(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101872317 B
(45)授权公告日 2012.12.26

(21)申请号 201010230519.0

(22)申请日 2010.07.16

(73)专利权人 山东中创软件工程股份有限公司
地址 250014 山东省济南市千佛山东路41-1号
专利权人 山东中创软件商用中间件股份有限公司

(72)发明人 王凯 刘江宁

(74)专利代理机构 北京集佳知识产权代理有限公司 11227
代理人 逯长明

(51)Int.Cl.
G06F 9/52(2006.01)
G06F 9/54(2006.01)

(56)对比文件
CN 101609417 A,2009.12.23, 全文.
CN 1744046 A,2006.03.08, 全文.
WO 2007051935 A1,2007.05.10, 全文.
曹庆年 张金森 孟开元.VxWorks多任务调度策略的研究.《中国科技信息》.2008,(第8期),全文.

审查员 钟阳雪

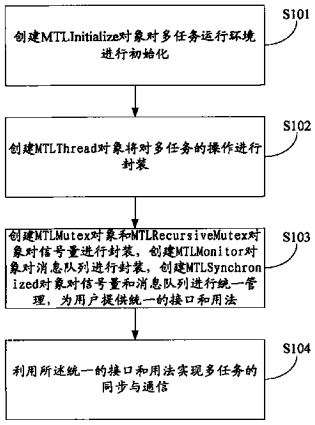
权利要求书 1 页 说明书 5 页 附图 1 页

(54)发明名称

VxWorks多任务同步与通信方法

(57)摘要

本发明公开了一种VxWorks多任务同步与通信方法,包括:创建MTLInitialize对象对多任务运行环境进行初始化;创建MTLThread对象将对多任务的操作进行封装;创建MTLMutex对象和MTLRecursiveMutex对象对信号量进行封装,创建MTLMonitor对象对消息队列进行封装,创建MTLSynchronized对象对信号量和消息队列进行统一管理,为用户提供统一的接口和用法;利用所述统一的接口和用法实现多任务的同步与通信。本发明利用C++类封装的方法对VxWorks的消息队列和信号量进行封装,轻松地实现多任务之间的同步和通信,大大降低了多任务协作的难度。



CN 101872317 B

1. 一种 VxWorks 多任务同步与通信方法，其特征在于，包括：

创建 MTLInitialize 对象对多任务运行环境进行初始化；

创建 MTLThread 对象将对多任务的操作进行封装，其中所述对多任务的操作包括：任务创建、任务运行、任务暂停和任务销毁；

创建 MTLMutex 对象和 MTLRecursiveMutex 对象对信号量进行封装，创建 MTLMonitor 对象对消息队列进行封装，创建 MTLSynchronized 对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法；

其中，所述创建 MTLMutex 对象对信号量进行封装包括：在 MTLMutex 的构造函数中创建信号量，在 MTLMutex 的析构函数中删除信号量，在 MTLMutex 的 lock 函数中获取信号量，在 MTLMutex 的 unlock 函数中释放信号量；

所述创建 MTLRecursiveMutex 对象对信号量进行封装包括：在 MTLRecursiveMutex 的构造函数中创建信号量，在 MTLRecursiveMutex 的析构函数中删除信号量，在 MTLRecursiveMutex 的 lock 函数中获取信号量，在 MTLRecursiveMutex 的 unlock 函数中释放信号量；

所述创建 MTLMonitor 对象对消息队列进行封装包括：在 MTLMonitor 的构造函数中创建消息队列，在 MTLMonitor 的析构函数中删除消息队列，在 MTLMonitor 的 wait 函数中从消息队列获取消息，在 MTLMonitor 的 notify 函数中向消息队列发送消息；

所述创建 MTLSynchronized 对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法包括：在 MTLSynchronized 的构造函数中获取信号量和 / 或从消息队列中获取消息，在 MTLSynchronized 的析构函数中释放信号量和 / 或向消息队列中发消息，在 MTLSynchronized 的成员函数中实现对信号量或消息队列的其它操作；

