嵌入式操作系统对多处理器的支持

鲍　峥　李红兵　金惠华
(北京航空航天大学计算机科学与技术学院　北京 100083)

摘　要　本文以 RTEMS 为例, 通过对源代码的剖析, 深入探讨嵌入式操作系统对多处理器系统的支持。主要通过多机支持模块的数据结构, 实现机制, 以及流程来阐述。

关键词　嵌入式　多处理器　RTEMS

THE MULTIPROCESSOR SUPPORT OF EMBEDDED OPERATING SYSTEM

Bao Zheng　Li Hongbing　Jin Huihua
(*Institute of Computer Science and Technology, Beijing University of Aeronautics and Astronautics, Beijing 100083*)

Abstract　In this paper, We discuss the multiprocessor support of embedded system, illustrating with an example of RTEMS. We demonstrate the data structure, mechanism, and flow of multiprocessor module to reveal the principle.

Keywords　Embedded　Multiprocessor　RTEMS

1　引　言

操作系统(OS)在嵌入式领域的使用日益广泛。其中大多数嵌入式系统都是单机环境, 不需要 OS 对多处理器(MP)支持。近年来随着并行计算等的兴起, 嵌入式操作系统(EOS)也常常出现在 MP 环境里。在纷繁复杂的 EOS 中, pSOS、VxWorks、QNX、Nucleus Plus、RTEMS 等都支持 MP。各种 EOS 特点各异, 对 MP 的支持也不尽相同, 但共性也是十分明显, 比如, 它们对 MP 的支持都是作为一个独立、可裁剪的模块出现。本文以 RTEMS 为例, 分析探讨嵌入式系统对 MP 支持是如何实现的, 希望对今后的嵌入式多机应用软件开发有所帮助。

2　RTEMS 简介

RTES(多处理器系统实时内核: Real Time Executive for Multiprocessor Systems), 是一个为嵌入式应用提供高性能环境的实时内核。它具有如下特点: 支持多任务; 支持同构或异构多处理器系统; 支持事件驱动、基于优先级、占先的调度算法; 具有可选的实时调度算法; 支持任务间的通信和同步; 支持 Priority Inheritance 算法; 快速响应的中断管理; 支持动态存储器分配; 具有高级别用户配置能力。

RTEMS 的所有服务都是建立在对象机制之上, 比如线程、信号灯、消息队列, 这些都是对象的一个子类, 所有操作也都是以对象为基础, 各种不同的对象有自己特有的一些操作, 这一点很像面向对象技术。

3　RTEMS 的多处理器支持

3.1 多处理器管理器

RTEMS 提供简单和灵活的实时多处理器功能。RTEMS 的内核既适用于紧耦合, 又适用于松耦合目标系统硬件的配套。此外, RTEMS 还支持由同构和异构混合的处理器和目标板组成的系统。

RTEMS 系统内核的一个主要设计目标是, 系统可以跨越目标硬件配置的边界。RTEMS 通过为应用程序软件提供一个处理器节点之间边界透明的目标系统的逻辑形式, 来实现这个目标。因此, 应用系统开发者可以将任务、消息队列、事件、信号、信号量和存储器块这样的对象指定为全局对象。随后, 任何任务都可以访问这些全局对象, 而不管这些对象和任务的物理位置如何。

RTEMS 自动确定被访问的对象位于哪一个处理器上, 并且执行访问远程对象所必需的操作。简单地说, RTEMS 允许整个系统——硬件和软件, 被视作逻辑上的一个单一系统。

RTEMS 多处理器的执行模型是多指令流、多数据流(MIMD)。在这个模型下, 每个处理器所执行的代码是独立于其它处理器的。由于使用这种并行机制, 应用系统设计者可以很容易保证系统行为的确定性。

