实验四：多线程程序设计实验

一、实验目的

1.学习mbed操作系统多线程程序的设计方法；

2.掌握多线程同步、多线程间通讯技术；

3.掌握调试程序的方法

二、实验仪器与设备

Mbed 开发板、连接线、计算机。

三、预备知识

对于mbed操作系统而言，它需要完成进程（线程）管理、存储管理、设备管理、作业管 理、安全管理等功能，但受到嵌入式系统资源及用途的闲置，嵌入式实时操作系统的主要功 能就是线程管理，具体包括：

- 线程控制及调度:线程是具有一定独立功能的程序关于某个数据集合上的一次运行 活动,是系统进行资源分配的单位。线程控制的主要任务就是创建线程、撤销线程 和控制线程运行时候的各种状态转换,而线程调度的任务是从线程的就绪队列中按 照一定的算法挑选出一个线程，把处理器资源分配给它，并准备好特定的执行环境 让它执行起来。
- 线程同步：多个线程的执行是并发的，它们以异步的方式运行，它们的执行进度也 是不可预知的;为了使多个线程可以有条不紊地运行,操作系统要提供线程同步机 制，以协调线程的执行。一般有两种协调方式：互斥和同步。互斥指多个线程对临 界资源访问时采用互斥的形式;同步则是在相互协作共同完成任务的线程之间，用 同步机制协调它们之间的执行顺序。
- 线程间通信:线程间通信主要发生在相互协作的线程之间，由操作系统提供的线程间通信机制是它们之间相互交换数据和消息的手段。
  - 线程调度：一个线程可能是以下状态中的一种:
  - 运行态:线程正在运行当中,一个时刻只能有一个线程处于运行态;
  - 就绪态:线程已经获得必要的资源,一旦正在运行的线程终止或这进入等待状态,队 列中具有最高优先级的线程将变成运行态;
  - 等待态:线程正在等待相关的事件发生;
  - 停止态:此时线程并没有创建， 也没有消耗任何系统资源。

（1） 线程控制

线程控制的主要目的是允许用户建立多个线程并发执行，从而简化各类应用的开发，mbed-rtos 提供了 Thread 对象来完成线程控制，它提供的主要方法有：

类名	方法	用途
Thread	Thread(void (*task)(void const *argument), void *argument=NULL, osPriority priority=osPriorityNormal, uint32_t stack_size=DEFAULT_STACK_SIZE,	构造函数，输出参数分别是任务函数指针，函数参数指针，线程优先级， 堆栈大小，堆栈指针，默认为普通优 先级，2K 字节内存堆栈，使用内部分 配方式，如果传入堆栈指针，则使用 传入指针指向的内存
	osStatus terminate();	结束本线程
	osStatus set_priority(osPriority priority);	设置本线程优先级
	osPriority get_priority();	获取本线程优先级

	int32_t signal_set(int32_t signals);	设置本线程信号量
	State get_state();	静态函数，获取当前线程状态
	static osEvent signal_wait(int32_t signals, uint32_t millisec=osWaitForever);	静态函数，等待信号量
	static osStatus wait(uint32_t millisec);	静态函数，挂起当前线程 millisec 毫秒
	static osStatus yield();	静态函数，把运行权交给下一个线程
	static osThreadId gettid();	静态函数，获取当前线程 ID

（2） 线程同步

需要线程同步是因为在多线程并 行执行的情况下，有可能会造成多线程同时使用同一系统资源的情况从而造成冲突的发生， 影响程序执行的正确性，在操作系统中一般使用互斥锁与信号量来实现线程同步，在 mbed 中对应的类是 Mutex 和 Semaphore，它们提供的主要方法如下：

类名	方法	用途
Mutex	Mutex();	构造函数，定义一个互斥锁
	osStatus lock(uint32_t millisec=osWaitForever);	等待直到互斥锁有效
	bool trylock();	判断互斥锁是否有效，它会立刻返回
	osStatus unlock();	释放本线程原先锁定互斥锁
Semaphore	Semaphore(int32_t count);	构造函数，定义一个信号量，传入参数为允许的资源数
	int32_t wait(uint32_t millisec=osWaitForever);	等待直到资源有效，返回为可用的资源数，-1 表示错误
	osStatus release(void);	释放本线程占用的资源