利用所述统一的接口和用法实现多任务的同步与通信，包括：通过调用 MTLMutex 和 MTLRecursiveMutex 类的 lock 和 unlock 函数进行多任务之间的同步；通过调用 MTLMonitor 类的 wait 和 notify 函数实现多任务之间的通信；通过定义 MTLSynchronized 类对象实现多任务的同步与通信；

其中，所述通过定义 MTLSynchronized 类对象实现多任务的同步与通信包括：

MTLSynchronized 的构造函数调用其成员变量 MTLMonitor 类对象的 wait 函数以及 MTLMutex 和 MTLRecursiveMutex 类对象的 lock 函数，获取信号量或从消息队列中获取消息；

MTLSynchronized 的析构函数中调用其成员变量 MTLMonitor 类对象的 notify 函数以及 MTLMutex 和 MTLRecursiveMutex 类对象的 unlock 函数，释放信号量和向消息队列发送消息；

MTLSynchronized 的成员函数实现对信号量和消息队列的其它操作。

VxWorks 多任务同步与通信方法

技术领域

[0001] 本发明涉及 VxWorks 操作系统技术领域，尤其涉及一种 VxWorks 多任务同步与通信方法。

背景技术

[0002] VxWorks 操作系统是一种嵌入式实时操作系统，允许多任务的同时运行。当多任务同时运行时，任务之间的同步与通信是非常必要的。

[0003] 共享数据是 VxWorks 操作系统实现多任务同步与通信的途径之一。采用共享数据时，通常采用 VxWorks 提供的信号量来保证共享数据的互斥访问，但当相互之间需要同步与通信的任务比较多时，共享数据互斥访问的算法会很复杂，加大了多任务协作的难度。

[0004] 此外，VxWorks 操作系统还提供了消息队列的方式来允许任务之间相互发送消息，任何任务都可以发送消息到消息队列和从消息队列接收消息，来实现相互间的同步与通信。但是，由于两个任务间全双工地通信一般需要两个消息队列，每个提供一个流通方向，因此当相互之间需要同步与通信的任务比较多时，消息队列的数量会变得很大，对消息队列的管理也会变得很复杂，加大了多任务协作的难度。

发明内容

[0005] 有鉴于此，本发明提供一种 VxWorks 多任务同步与通信方法，以解决现有技术中，当相互之间需要同步与通信的任务比较多时，由于共享数据互斥访问的算法很复杂或消息队列的数量很大，所造成的多任务协作难度大的问题，技术方案如下：

[0006] 一种 VxWorks 多任务同步与通信方法，包括：

[0007] 创建 MTLInitialize 对象对多任务运行环境进行初始化；

[0008] 创建 MTLThread 对象将对多任务的操作进行封装；

[0009] 创建 MTLMutex 对象和 MTLRecursiveMutex 对象对信号量进行封装，创建 MTLMonitor 对象对消息队列进行封装，创建 MTLSynchronized 对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法；

[0010] 利用所述统一的接口和用法实现多任务的同步与通信。

[0011] 优选的，上述方法中，所述对多任务的操作包括：

[0012] 任务创建、任务运行、任务暂停和任务销毁。

[0013] 优选的，上述方法中，所述创建 MTLMutex 对象对信号量进行封装包括：

[0014] 在 MTLMutex 的构造函数中创建信号量，在 MTLMutex 的析构函数中删除信号量，在 MTLMutex 的 lock 函数中获取信号量，在 MTLMutex 的 unlock 函数中释放信号量。

[0015] 优选的，上述方法中，所述创建 MTLRecursiveMutex 对象对信号量进行封装包括：

[0016] 在 MTLRecursiveMutex 的构造函数中创建信号量，在 MTLRecursiveMutex 的析构函数中删除信号量，在 MTLRecursiveMutex 的 lock 函数中获取信号量，在 MTLRecursiveMutex 的 unlock 函数中释放信号量。

[0017] 优选的，上述方法中，所述创建 MTLMonitor 对象对消息队列进行封装包括：

[0018] 在 MTLMonitor 的构造函数中创建消息队列，在 MTLMonitor 的析构函数中删除消息队列，在 MTLMonitor 的 wait 函数中从消息队列获取消息，在 MTLMonitor 的 notify 函数中向消息队列发送消息。