3.2 节　点

在 RTEMS 的多机模式中, 应用程序级有可能是主从模式, 但在操作系统层来看, 每个处理器都是平等的, 它们执行相同版本和配置的操作系统, 唯一的区别就在于处理器的节点标识号, 定义为 NODE_NUMBER, 应用程序的开发者在编译时, 为每个处理器定义一个唯一的值, 这样每个处理器拥有一个属于自己的处理器节点号, 从 1 号起编, 以后依次为 2, 3, ……, 这样, 处理器之间的操作系统版本差别达到最小, 简化了移植工作量。

3.3 各处理器之间的通信

由于不同的应用系统的处理器间通信方式千差万别, RTEMS 屏蔽了所有实现通信的细节, 用一个 MPCI 表包括了所有通信用到的例程, 其定义如下:

收稿日期 2003－04－01。鲍峥, 硕士生, 主研领域: 嵌入式计算机体系结构。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

```
typedef struct{
  unsigned32    default_timeout;   /*in ticks */
  unsigned32    maximum_packet_size;
  MPCI_initialization_entry     initialization;
  MPCI_get_packet_entry     get_packet;
  MPCI_return_packer_entry     return_packet;
  MPCI_send_entry     send_packet;
  MPCI_receive_entry     receive_packet;
  }MPCI_Control;
```

应用软件开发人员据实际的硬件实现填写该表，其中 get_packet、return_packet、send_packet、receive_packet 是最主要的例程，分别表示启动一个传输，终止传输，发送数据报和接受数据报，它们是通信的具体实现。这种类似分层协议的通信方式在最大程度上使上层脱离了硬件的束缚。

每个数据报文具有相同的报头，记录的是该数据报文的信息，其具体定义及各部分说明如下：

```
typedef struct{
  MP_packet_Classes   the_class;   /*该数据报文操作的对象类型 */
  Objects_Id   id;   /*操作对象的标识 */
  Objects_Id   source_tid;   /*产生该数据报文的线程标识 */
  Priority_Control   source_priority;   /*产生该数据报文的线程标识 */
  unsigned32   return_code;   /*远程操作的返回值 */
  unsigned32   length;   /*数据报文长度 */
  unsigned32   to_convert;   /*表示数据报实际长度，即去掉报头的数据长度 */
  Watchdog_Interval   timeout;   /*超时长度 */
}MP_packet_Prefix;
```

相同结构的报文头部可使报文处理例程抽取各种不同报文处理的共性，简化了代码。

它有三种类型的数据报文：

发送处理数据报 当全局对象创建、删除等操作时，需要发送处理数据报。线程填写了发送线程的标识，调整报文的大小之后，调用 BSP(Board Support Packet，板级支持程序)提供的发送函数。

发送请求数据报 当请求全局对象资源时，需要发送请求数据报。请求线程发送数据报以后，线程需挂起等待，等待对方的应答报文的到来。

发送应答数据报 当线程处理完远程请求报文之后，需要发送应答报文，指明发送线程的标识，调用 BSP 提供的发送函数。

每个处理器运行一个接收线程，并且让它无限循环地侦听通信数据，称为接收服务线程，可以保证在有数据报文到达时能够及时处理，不同的对象都有自己的数据报文处理函数，服务线程会根据数据报文内容中标识的对象类型，执行事先注册的不同对象的处理函数。然后再根据数据报自身的类型，执行相应的操作。

系统启动后，每个处理器都会向其它所有处理器发送一个广播确认消息，内容包括整个系统的最大节点数，可以生成的最大全局对象数，节点收到的信息和自己所记录的不符，则会报错。这个广播消息可以看成是一次握手动作。这个过程之后，就可以保证每个处理器记录的关于整体系统结构的信息是一致的。

3.4 远程操作

在 RTEMS 中，多处理器之间的协调同步主要体现在对全局对象的处理上，如何使全局对象具有唯一性，且使各个处理器都能独立、互斥地对其进行处理，如何使应用开发人员可以像操作局部对象那样操作全局对象，成为多处理器支持模块主要要解决的问题。

