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*5* | | | |
| 实验题目：进程互斥实验 | | | |
| 实验学时：2 | | 实验日期：2023.05.18 | |
| 实验目的：  进一步研究和实践操作系统中关于并发进程同步与互斥操作的一些经典问题 的解法，加深对于非对称性互斥问题有关概念的理解。观察和体验非对称性互 斥问题的并发控制方法。进一步了解 Linux 系统中 IPC 进程同步工具的用法，训 练解决对该类问题的实际编程、调试和分析问题的能力。 | | | |
| 硬件环境：  ThinkPad 8核 Intel Core i5-8265U 1.60GHz 8G内存 | | | |
| 软件环境：  Vmware workstation;Ubuntu 20.04 | | | |
| 实验步骤与内容：  独立实验要求利用 linux 系统提供的 IPC 进程通信机制实验并实现理发店问题的一个解法。理发师问题是指：假设理发店的理发室中有 3 个理发椅子和 3 个理发师，有一个可容 纳 4 个顾客坐等理发的沙发。此外还有一间等候室，可容纳13 位顾客等候进入理发 室。顾客如果发现理发店中顾客已满（超过 20 人），就不进入理发店。 在理发店内，理发师一旦有空就为坐在沙发上等待时间最长的顾客理发，同时空 出的沙发让在等候室中等待时间最长的的顾客就坐。顾客理完发后，可向任何一位理 发师付款。但理发店只有一本现金登记册，在任一时刻只能记录一个顾客的付款。理 发师在没有顾客的时候就坐在理发椅子上睡眠。理发师的时间就用在理发、收款、睡眠上  仿照示例程序给出的代码，建立ipc函数库ipc.h和ipc.c，分别声明要用到的函数和实现这些函数，这一部分就是照搬示例程序。    注：ipc.c代码很长，这里截图就不放上来了，具体参照code文件夹下的ipc.c文件。  随后建立理发师barber和顾客customer文件。  代码目录结构如下。    首先需要实现理发师功能。实际上理发师问题的消息队列是一个三级缓存的形式，也就是说一个顾客可能会处在三种状态：第一是坐在理发椅上得到理发服务，第二是坐在沙发上等候理发，第三是坐在等候室等待理发。因此沙发和等候室都是需要一个消息队列来接受并阻塞顾客请求的。  声明变量wait\_request\_id创建等候室消息队列，声明变量sofa\_request\_id创建沙发消息队列。分别创建这两个消息队列的响应队列。沙发请求队列是指顾客从沙发向理发师请求理发；等候室消息队列是指顾客从等候室请求到沙发。    其次还需要对账本这个临界资源进行处理，同一时间只能有一个顾客访问账本，那么需要创建account\_sem信号量来表示账本使用的互斥。  其余创建共享内存和信号量的代码和示例程序完全一致，这里不赘述了。  之后，实验要求有三个理发师，每个理发师都可以为顾客理发、结账，这一功能我们可以用之前学习到的fork()函数来实现，我们可以fork()两个子进程，让他们和父进程组合在一起，表示三个理发师，每个进程实现的功能都是一样的。    随后，在每个进程中，我们要查询当前消息队列中是否有顾客从沙发发来的请求，（只需要查看沙发的消息队列即可，因为不可能有顾客直接从等候室来理发，一定会经过沙发），如果有，读取这条消息并向响应队列添加记录，理发师提供理发服务，随后锁定账本，结账，然后释放账本，完成一次理发。    然后需要实现顾客的功能。同样的，共享内存、消息队列、信号量的配置和声明在这里不加赘述了。  由于等候室和沙发都是有人数上限的，我们需要记录当前在等候室/沙发上的等候人数，并记录顾客的id方便控制台打印。Current\_id记录的是等候室等待最久的顾客的id。    随后顾客到来时需要根据等候情况决定分配到哪个等候区域或是离开。首先是沙发还有空位置的情况。    沙发有空位置还分成两种情况，一是等候室有顾客，那么将等候室的第一个顾客（也是等待时间最久的）挪到沙发上；如果没有顾客，新顾客可以直接到沙发上。    如果沙发已经没有空位置了，新顾客要到等候室去，如果等候室有空，进入等候室，如果没空，新顾客离开。这里注意要更新在等候室等候最久的顾客的id。    之后更新等候人数和等候最久的顾客的id。    在执行实验代码部分上，我采用了vmware workstation和Ubuntu20.04。将makefile文件和c文件放在了一个文件夹下。  在终端中gmake，对代码进行编译。    首先启动理发师进程。一开始没顾客，是阻塞的。    启动customer进程，开始工作。      可以看到由于理发师进程的执行周期比顾客进程的周期长，导致后续的顾客依次从沙发到等候室，再到等候室满顾客离开。  到这里整个独立实验要求的功能就都已经实现了。这一部分的详细动态演示在screenCap文件夹下的录屏文件中。  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  实验采用一种测试方式。执行理发师进程和顾客进程。  实验中我创建了4个并发的进程，三个为生产者，一个为消费者。进程互斥控制的功能采用c语言的ipc库编程实现。 | | | |
| 结论分析与体会：  实验中我遇到饿死现象，我用id控制顾客的身份标识，每新来一个顾客就自增1。但这样会导致后来的顾客的id覆盖了前一个顾客的id，然后后来的顾客反而会先从等候室到沙发，先得到理发。这就产生了饿死现象。  解决的方案是加上一个最久等候顾客id的标识current\_id，然后根据不同情况更新这个标识。        实验结果在上一个部分中已经列出，与预测的结果一致，复现了设计目标。  这次实验也是使用ipc库在linux系统下做进程互斥，和实验四有点类似。有了实验四的基础，这个实验做起来就没这么难了。通过进一步研究和实践操作系统中关于并发进程同步与互斥操作的一些经典问题的解法，加深了对于非对称性互斥问题有关概念的理解。  实验中我创建了4个并发的进程，三个为生产者，一个为消费者。进程互斥控制的功能采用c语言的ipc库编程实现。 | | | |