[0019] 优选的，上述方法中，所述创建 MTLSynchronized 对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法包括：

[0020] 在 MTLSynchronized 的构造函数中获取信号量和 / 或从消息队列中获取消息，在 MTLSynchronized 的析构函数中释放信号量和 / 或向消息队列中发消息，在 MTLSynchronized 的成员函数中实现对信号量或消息队列的其它操作。

[0021] 优选的，上述方法中，所述利用所述统一的接口和用法实现多任务的同步与通信包括：

[0022] 通过调用 MTLMutex 和 MTLRecursiveMutex 类的 lock 和 unlock 函数进行多任务之间的同步；

[0023] 通过调用 MTLMonitor 类的 wait 和 notify 函数实现多任务之间的通信；

[0024] 通过定义 MTLSynchronized 类对象实现多任务的同步与通信。

[0025] 优选的，上述方法中，所述通过定义 MTLSynchronized 类对象实现多任务的同步与通信包括：

[0026] MTLSynchronized 的构造函数调用其成员变量 MTLMonitor 类对象的 wait 函数以及 MTLMutex 和 MTLRecursiveMutex 类对象的 lock 函数，获取信号量或从消息队列中获取消息；

[0027] MTLSynchronized 的析构函数中调用其成员变量 MTLMonitor 类对象的 notify 函数以及 MTLMutex 和 MTLRecursiveMutex 类对象的 unlock 函数，释放信号量和向消息队列发送消息；

[0028] MTLSynchronized 的成员函数实现对信号量和消息队列的其它操作。

[0029] 通过以上技术方案可知，本发明通过将 VxWorks 的信号量和消息队列封装成多任务库，即创建 MTLMutex 对象和 MTLRecursiveMutex 对象对信号量进行封装，创建 MTLMonitor 对象对消息队列进行封装，创建 MTLSynchronized 对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法；从而使用户仅通过创建和使用 C++ 对象的方法即可对多任务进行同步和通信控制，大大降低了多任务协作的难度。

附图说明

[0030] 为了更清楚地说明本发明的技术方案，下面将对本发明描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

[0031] 图 1 为本发明实施例提供的 VxWorks 多任务同步与通信方法流程图。

具体实施方式

[0032] 本发明实施例针对现有技术存在的多任务协作的难度大的问题，提出了一种多任务同步与通信方法，该方法包括：创建 MTLInitialize 对象对多任务运行环境进

行初始化；创建 MTLThread 对象将对多任务的操作进行封装；创建 MTLMutex 对象和 MTLRecursiveMutex 对象对信号量进行封装，创建 MTLMonitor 对象对消息队列进行封装，创建 MTLSynchronized 对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法；利用所述统一的接口和用法实现多任务的同步与通信。

[0033] 本发明实施例提供的多任务同步与通信方法，利用 C++ 类封装的方法对 VxWorks 的消息队列和信号量进行封装，将消息队列和信号量与一定的数据结构相结合，使用户通过定义 C++ 类对象的方法来创建、使用和销毁消息队列与信号量，通过调用对象的方法来与其它任务通信，从而轻松地实现多任务之间的同步和通信，大大降低了多任务协作的难度。

[0034] 为了使本领域技术人员更好的理解和实施本发明，下面将结合说明书附图对本发明实施例的技术方案进行进一步详细阐述。

[0035] 参见图 1 所示，本发明实施例提供的 VxWorks 多任务同步与通信方法可以包括以下步骤：

[0036] S101，创建 MTLInitialize 对象对多任务运行环境进行初始化。

[0037] 使用多任务库之前，必须创建 MTLInitialize 对象，对 VxWorks 操作系统的多任务运行环境进行初始化。MTLInitialize 类的构造函数会进行任务私有变量的分配、创建任务管理类等初始化操作，通过创建 MTLInitialize 对象即可进行多任务运行环境的创建。

[0038] S102，创建 MTLThread 对象将对多任务的操作进行封装。

[0039] 通过创建 MTLThread 对象可以实现对多任务的操作进行封装，对多任务的操作包括任务创建、任务运行、任务暂停和任务销毁。其中，MTLThread 类的构造函数会进行 VxWorks 操作系统新任务的创建；通过从 MTLThread 类派生子类、再重载 MTLThread 类的 start 等方法即可运行和控制新任务。

