**《嵌入式系统设计技术》实验四报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **实验题目** | **嵌入式操作系统时钟中断功能仿真** | | |
| **小组成员** | **李万达，章洮与，何一鸣** | | |
| **实验时间** | **2024/3/31** | **实验地点** | **教研室** |
| **实验成绩** |  | **实验性质** | **R验证性 □设计性 □综合性** |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | |
| **一、实验目的**  1. 现代计算机系统的硬件部分都设有中断机构，中断机构能发现中断事件，且当发现中断事件后迫使正在处理器上执行的进程暂时停止执行，而让操作系统的中断处理程序占有处理器去处理出现的中断事件；  2. 本实验模拟“时钟中断事件”的处理，对其它中断事件的模拟处理；  3. 通过本实验掌握嵌入式实时操作系统中中断的使用情况。  **二、实验背景**  在操作系统中，时钟中断是一种重要的机制，用于实现时间片轮转调度、计时器功能以及其他与时间相关的操作。时钟中断通过周期性触发中断来通知操作系统和应用程序时间的流逝，使得操作系统能够及时地进行调度和处理。本实验的模拟程序用于模拟硬件产生时钟中断，另一部分模拟操作系统的时钟中断处理程序。 | | | |
| **三、实验项目内容**  **实验名称：**嵌入式操作系统时钟中断功能仿真  **实验设备：**一台笔记本  **主要功能：**  **1.实验设计**  在本实验中，设计了三个任务Task1、Task2、Task3，创建了一个信号量InterruptSem。  **2.系统运行流程**   * 系统初始化，创建并启动任务Task1、Task2、Task3。Task1的运行时间为2 s，Task2的运行时间为5 s，Task3的运行时间为3 s； * 在主任务中创建一个信号量InterruptSem（初值为1）； * 创建时钟中断仿真器，设置中断时间为2 s，中断时会占用信号量，中断结束会释放信号量，允许任务执行; * 等待中断信号，然后执行三个任务，并执行5次； * 完成后停止仿真器，等待线程结束；   **仿真方式：**使用Python编程语言进行编程仿真  **提交内容：①** 实验报告文档（本文档）**②**实验源码**（****②作为附件上传）**  **四、操作指南**  在本实验中，将通过Python编程来模拟一个简单的系统，其中包含三个任务和一个时钟中断仿真器，使用信号量来协调任务的执行和中断的处理。  1. 实验准备   * 确保Python环境已安装。 * 确保了解基本的Python编程和多线程编程。 * 理解时钟中断的概念以及信号量在线程同步中的作用。   2. 编码实现   * 创建任务：定义三个任务，每个任务在被调用时执行一段时间。使用threading模块中的Thread来创建任务线程。      * 定义时钟中断仿真器：创建一个时钟中断仿真器类，模拟每2秒触发一次中断。使用threading模块中的Semaphore来表示中断信号量。      * 主程序流程：在主程序中初始化任务和时钟中断仿真器，并启动它们。     3. 测试和验证   * 运行程序，并观察输出。检查时钟中断是否每2秒触发一次，并且在中断处理期间，任务的执行是否被暂停。 * 确认三个任务是否按照预定的时间间隔运行，并在中断期间正确地等待信号量。 * 通过上述步骤，使用Python的`threading`模块来模拟了一个简单的多任务操作系统，其中包含时钟中断的处理和任务的同步执行。这个模拟有助于理解操作系统中的任务调度和中断处理机制。   **五、实验记录**  （详细描述本实验的主要步骤、关键代码、实验图片等）  **主要步骤：**   * 定义任务函数： 定义了三个任务函数 task1()、task2() 和 task3()，分别模拟了三个不同的任务，每个任务在循环中打印一条正在运行的消息，并执行一定时间。 * 定义定时器中断模拟线程类： 定义了一个 TimerInterpret 类，继承自 threading.Thread 类，用于模拟定时器中断。在 run() 方法中，该线程会循环执行，每次循环休眠 2 秒，然后打印 "Timer triggered" 消息，并且获取一个信号量（Semaphore），模拟定时器中断的发生。在获取信号量后，打印 "processing" 消息，然后释放信号量。 * 创建并启动线程： 创建了 TimerInterpret 的实例 Timer1，并启动该线程。同时创建了三个线程对象 t1、t2 和 t3，分别对应三个任务函数，然后启动这三个线程。然后等待线程执行完成，调用 join() 方法，等待所有线程执行完成。   **实验图片：**      **六、****实验调试过程**  （详细描述本实验中所遇问题与解决方案等）  **线程同步问题：** 多个线程并发执行时，可能会出现资源竞争和数据不一致的情况。  **解决方案：**使用信号量确保线程同步，在 TimerInterpret 类的 run() 方法中，使用 self.semaphore.acquire() 和 self.semaphore.release() 方法，确保定时器线程和其他任务线程之间的同步和互斥。  **定时器中断与任务执行的同步：** 如何确保定时器中断发生时，其他任务能够及时被阻塞，而不是在中断处理完成之前继续执行。  **解决方案：**合理设计任务执行逻辑，在 TimerInterpret 类的 run() 方法中，定时器线程在每次中断发生时获取信号量，其他任务线程在需要阻塞时尝试获取信号量。这样可以确保定时器中断发生时，其他任务能够及时被阻塞，直到定时器中断处理完成。  **七、实验效果展示**  （详细描述本实验中所遇问题与解决方案等）     * "Task1 is running"、"Task2 is running" 和 "Task3 is running" 是三个任务的执行过程，在循环中打印的任务执行信息。 * "Timer triggered" 是定时器中断触发的信息，表示模拟的定时器中断发生了。 * "processing" 是定时器中断处理过程中打印的信息，表示定时器中断处理正在进行中。   在任务开始时，任务1、任务2和任务3按顺序启动并开始执行，因此会看到它们的输出信息 "Task1 is running"、"Task2 is running" 和 "Task3 is running"。随后，定时器中断被触发，输出了 "Timer triggered"。紧接着，定时器中断处理过程开始执行，输出了 "processing"。定时器中断处理完成后，之前未完成的任务会继续执行，当定时器再次触发时，整个过程会重复。  **八、实验总结**  本实验旨在探讨如何使用信号量（Semaphore）实现定时器中断与任务的同步，以模拟实时系统中任务调度和同步的情况。在实验中，我们设计了多个任务，并模拟了定时器中断的发生，以及定时器中断处理对任务的影响。  本实验深入探讨了多线程编程中的任务调度和同步问题，在模拟实时系统中的任务调度和定时器中断时，使用了信号量等同步机制，有效地实现了任务之间的同步和互斥。同时，实验也展示了Python中多线程编程的基本方法和技巧，以及如何使用线程同步机制来解决多线程并发执行时可能遇到的问题。通过本实验的实践，加深了对多线程编程的理解和掌握，为进一步研究和应用多线程编程打下了基础。  **九、小组分工**  （列出小组组长、成员名字，并详细描述每位所承担工作）  小组组长：章洮与  成员：章洮与，何一鸣，李万达  分工：本次实验为小组所有成员共同探讨所完成，三人工作量基本一致。 | | | |
|  | | | |