所有的对象都有一个 32 位的全局唯一的 ID，高 6 位标识对象类型，有线程、信号灯、消息队列、定时器等，中间 10 位标识其处理器的节点号，低 16 位是具体的索引，所以通过每个对象的 ID，就可以知道其所在节点，就可以判断是否是远程对象。

31 26	25 16	15 0
对象类型	处理器节点号	对象索引值

图 1 对象标识的各位段的意义

以信号灯例，如果为单机，其请求操作过程大致如下：

```
线程 1
sem_create();/ *创建信号灯 */
sem_obtain();/ *信号灯获取 */
obtained; mutex_process; / *成功，执行原子操作； */
sem_release();/ *信号灯释放 */
thread_dequeue();/ *从线程队列中取出一个线程，使它恢复运行 */
other_process();/ *其它操作 */
线程 2
sem_obtain();/ *信号灯获取 */
thread_enqueue();/ *进入线程队列等待 */
/ *阻塞中 */
/ *被激活，继续运行 */
obtained; mutex_process; / *成功，执行原子操作 */
sem_release();/ *信号灯释放 */
sem_unlock();/ *将信号灯置为可用 */
other_process;/ *其它操作 */
```

过程示意图如图 2 所示。

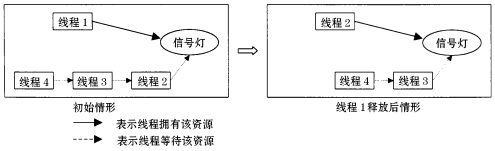

图 2 单机的信号灯操作原理图

在多机系统中，所有的信号灯对象分为局部对象和全局对象，但它们都是存在于某个处理器上的，亦即是由某个特定的处理器创建的。每个处理器都有两个信号灯表，分别记录的是全局信号灯和局部信号灯(对于其它资源对象也是一样的)，各个处理器的全局信号表都是一样的，因为它们记录的内容都是整个系统的全局信号灯。当一个全局信号灯创建时，处理器除了在本地填写这两个信号灯表，还要向其它所有节点发送创建通知，其它处理器收到通知后，将信息记录到自己的全局表；同样在它被删除时，也要发送通知，通知其它处理器将表中的记录删除掉。

对于局部对象，所有操作都只能在本地进行，具有全局属性的对象才可以让远程的处理器操作。当某处理器生成一个新的全局信号灯，对于本地线程来说，访问该信号灯和访问其他局部信号灯没有区别，单机的信号灯处理模块已经可以保证正确的访问操作，而远程线程对它的访问，由于多了一层处理器通信，

(下转第 51 页)

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

电路的最优化设计。

4 结束语

用 PSpice 对电路进行仿真分析, 既能降低设计成本, 又能提高设计效率。本文提出的用 PSpice 的参数扫描分析和优化设计相结合的方法对电路进行最优化设计, 可以进一步地提高电路设计的可靠性和精确度, 实践证明这种方法在电路的最优化设计中具有很高的实用价值, 并将得到广泛的应用。

参 考 文 献

[1] 贾新章、郝跃, OrCAD/PSpice9 实用教程, 西安电子科技大学出版社, 1999.

[2] 郑光钦, 全能 OrCAD 混合电路仿真, 中国铁道出版社, 2000.

[3] 王筱颖, 模拟电路导论, 高等教育出版社, 1986.