[0040] S103，创建 MTLMutex 对象和 MTLRecursiveMutex 对象对信号量进行封装，创建 MTLMonitor 对象对消息队列进行封装，创建 MTLSynchronized 对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法。

[0041] MTLMutex 和 MTLRecursiveMutex 类对 VxWorks 的信号量进行了封装，MTLMonitor 类对 VxWorks 的消息队列进行了封装，这三个类都是 MTLSynchronized 类的成员变量，MTLSynchronized 类负责对信号量和消息队列进行统一管理，为用户提供统一的接口和用法。

[0042] 需要说明的是，所述创建 MTLMutex 对象对信号量进行封装包括：

[0043] 在 MTLMutex 的构造函数中创建信号量，在 MTLMutex 的析构函数中删除信号量，在 MTLMutex 的 lock 函数中获取信号量，在 MTLMutex 的 unlock 函数中释放信号量。

[0044] 所述创建 MTLRecursiveMutex 对象对信号量进行封装包括：

[0045] 在 MTLRecursiveMutex 的构造函数中创建信号量，在 MTLRecursiveMutex 的析构函数中删除信号量，在 MTLRecursiveMutex 的 lock 函数中获取信号量，在 MTLRecursiveMutex 的 unlock 函数中释放信号量。

[0046] 所述创建 MTLMonitor 对象对消息队列进行封装包括：

[0047] 在 MTLMonitor 的构造函数中创建消息队列，在 MTLMonitor 的析构函数中删除消息队列，在 MTLMonitor 的 wait 函数中从消息队列获取消息，在 MTLMonitor 的 notify 函数

中向消息队列发送消息。

[0048] 所述创建 MTLSynchronized 对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法包括：

[0049] 在 MTLSynchronized 的构造函数中获取信号量和 / 或从消息队列中获取消息，在 MTLSynchronized 的析构函数中释放信号量和 / 或向消息队列中发消息，

[0050] 在 MTLSynchronized 的成员函数中实现对信号量或消息队列的其它操作。

[0051] 由于信号量可实现多任务之间的同步和对资源的互斥访问，消息队列可提供多任务之间更高级别的通信与数据传送，因此这种统一封装的方式可实现绝大多数情况下的多任务之间的同步与通信处理，且符合面向对象的封装思想。

[0052] S104，利用所述统一的接口和用法实现多任务的同步与通信。

[0053] 在多任务库构造完成之后，用户就可以利用统一的接口和用法来实现多任务的同步与通信。

[0054] 由于 MTLMutex 和 MTLRecursiveMutex 类对信号量进行了封装，用户通过调用其 lock 和 unlock 等成员函数即可在多任务之间进行同步；MTLMonitor 类对消息队列进行了封装，用户通过调用其 wait 和 notify 等成员函数即可在多任务之间进行通信；此外，还通过定义 MTLSynchronized 类对象实现多任务的同步与通信。

[0055] 其中，通过定义 MTLSynchronized 类对象实现多任务的同步与通信可以包括：

[0056] MTLSynchronized 类的构造函数调用其成员变量 MTLMonitor 类对象的 wait 函数以及 MTLMutex 和 MTLRecursiveMutex 类对象的 lock 函数，获取信号量或从消息队列中获取消息；MTLSynchronized 类的析构函数中调用其成员变量 MTLMonitor 类对象的 notify 函数以及 MTLMutex 和 MTLRecursiveMutex 类对象的 unlock 函数，释放信号量和向消息队列发送消息；MTLSynchronized 的成员函数实现对信号量和消息队列的其它操作。

[0057] 从以上实施例可以看出，本发明通过将 VxWorks 的信号量和消息队列封装成多任务库，即创建 MTLMutex 对象和 MTLRecursiveMutex 对象对信号量进行封装，创建 MTLMonitor 对象对消息队列进行封装，创建 MTLSynchronized 对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法；从而使用户仅通过创建和使用 C++ 对象的方法即可对多任务进行同步和通信控制，大大降低了多任务协作的难度。

