南京信息工程大学 操作系统实验报告

实验项目 生产者-消费者问题的实现

|  |  |  |  |
| --- | --- | --- | --- |
| 专业 计算机科学与技术 | 年级 22 | 学号 202283290159 | 姓名 张瑞晨 |

1. 主要实验步骤

|  |
| --- |
| 实验原理  生产者-消费者问题是经典的线程同步问题，主要描述生产者线程将数据放入缓冲区，而消费者线程从缓冲区取数据的过程。为了确保线程安全，必须保证：  当缓冲区已满时，生产者必须等待。  当缓冲区为空时，消费者必须等待。  生产者和消费者线程通过缓冲区进行通信。  1. 创建 Buffer 类  该类用于存储共享的产品，并包含 setProduct 和 getProduct 方法来控制生产者和消费者的访问。  package producer\_consumer;  public class Buffer {  private int product = 0;  private boolean enable = true;  public synchronized void setProduct(int value) {  while (!enable) {  try {  wait();  } catch (InterruptedException e) { }  }  product = value;  enable = false;  notify();  }  public synchronized int getProduct() {  while (enable) {  try {  wait();  } catch (InterruptedException e) { }  }  enable = true;  notify();  return product;  }  }  2. 创建 Producer 类  该类继承自 Thread 类，表示生产者线程。  package producer\_consumer;  public class Producer extends Thread {  private Buffer product;  public Producer(Buffer product) {  this.product = product;  }  public void run() {  for (int count = 1; count <= 10; count++) {  try {  Thread.sleep(2000);  } catch (InterruptedException e) { }  product.setProduct(count);  System.out.println("生产: " + count);  }  }  }  3. 创建 Consumer 类  该类继承自 Thread 类，表示消费者线程。  package producer\_consumer;  public class Consumer extends Thread {  private Buffer product;  public Consumer(Buffer product) {  this.product = product;  }  public void run() {  int value;  do {  try {  Thread.sleep(2000);  } catch (InterruptedException e) { }  value = product.getProduct();  System.out.println("消费: " + value);  } while (value != 10);  }  }  4. 创建主类 ProducerConsumerTest  该类包含主方法，用于启动生产者和消费者线程。 |

1. 实验结果

|  |
| --- |
| 实验结果  执行程序后，生产者线程每隔 2 秒生产一个产品，消费者线程每隔 2 秒消费一个产品，直到生产者生产出第 10 个产品。  输出示例  生产: 1  消费: 1  生产: 2  消费: 2  生产: 3  消费: 3  生产: 4  消费: 4  生产: 5  消费: 5  生产: 6  消费: 6  生产: 7  消费: 7  生产: 8  消费: 8  生产: 9  消费: 9  生产: 10  消费: 10 |

1. 实验遇到问题及解决

|  |
| --- |
| 问题：消费者线程在缓冲区为空时会一直等待，导致程序卡死。  解决方法： 使用 wait() 和 notify() 方法来实现线程之间的通信。确保当缓冲区为空时，消费者线程会进入等待状态；当缓冲区有产品时，消费者线程被唤醒。  问题：线程中断导致程序异常退出。  解决方法： 在捕获 InterruptedException 异常时，使用适当的处理逻辑，避免程序异常退出。 |

1. 总结

|  |
| --- |
| 通过本次实验，掌握了多线程同步的基本原理和实现方法。在生产者-消费者问题中，合理使用 synchronized、wait() 和 notify() 可以有效地解决线程之间的通信和同步问题。同时，深入理解了 Java 中多线程编程的核心思想及其应用场景。 |