（3）线程间通讯

mbed-rtos 提供了多种方式的线程间通讯，包括队列（Queue），邮件（Mail）两种，另外 还提供了一个辅助类内存池（MemoryPool，用于 Queue 中的数据存储），它们提供的主要方 法如下：

类名	方法	用途
Queue	template<typename T, uint32_t queue_sz> Queue()	构造函数,构造大小为 queue_sz,类型为 T 的队列
	osStatus put(T* data, uint32_t millisec=0)	把 T 类型的数据放入队列
	osEvent get(uint32_t millisec=osWaitForever)	获取队列信息，取出的数据包含在 osEvent 对象中
MemoryPool	template<typename T, uint32_t pool_sz> MemoryPool()	构造函数，构造大小为 pool_sz，类型为 T 的内存池
	T* alloc(void)	从内存池中分配存储 T 类型的内存块，返回的是内存地址，如果分配不成功则返回 NULL

	T* calloc(void)	从内存池中分配存储 T 类型的内存块 并把它填充为 0，返回的是内存地址，
	osStatus free(T *block)	把 block 指向的内存释放给内存池
Mail	template<typename T, uint32_t queue_sz> Mail()	构造函数，构造大小为 queue_sz，类型为 T 的邮件队列
	T* alloc(uint32_t millisec=0)	在 Mail 中分配存储 T 类型的内存块，返回的是内存地址，如果分配不成功则返回 NULL
	T* calloc(uint32_t millisec=0)	从系统中分配存储 T 类型的内存块并把它填充为 0，返回的是内存地址，如果分配不成功则返回 NULL
	osStatus put(T *mptr)	把 T 类型的数据放入邮件队列中
	osEvent get(uint32_t millisec=osWaitForever)	获取队列信息，取出的数据包含在 osEvent 对象中
	osStatus free(T *mptr)	把 mptr 指向的内存释放给系统

四、实验内容

（1）利用互斥锁实现线程同步

```
#include "rtos.h"
Serial pc(USBTX,USBRX);
Mutex stdio_mutex; // Semaphore
slots(1); uint8_t theadindex[3];
long count[3];
void printstr(void const *args)
{
  while (true) {
        stdio_mutex.lock();//slots.wait();
        pc.printf("Hello World,My Thread Name is
 %d,Thread ID is %d.\n",*(uint8_t *)args,Thread::gettid());
        count[*(uint8_t *)args-1]++;
        stdio_mutex.unlock();//slot.releas
        e(); Thread::yield();
  }
}
int main()
{
  theadindex[
  0]=1;
  theadindex[
  1]=2;
  theadindex[
  2]=3;
  Thread thread1(printstr,(void *)theadindex,osPriorityNormal);
  Thread thread2(printstr,(void
  *)(theadindex+1),osPriorityNormal); Thread
  thread3(printstr,(void *)(theadindex+2),osPriorityNormal); while
  (1){
        stdio_mutex.lock();//slots.wait();
        pc.printf("Thread1 count is %ld,Thread2 count is %ld,,Thread3 count
```

```
is %ld. \n",count[0],count[1],count[2]);
                    stdio_mutex.unlock();//slot.release();
                    Thread::wait(1000);
        }
```

（2）利用邮箱机制实现线程间通讯

```
#include "rtos.h"
typedef  struct  {  uint32_t
  length; char str[255];
} message_t;

Mail<message_t, 16> queue; Serial
pc(USBTX,USBRX);

void send_thread (void const *args) { int32_t i =-1;
  char serialdata[255];

  while (true) {
        serialdata[++i]=pc.getc();
        if (serialdata[i]=='\n' || i==255)
        {
              message_t *message = queue.alloc();
              message->length = i;
              memcpy(message->str,serialdata,i);
              queue.put(message);
                  i=-1;
        }
        Thread::yield();
  }
}
    int main (void) {
  Thread thread(send_thread);
  while (true) {
        osEvent evt = queue.get();
        if (evt.status == osEventMail) {
              message_t *message = (message_t*)evt.value.p; pc.printf("User
              input length is %d.\n",message->length); for (uint8_t
              i=0;i<message->length;i++)
                        pc.putc(message->str[i]);
              pc.printf("\n"); queue.free(message);
        }
        Thread::yield();
  }
}
```

五、实验要求

1、认真阅读预备知识和实验内容；
2、编写代码，编译、下载到 mbed 开发板，验证实验效果。
3、撰写实验报告。