[0058] 此外，本发明通过采用 VxWorks 多任务库，在保证功能实现的前提下，使得多任务的同步和通信控制变得非常简单，大大提高了 VxWorks 程序的健壮性，有利于缩短项目周期，提高软件质量。

[0059] 通过以上的方法实施例的描述，所属领域的技术人员可以清楚地了解到本发明可借助软件加必需的通用硬件平台的方式来实现，当然也可以通过硬件，但很多情况下前者是更佳的实施方式。基于这样的理解，本发明的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得一台计算机设备（可以是个人计算机，服务器，或者网络设备等）执行本发明各个实施例所述方法的全部或部分步骤。而前述的存储介质包括：只读存储器 (ROM)、随机存取存储器 (RAM)、磁碟或者光盘等各种可以存储程序代码的介质。

[0060] 对所公开的实施例的上述说明，使本领域专业技术人员能够实现或使用本发明。对这些实施例的多种修改对本领域的专业技术人员来说将是显而易见的，本文中所定义的

一般原理可以在不脱离本发明的精神或范围的情况下，在其它实施例中实现。因此，本发明将不会被限制于本文所示的这些实施例，而是要符合与本文所公开的原理和新颖特点相一致的最宽的范围。

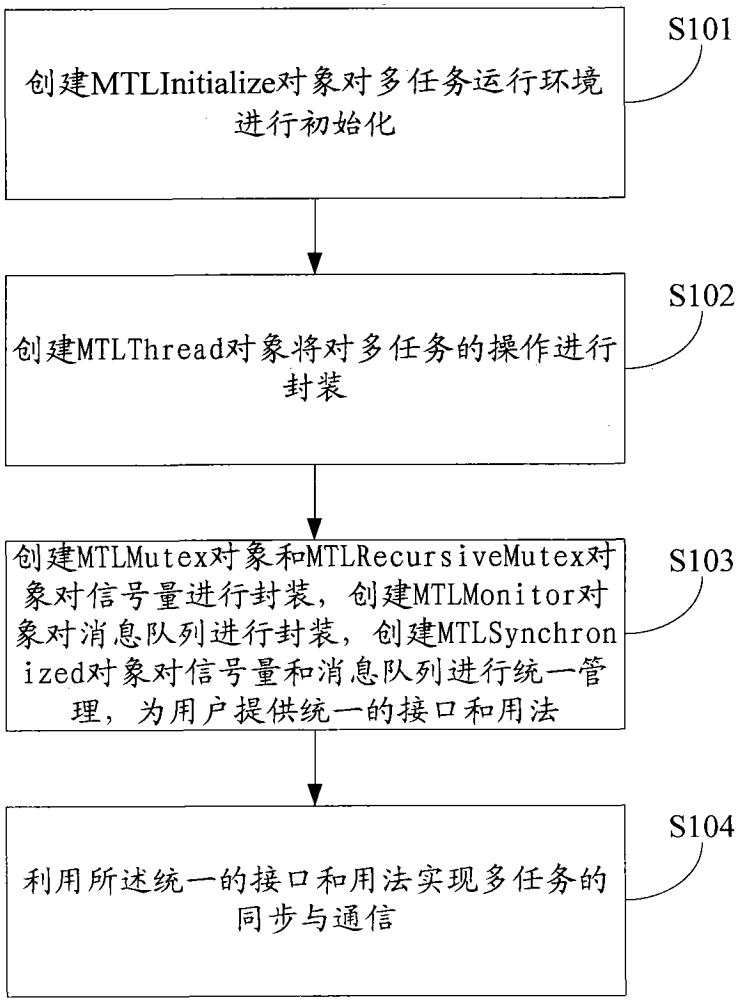

S101
创建MTLInitialize对象对多任务运行环境进行初始化
S102
创建MTLThread对象将对多任务的操作进行封装
S103
创建MTLMutex对象和MTLRecursiveMutex对象对信号量进行封装，创建MTLMonitor对象对消息队列进行封装，创建MTLSynchronized对象对信号量和消息队列进行统一管理，为用户提供统一的接口和用法
S104
利用所述统一的接口和用法实现多任务的同步与通信

图 1