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*2* | | | |
| 实验题目：进程同步实验 | | | |
| 实验学时：2 | | 实验日期：2023.05.12 | |
| 实验目的：  加深对并发协作进程同步与互斥概念的理解，观察和体验并发进程同步与互斥 操作的效果，分析与研究经典进程同步与互斥问题的实际解决方案。了解 Linux 系统 中 IPC 进程同步工具的用法，练习并发协作进程的同步与互斥操作的编程与调试技 术。 | | | |
| 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：  独立实验要求一个系统中有三个抽烟者进程，每个抽烟者不断地卷烟并抽 烟。 抽烟者卷起并抽掉一颗烟需要有三种材料：烟草、纸和胶水。一个抽烟者有烟草，一 个有纸，另一个有胶水。系统中还有两个供应者进程，它们无限地供应所有三种材料， 但每次仅轮流提供三种材料中的两种。得到缺失的两种材料的抽烟者在卷起并抽掉一 颗烟后会发信号通知供应者，让它继续提供另外的两种材料。这一过程重复进行。  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。  随后建立生产者与消费者，因为两个供应者都可以提供三种材料，因此这两个供应者算同一种，只需要一个.c文件就可以实现了。另外的三个消费者进程，需求均不相同（需要吸烟的材料不一样），因此需要三种不同的实现，则建立三个.c文件。  代码目录结构如下。    首先需要实现生产者功能。声明变量all\_int，用来表示生产者同步信号；声明变量produce\_int，用来表示生产者互斥信号。        声明ab/ac/bc\_int，用来表示消费者同步信号。这里我将烟草、纸和胶水符号化为a\b\c三个符号（注意这个符号和后续代码中的字符’A’,’B’,’C’没有任何关系!），而ab\_int代表第一个消费者拥有胶水A，则它需要烟草和纸（a,b），其他的几个变量同理。    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。  随后，要让生产者不断的生产材料，每次生产两种。这一功能我通过一个轮次标识符来实现。声明定义标识符d，用来表示这一轮应该生产哪些材料，d=0生产烟草和纸，d=1生产胶水和纸，d=2生产烟草和胶水。    如果缓存区没有满，则down(all\_int)不会阻塞生产者，若此时生产区中没有生产者，则down(produce\_int)也不会阻塞它，生产者进入缓存区按照轮次生产材料。    随后生产者根据轮次唤醒对应的消费者，消费者进入消费状态。  第二步我们要实现消费者功能。创建共享内存、信号量的代码与生产者部分的代码大同小异，不再赘述。  在具体消费功能的实现上，也比较容易。只需要在每次生产者生产完毕后消费即可。    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。    首先启动第一个消费者Consumer\_A,由于现在还没有生产者，因此它肯定是被阻塞的。    同样启动Consumer\_B/C,也都是阻塞的。      然后启动第一个生产者，可以看到生产者按轮次生产，消费者也按轮次拿到材料进行消费。    随后启动第二个生产者进程，两个生产者进程互斥生产，消费者进程按两个生产者进程的up(xx\_int)消费。    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。执行三个生产者进程和两个消费者进程。两个生产者进程互斥生产，消费者进程按两个生产者进程的up(xx\_int)消费。  实验中我创建了五个并发的进程，两个为生产者，三个为消费者。进程同步控制的功能采用c语言的ipc库编程实现。 | | | |
| 结论分析与体会：  实验中我遇到一个编译错误，即gmake后产生了报错:”multiple definition’，后来发现是我的gcc版本较高，在ipc.h中声明的全局变量和具体实现代码中的变量产生了重复定义。通过删除全局变量改为局部变量完成debug。  实验结果在上一个部分中已经列出，与预测的结果完全一致，完全复现了设计目标。  这次实验相比与前三次更难，不仅是代码实现上，更在同步机制的理解上。通过分析与研究经典进程同步与互斥问题的实际解决方案。了解了Linux 系统 中 IPC 进程同步工具的用法。  实验中我创建了五个并发的进程，两个为生产者，三个为消费者。进程同步控制的功能采用c语言的ipc库编程实现。 | | | |