|  |  |  |  |
| --- | --- | --- | --- |
| 姓名：刘赟 | 专业：物联网工程 | 班级：20181191班 | 学号：2018119117 |
| 科目：嵌入式应用系统开发 | | 实验日期：2020/7/6 | |
| 实验题目：互斥锁 | | | |
| 【实验目的】   1. 在多任务环境下，往往存在多个任务竞争同一共享资源的应用场景，互斥锁可被用于对共享资源的保护从而实现独占式访问。通过互斥锁，用于实现对共享资源的独占式处理。 | | | |
| 【实验内容】  1.实验设备   硬件：PC机一台   小熊派开发板一套   软件：XP/win7、8、8.1、10 系统，IoT Studio，git 客户端 [sourcetree](https://www.sourcetreeapp.com/)，课程使用 sourcetree 和 [github](http://www.github.com/) 作为项目管理和协作系统。  2.实验内容和要求  本实验中将创建两个任务，一个低优先级任务task1，一个高优先级任务task2，两个任务之间依次对共享资源上锁、操作、解锁，在串口终端中观察两个任务的运行情况。 | | | |
| 【实验结果】   1. 代码实现   osal\_mutex\_demo.c  /\* 使用osal接口需要包含该头文件 \*/  #include <osal.h>  /\* 任务优先级宏定义（shell任务的优先级为10） \*/  #define USER\_TASK1\_PRI  12  //低优先级  #define USER\_TASK2\_PRI  11  //高优先级  /\* 共享资源 \*/  uint32\_t public\_value = 0;  /\* 互斥锁索引ID \*/  osal\_mutex\_t public\_value\_mutex;  /\* 任务task1入口函数 \*/  static int user\_task1\_entry()  {      while(1)      {          /\* 尝试获取互斥锁 \*/          if(true == osal\_mutex\_lock(public\_value\_mutex))          {              /\* 获取到互斥锁，对共享资源进行操作 \*/              printf("\r\ntask1: lock a mutex.\r\n");              public\_value += 10;              printf("task1: public\_value = %ld.\r\n", public\_value);              /\* 对共享资源操作完毕，释放互斥锁 \*/              printf("task1: unlock a mutex.\r\n\r\n");              osal\_mutex\_unlock(public\_value\_mutex);              /\* 满足条件则结束任务 \*/              if(public\_value > 100)                  break;          }      }      /\* while(1)会执行结束，所以需要返回值 \*/      return 0;  }  /\* 任务task2入口函数 \*/  static int user\_task2\_entry()  {      while (1)      {          /\* 尝试获取互斥锁 \*/         if(true == osal\_mutex\_lock(public\_value\_mutex))          {              /\* 获取到互斥锁，对共享资源进行操作 \*/              printf("\r\ntask2: lock a mutex.\r\n");              public\_value += 5;              printf("task2: public\_value = %ld.\r\n", public\_value);              /\* 对共享资源操作完毕，释放互斥锁 \*/              printf("task2: unlock a mutex.\r\n\r\n");              osal\_mutex\_unlock(public\_value\_mutex);              /\* 满足条件则结束任务 \*/              if(public\_value > 90)                  break;              /\* 优先级较高，需要挂起一下，让task1获取到互斥锁，否则task2再次上锁，形成死锁 \*/              osal\_task\_sleep(10);          }      }      /\* while(1)会执行结束，所以需要返回值 \*/      return 0;  }  /\* 标准demo启动函数，函数名不要修改，否则会影响下一步实验 \*/  int standard\_app\_demo\_main()  {      /\* 创建互斥锁public\_value\_mutex \*/      osal\_mutex\_create(&public\_value\_mutex);      /\* 创建任务task1 \*/      osal\_task\_create("user\_task1",user\_task1\_entry,NULL,0x400,NULL,USER\_TASK1\_PRI);      /\* 创建任务task2 \*/      osal\_task\_create("user\_task2",user\_task2\_entry,NULL,0x400,NULL,USER\_TASK2\_PRI);      return 0;  }  将编写的osal\_mutex\_demo.c文件添加到makefile中，直接修改Demos文件夹下的user\_demo.mk配置文件  #example for osal\_mutex\_demo  ifeq ($(CONFIG\_USER\_DEMO), "osal\_mutex\_demo")      user\_demo\_src  = ${wildcard $(TOP\_DIR)/targets/STM32L431\_BearPi/Demos/osal\_kernel\_demo/osal\_mutex\_demo.c}  endif  4.实验现象  a9f8d7bdfb649618958f23940219780  **串口现象：**  **linkmain:V1.2.1 AT 13:52:23 ON July 6 2021**  **WELCOME TO IOT\_LINK SHELL**  **LiteOS:/>**  **task2: lock a mutex.**  **task2: public\_value = 5.**  **task2: unlock a mutex.**  **task1: lock a mutex.**  **task1: public\_value = 15.**  **task1: unlock a mutex.**  **task1: lock a mutex.**  **task1: public\_value = 25.**  **task1: unlock a mutex.** | | | |
| **【**教师评语和成绩**】**  **成绩：** **指导教师：** **日期：** | | | |