进程中，返回新创建子进程的PID（正整数），而在子进程中，返回0。这使得父进程和子进程可以根据返回值来区分自己的角色。

## 系统调用exit()用于终止一个进程。它的工作方式如下：

## 1.当调用exit()时，进程会通知操作系统它的终止意图。

## 2.进程可以提供一个退出状态码，通常用来表示进程的执行状态。这个状态码可以被其他进程获取，以了解进程的退出状态。

## 3.操作系统会清理进程的资源，包括释放内存、关闭文件描述符等。

## 4.进程的退出状态信息会被保存，以供父进程使用wait()或waitpid()来获取。这样，父进程可以了解子进程的终止状态。

## 总之，fork() 创建一个新进程，而 exit() 终止一个进程，并允许进程传递退出状态信息。这些系统调用是操作系统中进程管理的核心部分。

1. **系统调用 exec 系列函数是如何更换进程的可执行代码的？**

## 系统调用 exec 系列函数允许进程将自己的地址空间替换为一个新的程序。这些函数包括 execve(),execl(),execle(),execlp(),execv(),execvp()等。它们的工作方式如下：

## 1.当调用exec()函数时，操作系统会加载一个新的程序文件到当前进程的地址空间中，取代原有的程序代码和数据。

## 2.新的程序文件通常是可执行文件，它可以是一个二进制可执行文件或者是一个可执行的脚本文件。

## 3. 被加载的新程序会开始执行，取代原有进程的执行。这意味着原来进程的代码、数据和堆栈都会被新程序的代码、数据和堆栈所取代。

## 4. 除了代码和数据，新程序还会继承原有进程的一些属性，比如文件描述符、信号处理程序等，但是它们可能会被新程序所覆盖或者重置。

## 5. 通常情况下,exec()函数不会返回，除非发生了错误。如果发生了错误，它会返回一个负值来表示错误的原因。

## 通过使用exec()系列函数，一个进程可以以一个全新的程序来替换自己的执行代码。这使得进程可以动态地加载不同的可执行文件，实现动态程序的切换和扩展。