(上接第 30 页)

它的过程就复杂得多, 需要向远程发送请求, 等待回应, 并在远程生成代理线程, 代理线程进入等待队列, 这样就可以享受和远程线程同等"待遇"。

具体过程表示如下:

节点 1 的线程 1

```
sem_create();/*创建信号灯*/
sem_obtain();/*信号灯获取*/
obtained;mutex_process;/*成功,执行原子操作;*/
sem_release();/*信号灯释放*/
th=thread_dequeue();/*从线程队列中取出一个线程*/
if(is_proxy(th)){
release_proxy(th);/*释放代理*/
send_response();/*发送回应*/
}
other_process();/*其它操作*/
节点 1 的接收服务线程
for(;;){
packet_receive();/*等待接收数据*/
/*有数据到*/
read_packet();/*确定数据内容*/
sem_obtain();/*信号灯释放*/
proxy_enqueue();/*生成代理,进入队列
                  等待*/
}
节点 2 的线程 R
sem_obtain();/*信号灯获取*/
lookup_table();/*查表发现是远程节点,确定
                 其所在节点*/
send_request();/*发送获取请求*/
/*阻塞中*/
/*收到远程的回应,继续运行*/
sem_release();/*信号灯释放*/
thread_dequeue();/*从线程队列中
取出一个线程,使它恢复运行*/
other_process();/*其它操作*/
```

下面以图 3 表示的是信号灯远程请求操作的一般过程示意。

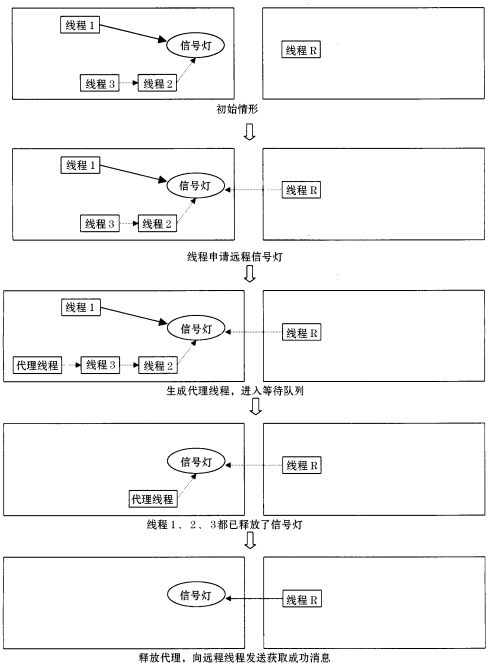

图 3　远程对象操作过程

线程释放信号灯时, 过程相对简单, 只需要发送一个释放操作命令数据报。远程信号灯收到后, 执行本地的释放操作即可。

相对于本地操作, 有两点需要注意:

1) 当一个等待远程资源的线程被删除或者重新启动时, 不能简单的将它的资源释放了事, 因为在另一个处理器上还有一个它的代理在等待队列中, 这时需要它发送一个释放代理的消息, 释放远程代理占用的资源。

2) 当信号灯被删除或者复位的时候, 处于等待队列中的线程应该全部返回失败, 对于本地线程和远程代理应该区别对待, 本地只需要在线程控制块理填入返回值, 并使其恢复到就绪状态, 而远程则需要发送获取资源失败的应答消息。

4 总　结

嵌入式操作系统对多机系统的支持只能是系统级的, 这些底层的系统调用提供了多处理器编程手段。RTEMS 提供的多处理器支持环境已经可以让开发者像开发单机系统那样开发多机系统, 而如何让各个处理器之间协调地完成并行计算、并行处理, 还要靠开发人员在应用程序中实现。

通过对 RTEMS 多处理器模块源代码的深入分析, 使我们了解了系统的内部结构与实现原理, 这就有助于更好的开发多处理器的系统和应用软件。

参 考 文 献

[1] Donald W. Gillies, 《Real-Time System References and Reading Materials》, Elec 494 Course on Real-Time Software-Spring, 1995.

[2] 《RTEMS Development Environment Guide》, OAR Corporation, Edition 4.0.0, for RTEMS 4.0.0, October 1998.

[3] 《RTEMS Porting Guide》, OAR Corporation, Edition 4.0.0, for RTEMS 4.0.0, October 1998.

[4] 毛德操、胡希明著, Linux 内核源代码情景分析, 浙江大学出版社, 2001.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net