练习04

在本练习中,您将使用 POSIX 线程 API (pthreads) 编写多线程程序。您还将了解执行顺序是如何难以预测的,并且每次执行都可能有所不同。这是我们要做的

当我们开始谈论竞争条件和同步时,需要理解。

在课程主页上,您会找到一个名为 交织.c .这
程序的主线程运行一个循环 100,000 次迭代,使用 write() 系统调用打印出换行符
每次迭代结束。

添加代码以在此循环的每次迭代中创建三个新线程,然后在打印之前将它们全部加入
新队。每个线程将使用不同的启动例程将不同的字符打印到标准输出。一个线程将
打印一个 a 然后一个 b 。你的线程应该使用 write() 系统调用而不是使用 C 标准输出库
打印每个字符;单独调用 write() 来打印每个字符。我们可以轻松地打印两个字符
一次调用 write(),但是进行两次调用将有助于说明我们为这些线程获得的执行顺序。

使用相同的技术,另一个线程将打印“c”然后“d”,第三个线程将打印“e”然后“f”。后
打印它的两个字符,每个线程将终止。

我们使用单独调用 write() 来打印每个字符,以便我们可以看到不同执行顺序的证据
线程之间。在 main() 循环的每次迭代中,我们应该得到一个包含六个字符的输出行,但是
这些字符的顺序将告诉我们一些关于线程运行顺序的信息。例如,也许我们会得到
一行“abcdef”。这告诉我们第一个线程打印了它的两个字符,然后是第二个线程,然后是最后一个线程。
由于每个字符都是单独打印的,所以我们可以得到像“acebdf”这样的命令。这告诉我们每个线程都必须打印
它的第一个字符,然后每个人都必须打印它的第二个字符。

每个线程运行如此短的语句序列然后终止,您可能认为我们会
总是得到相同的输出。让我们看看我们实际得到了什么。编译程序然后按如下方式运行。什么时候
你编译时,一定要使用 -lpthread 标志进行链接,这样你就可以得到 pthreads API 中的函数。这个命令
获取程序的输出,对其进行排序,然后使用 uniq 命令丢弃重复的行。这应该给我们留下程序编写的每个唯一输出行的副本。

壳 $ ./交错 |排序 |独特的

检查你的输出,看看你得到了什么。当我在 EOS 上运行这个程序时,我得到了 12 个不同的执行命令
linux 机器,但是当我在我自己的双核机器上运行它时大约 50。更多的核心会给你更多的乐趣
结果。您可以从 CPU 调度程序中获得更多有趣的行为,甚至可以并行执行

线程。

完成后，上交已完成的 interleaving.c 程序的源代码。提交文件至名为 EX04 的成绩范围作业。