**电子科技大学信息与软件工程学院**

## 实验指导书

**（实验）课程名称：操作系统基础**

**电子科技大学教务处制表**

## 实验一、信号量经典问题的实现

### 实验学时：4

### 实验内容和目的：

**实验目的**：本实验分为两部分，首先实现哲学家就餐问题，要求不能出现死锁。通过本实验熟悉Linux系统的基本环境，了解Linux下进程和线程的实现。然后实现生产者/消费者问题，通过本实验掌握进程间的同步和互斥机制的使用。熟悉基于某操作系统进程和线程的编程。

1. **掌握进程、线程的概念，熟悉相关的控制原语。**
2. **掌握进程、线程间的同步原理和方法。**
3. **掌握进程、线程间的互斥原理和方法。**
4. **掌握使用信号量原语解决进程、线程间互斥和同步方法。**

**实验内容：**

该实验分为两部分内容:

第一步：实现哲学家就餐问题

1）熟悉Ubuntu系统环境和命令；

2）熟悉Ubuntu系统下的多线程编程；

2）在Ubuntu系统下编程实现哲学家就餐问题。

实现教材2.5.2节中所描述的哲学家就餐问题。要求显示出每个哲学家的工作状态，如吃饭，思考。连续运行30次以上都未出现死锁现象。

第二步：实现生产者/消费者问题

1）有一群生产者进程在生产产品，并将这些产品提供给消费者进程去消费。为使生产者进程与消费者进程能并发执行，在两者之间设置了一个具有n个缓冲区的缓冲池：生产者进程从文件中读取一个数据，并将它存放到一个缓冲区中； 消费者进程从一个缓冲区中取走数据，并输出此数据。生产者和消费者之间必须保持同步原则：不允许消费者进程到一个空缓冲区去取产品；也不允许生产者进程向一个已装满产品且尚未被取走的缓冲区中投放产品。

2）、创建3进程（或者线程）作为生产者，4个进程（或者线程）作为消费者。创建一个文件作为数据源，文件中事先写入一些内容作为数据。

3）、生产者和消费者进程（或者线程）都具有相同的优先级。

### 实验原理：

由Dijkstra提出并解决的哲学家进餐问题(The Dinning Philosophers Problem)是典型的同步问题。该问题是描述有五个哲学家共用一张圆桌，分别坐在周围的五张椅子上，在圆桌上有五个碗和五只筷子，他们的生活方式是交替地进行思考和进餐。平时，一个哲学家进行思考，饥饿时便试图取用其左右最靠近他的筷子，只有在他拿到两只筷子时才能进餐。进餐完毕，放下筷子继续思考。

第二个问题考虑n个缓冲区的缓冲池作为一个共享资源，当生产者进程从数据源—文件中读取数据后将会申请一个缓冲区，并将此数据放入缓冲区中。消费者从一个缓冲区中取走数据，并将其中的内容打印输出。当生产者进程正在访问缓冲区时，消费者进程不能同时访问缓冲区，因此缓冲区是个互斥资源。

### 实验器材（设备、元器件）

1. 学生每人一台PC，安装Windows8操作系统。
2. 个人PC安装VMware虚拟机和Ubuntu系统。

### 实验步骤：

（一） 熟悉Ubuntu系统下的多线程编程。

1. 使用“Ctrl+Alt+T”打开终端；

2. 使用gedit或vim命令打开文本编辑器进行编码： “gedit 文件名.c”

3．编译程序：

“ gcc 文件名.c -o 可执行程序名 ”

（如果只输入 gcc 文件名.c，默认可执行程序名为a.out）

使用线程库时，gcc编译需要添加-lpthread

4. 执行程序：./可执行程序名

（二）信号量同步问题实现

步骤1：分配具有n个缓冲区的缓冲池，作为共享资源。

步骤2：定义两个资源型信号量empty 和full，empty信号量表示当前空的缓冲区数量，full表示当前满的缓冲区数量。

步骤3：定义互斥信号量mutex，当某个进程访问缓冲区之前先获取此信号量，在对缓冲区的操作完成后再释放此互斥信号量。以此实现多个进程对共享资源的互斥访问。

步骤4：创建3进程（或者线程）作为生产者，4个进程（或者线程）作为消费者。创建一个文件作为数据源，文件中事先写入一些内容作为内容。

步骤5 ：编写代码实现生产者进程的工作内容，即从文件中读取数据，然后申请一个empty信号量，和互斥信号量，然后进入临界区操作将读取的数据放入此缓冲区中。并释放empty信号量和互斥信号量。

步骤6：编写代码实现消费者者进程的工作内容，即先申请一个full信号量，和互斥信号量，然后进入临界区操作从缓冲区中读取数据并打印输出。

### 实验报告

1. 说明实验过程。
2. 进行结果分析。

## 实验二、利用管道实现两个进程的通信

### 实验学时：4

### 实验目的

1. **熟悉Linux下的应用程序开发**
2. **熟悉Linux的进程控制原语的使用**
3. **掌握Linux操作系统的进程间通信机制管道的使用。**
4. **掌握Linux操作系统中父进程与子进程的同步。**

### 实验内容

在Linux系统中使用系统调用fork()创建两个子进程，使用系统调用pipe()建立一个管道，两个子进程分别向管道各写一句话：

Child process 1 is sending a message!

Child process 2 is sending a message!

而父进程则从管道中读出来自于两个子进程的信息，显示在屏幕上。然后分别结束两个子进程的运行。

要求：

1.父进程先接收子进程P1发来的消息，然后再接收子进程P2发来的消息。

2.在Linux平台下实现。

### 实验原理

首先创建两个子进程，注意Linux下使用fork()函数创建进程的方法。父进程和两个子进程间需要同步，使用waitpid（）函数实现父进程等待子进程运行完毕后从管道中读取数据并打印，只有子进程将数据写入管道后，父进程才能够执行打开管道操作。

由于fork函数让子进程完整地拷贝了父进程的整个地址空间，所以子进程都有管道的读端和写端。所以在相关进程中最好关掉不用的那一端。根据要求，“父进程先接收子进程P1发来的消息，然后再接收子进程P2发来的消息。”存在两个同步问题，两个子进程和父进程之间（先子写后父读）同步、子进程1和子进程2之间（先1写，再2写）

### 实验器材（设备、元器件）

1. 学生每人一台PC，安装Windows8操作系统。
2. 个人PC安装VMware虚拟机和Ubuntu系统。

### 实验步骤

步骤1：创建一个管道。

步骤2：创建子进程1，向管道中写入“Child process 1 is sending a message!”，并做好跟父进程的同步执行。

步骤3：创建子进程2，向管道中写入“Child process 2 is sending a message!”，并做好跟父进程的同步执行。

步骤4：父进程从管道中读取数据，并打印输出。

### 实验报告

1. 说明实验过程。
2. 进行结果分析。